2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Tight Motion Planning by Riemannian Optimization for Sliding and Rolling with Finite Number of Contact Points

Dror Livnat[1] Michael M. Bilevich[1] Dan Halperin[1]

***Abstract*— We address a challenging problem in motion planning where robots must navigate through *narrow passages* in their configuration space. Our novel approach leverages optimization techniques to facilitate sliding and rolling movements across critical regions, which represent semi-free configurations, where the robot and the obstacles are in contact. Our algorithm seamlessly traverses widely free regions, follows semi-free paths in narrow passages, and smoothly transitions between the two types. We specifically focus on scenarios resembling 3D puzzles, intentionally designed to be complex for humans by requiring intricate simultaneous translations and rotations. Remarkably, these complexities also present computational challenges. Our contributions are threefold: First, we solve previously unsolved problems; second, we outperform state-of-the-art algorithms on certain problem types; and third, we present a rigorous analysis supporting the consistency of the algorithm. In the Supplementary Material we provide theoretical foundations for our approach. The Supplementary Material and our open source software are available at `https://github.com/TAU-CGL/tr-rrt-public`. This research sheds light on effective approaches to address motion planning difficulties in intricate 3D puzzle-like scenarios.**

I. Introduction

Motion planning plays a pivotal role in the field of robotics, as well as in assembly automation, computer graphics, and other fields. It is a fundamental component in the generation of efficient and collision-free trajectories for robotic systems. A challenging sub-domain is scenarios where robots must operate within tight settings, engage in physical contact with objects or surfaces, or traverse through narrow passages[2]. These enable a variety of applications, such as multi-arm motion planning in close proximity, solving 3D puzzles, and rearrangement of products in the fridge.

Over the past three decades, sampling-based methods have emerged as a cornerstone in the field of motion planning, contributing significantly to the success of various applications. Among these methods, Probabilistic Roadmaps (PRM) [1] and Rapidly-Exploring Random Trees (RRT) [2] have stood out as fundamental techniques that guarantee probabilistic completeness. They were followed by numerous variants aimed at improving parameters such as computational speed [3]–[5], path quality (e.g. optimality guarantees) [4, 6], or guaranteeing anytime performance [5, 7].

Despite the widespread success of sampling-based methods, navigating through narrow passages remains a persistent and substantial challenge to this date. Various strategies were proposed to address this intricate issue for dedicated tasks, such as non uniform sampling designed for the task [8, 9], learning based approaches [10, 11], and approaches based on geometry features of the moving elements [12, 13].

Compliant motion planning [14]–[18] is one approach to cope with movement while in contact with obstacles. It is primarily a motion planning concept, focusing on generating suitable trajectories considering compliance and flexibility, while the execution of compliant motions requires close integration with control algorithms to ensure that the robot behaves as planned while adapting to its environment.

Another direction that has demonstrated promising results in cases with tight constraints, such as solving puzzles, is Physical-simulation based methods [19, 20].

A different approach, coping with similar problems, uses search along constraints manifold, either by maintaining an atlas of manifolds to cope with loop closure constraints [21]–[24], or by providing task specific projection functions [25].

A more recent work, by Zhang et al. [13], focuses on rigid disentanglement puzzles, which are interesting because they are specifically designed to be difficult to take apart for humans due to the non-trivial translation and rotation. Apparently, these are challenging for motion planning algorithms as well. They use local features of each component to identify suspected tunnels, and implement it on 3D puzzle, starting from the classic alpha puzzle and continue to more complicated ones. Their algorithm is limited to puzzles which posses that local feature and solve many of them successfully.

Although the above methods do solve many new problems previously unsolved, or solved inefficiently, the general family of motion planning problems in mixed settings (wide and tight) is still challenging and in many of them the above approaches fail.

In this work, we propose a novel approach to motion planning that traverses directly on critical sub-surfaces in the configuration space, allowing for explicitly searching configurations for which the robot slides and rolls over the obstacles.

[1]Blavatnik School of Computer Science, Tel-Aviv University, Israel. Work on this paper has been supported in part by the Israel Science Foundation (grant nos. 1736/19 and 2261/23), by NSF/US-Israel-BSF (grant no. 2019754), by the Israel Ministry of Science and Technology (grant no. 103129), by the Blavatnik Computer Science Research Fund, and by the Yandex Machine Learning Initiative for Machine Learning at Tel Aviv University.

[2]By "narrow passages" as a major problem in sampling-based motion planning we typically refer to passages in the underlying *configuration space*, which we define later. Intuitively, narrow passages arise when a robot moves in very close proximity to the obstacles or to other parts (links) of itself. See Subsection VI-C for the *Elk* puzzle, where the obstacle seems wide open in the workspace, while in the configuration space the movement is extremely tight.

Fig. 1. Example of a solution our method produced for the az (alpha-z) puzzle.

Our approach combines the strength of RRT in free areas, with machinery that implements rolling and sliding around any contact points that are encountered during the search. It exhibits promising results in solving problems characterized by a complex interplay of narrow passages and more accessible regions. The search in the tight areas, near the obstacles, is in fact reduced to a traversal on a lower dimensional manifold. This reduction of the search space significantly enhances the likelihood of sampling feasible paths through narrow passages, thus addressing a persistent challenge in motion planning. The results (shown in Section VI) are three fold. First, we outperform state-of-the-art algorithms on certain problem types; second, we demonstrate solutions for types of problems that were not covered in previous works; and third, we present a rigorous analysis of the algorithm consistency, that is, continuously sliding and rolling over the obstacles.

II. Preliminaries

A. The Signed Distance Function (SDF)

Given some compact manifold $M \subseteq \mathbb{R}^3$, the *Signed Distance Function (SDF)* (also referred to as the *Signed Distance Field*) $F_M : \mathbb{R}^3 \to \mathbb{R}$ maps a point $p \in \mathbb{R}^3$ to its closest distance to M. The distance is signed positive if the point p is outside M and negative if p is inside M. See [26] for a survey of classical techniques for computing SDFs. Instead of computing the SDF directly, one might evaluate the SDF only on a grid, and use that grid as an approximation for the SDF value when querying new points [19, 27].

Recently, there has been an emergence of deep-learning methods (e.g., [28]–[32], to name a few) for approximating the SDF of a given object. Such methods [28, 32] strive to be a compact replacement for traditional computer representations of 3D objects. We note that for such representations, since neural networks utilize GPUs, we are able to query multiple (three-dimensional) points at once. A related technique is Neural Radiance Fields (NeRFs) [33, 34], which learn for a given scene a mapping that for some 5D point (x, y, z, θ, ϕ), returns the RGB color and density σ of a ray emanating from (x, y, z) in direction (θ, ϕ) with the scene. In this work we use SDFs, as we show that we only require the signed distance from the obstacles.

B. SDF as a collision detection tool

One use of SDFs is for computing collision detection. Collision detection is the task of determining, between any two objects, if any collision occurs. We can represent one object as a collection of points, and evaluate each point in the SDF of the second body. If there are any points with negative distance, they are in penetration and a collision occurs. Conversely, if all points have non-negative signed distance then there are no collisions. See [35]–[39] for examples.

III. Problem Statement

In this work we deal with a setting of two entangled rigid bodies[3] $M_1, M_2 \subseteq \mathbb{R}^3$ (see Figure 2), which we wish to disassemble. By *entangled* we refer to configurations where a straightforward motion (or a simple sequence of motions) does not separate the objects. Regarding *disassemble* we note that it suffices to choose an arbitrary q_{goal} such that the convex hull of the robot is disjoint from the convex hull of the obstacles, in cases where q_{goal} is not provided.

We assume that M_1 is fixed in some pose in $SE(3)$, and that M_2 is the object we wish to move from configuration q_{start} to configuration q_{goal} in $SE(3)$. Henceforth, we refer to M_1 as the *obstacle(s)*, and to M_2 as the *robot* or the *free-flying object*. Without loss of generality, we will also assume that the fixed pose of M_1 in $SE(3)$ is the identity transformation. The configuration space, C-space for short, of the robot M_2 is the set of rigid-body transformations, denoted by $\mathcal{C} = SE(3)$. Let $q \in \mathcal{C}$ be a configuration of M_2, then we denote by A_q its corresponding transformation matrix. The *forbidden region* $\mathcal{C}_{\text{obs}} \subseteq \mathcal{C}$, where the robot penetrates the obstacles, is defined as

$$\mathcal{C}_{\text{obs}} = \{q \in \mathcal{C} \mid \text{int}(A_q(M_2)) \cap \text{int}(M_1) \neq \emptyset\} \ , \quad (1)$$

where $\text{int}(\cdot)$ is the interior of a set. The *semi-free region* $\mathcal{C}_{\text{semi-free}} \subseteq \mathcal{C}$, where the robot and the obstacles overlap only at their boundaries, is defined as

$$\mathcal{C}_{\text{semi-free}} = \{q \in \mathcal{C} \setminus \mathcal{C}_{\text{obs}} \mid A_q(M_2) \cap M_1 \neq \emptyset\} \ . \quad (2)$$

We finally define the *free region* as all configurations that yield no intersection for the robot with the obstacles,

$$\mathcal{C}_{\text{free}} = (\mathcal{C} \setminus \mathcal{C}_{\text{obs}}) \setminus \mathcal{C}_{\text{semi-free}} \ . \quad (3)$$

In this work we focus our search mostly on the semi-free region $\mathcal{C}_{\text{semi-free}}$.

Our problem is thus finding a collision free path from $q_{\text{start}} \in \mathcal{C}_{\text{free}}$ to $q_{\text{goal}} \in \mathcal{C}_{\text{free}}$, i.e., to find some path $\gamma : [0, 1] \to \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$ such that $\gamma(0) = q_{\text{start}}$ and $\gamma(1) = q_{\text{goal}}$.

In this work we restrict ourselves to paths γ that are piecewise linear, that is, a sequence:

$$q_0 = q_{\text{start}}, q_1, \ldots, q_{T-1}, q_T = q_{\text{goal}}, \quad (4)$$

[3]Compact sub-manifolds of $\mathbb{R}^3$, with or without a boundary.

Fig. 2. The az (alpha-z) puzzle. The silver object is M_1 and is referred to as the obstacle. The gold object is M_2 and is referred to as the robot.

and the line segment between each pair of consecutive configurations in the sequence, such that γ is contained in $\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{semi-free}}$.

Finally, in this work we identify the C-space $SE(3)$ with $\mathbb{R}^6$, such that

$$q \simeq (x, y, z, \theta_x, \theta_y, \theta_z) \ , \tag{5}$$

where (x, y, z) is the translation of q, and $(\theta_x, \theta_y, \theta_z)$ is the rotation of q, with $\theta_x, \theta_z \in [-\pi, \pi)$ and $\theta_y \in [-\pi/2, \pi/2)$, represented as Euler angles[4].

IV. Algorithm Overview

We present an algorithm for motion planning of a non-convex rigid body, which we refer to as the *robot*, in *mixed tight and wide* settings. That is, we strive to find a collision-free path of motion from configuration q_{start} to q_{goal} in problems for which any solution path passes through both tight and wide regions of the C-space. By *tight* we mean subsets of the free C-space, which have measure of almost zero, i.e., small enough such that randomly sampling points in that portion is not viable[5]. In contrast, by *wide* we refer to all other areas of the free C-space. That is, portions of the free C-space corresponding to motion of the robot with clearance significantly larger than zero. Due to the nature of the algorithm, it works even better in cases where the entire path is tight or the entire path is wide, because these are easier cases, hence we focus on the more challenging combination of wide and tight.

Zhang *et al* [13] refer to the narrow and wide portions of the C-space as *tunnels* and *bubbles*, respectively; we will use tunnels and narrow (regions) interchangeably.

We call the new algorithm *TouchRollRRT (TR-RRT)* as in the *extend* phase, it uses the SDF of the obstacles and its derivatives to maintain two or more contact points between the robot and the obstacles at (almost) all times. By contact points we refer to points on M_2 which have signed distance between $-\delta$ and δ to the boundary of the obstacles M_1, for some small $\delta > 0$. These motions, which we call "sliding" and "rolling", are described in detail in Section V.

Building the tree in TR-RRT uses uniform samples in $SE(3)$ followed by linear steps towards them in C-space.

[4]Note that Euler angles should be handled carefully to address singularities. See [40, Section 4.2] for details.

[5]If we were able to carry out all computations using exact numbers, say, using special algebraic varieties, our method would have been able to tackle also cases of measure exactly zero.

Algorithm 1 TouchRollRRT

$V \leftarrow \{q_{\text{start}}\}, E \leftarrow \{\}$
for $i = 1, \ldots,$ ITERATIONS **do**
  if $i \mod N_{\text{rotations}} == 0$ **then**
    randomRotate()
  end if
  $s \leftarrow$ sample()
  $q_{\text{near}} \leftarrow \arg\min(\text{distance}, s, V)$
  $v \leftarrow \text{direction}(q_{\text{near}}, s)$
  $qs \leftarrow \text{extend}(q_{\text{near}}, v)$
  if isEdgeValid$(qs[-1], q_{\text{goal}})$ **then**
    return getPath$(V, E, q_{\text{start}}, q_{\text{goal}})$ ▷ path found
  end if
  for $j = 2, ..., len(qs)$ **do**:
    $V \leftarrow V \bigcup \{q_j\}$
    $E \leftarrow E \bigcup \{[q_{j-1}, q_j]\}$
  end for
end for
return None

Hence, wide areas are covered by the tree as effectively as in standard RRT. Whenever such a step hits an obstacle (more precisely, when a point $p \in M_2$ is closer than δ to M_1) then the direction of the movement v is changed for p to roll or slide over M_1. This way, the tree grows effectively around the obstacles in a subspace of lower degree, and finds tunnels, which in this sub-space have significantly larger probability to be discovered. As there is typically no simple motion that leads the robot to q_{goal}, then in every motion direction, the robot gets into contact with the obstacles sooner or later, and the sliding mechanism ensures that the contact is maintained from that point and on.

When the steering gets out of a narrow passage, the number of contact points that can be simultaneously maintained drops down, and correspondingly the dimension of the manifold where we sample goes up, possibly to six, which is the dimension of the C-space.

By maintaining, where possible, at least two contact points between the robot and the obstacles, the number of degrees of freedom (DOFs) of the robot decreases from six to at most four, which in turn makes the search on the underlying C-space tractable. The flow of the entire algorithm is presented in Algorithm 1.

We thus get a path of motion which is valid (up to some slight penetrations of depth at most $\delta > 0$) for the SDF representation of the obstacles. We then try to transform this path into a valid path of motion for the polyhedral representation of the obstacles, by retracting the path points that have a small penetration back into free space; this is described in detail in Section V.

In the remainder of this section we describe the general flow of the algorithm. In section V we dive into the rolling and sliding mechanism, which is the main contribution of this work. In the Supplementary Material we describe the theoretical framework upon which our algorithm is based.

We first assume that we have an SDF representation of the obstacles as detailed in Section II-A. The flow of TR-RRT is similar to the RRT Algorithm [2]. It starts with an empty graph (V, E) and adds q_{start} to it. Then, for a predefined number of iterations, it repeats the following. It uniformly samples a configuration $s \in SE(3)$, finds the closest node in the graph to s, q_{near}, and calculates the direction v from q_{near} to s. Once every predefined number of iterations ($N_{\text{rotations}}$), the entire scene is rotated at a random angle (randomRotate). That way, any gimbal lock that might have appeared is removed from the way.

The major difference between the standard RRT and our algorithm appears in the *extend* procedure. At a high level, the procedure does the following. Assuming that the current configuration q_{near} lies near some sub-manifold of $SE(3)$ which represents all configurations having some amount of contact points with the obstacles, we use the direction v to locally advance and produce a sequence of points, which also lie in close proximity to that sub-manifold. A series of such steps is repeated and the resulting free or semi-free configurations are kept in qs. Then, we test if there is a direct and collision-free edge towards the goal configuration q_{goal}. If such an edge exists, the algorithm quits with a valid path. At the last step of each iteration, all nodes in qs and edges connecting them are added to the graph.

The pseudo-code for the TR-RRT algorithm is presented in Algorithm 1.

V. The Slide Step

In this section we present our novel algorithm for sliding and rolling the bodies, by looking at constraint sub-manifolds of configurations that yield contacts between the robot and the obstacles. Assuming we already know the current points of contact between the robot and the obstacles, we implicitly define a sub-manifold of $SE(3)$ of all configurations for which the given points, *or their neighbors*, remain in contact. We then show a method for locally advancing, towards a given direction, in close proximity to that manifold.

First, we define the *contact-SDF* functions, which will implicitly define our *critical* sub-manifolds of $SE(3)$.

Definition 5.1: Let $F_{M_1} : \mathbb{R}^3 \to \mathbb{R}$ be the SDF of the obstacles M_1. Let p be a point in M_2. The function $F^p_{M_1} : SE(3) \to \mathbb{R}$, which we refer to as the *contact-SDF* (of that point p) is defined as:

$$F^p_{M_1}(q) \coloneqq F_{M_1}(\varphi(q) \cdot p) \ , \tag{6}$$

with $\varphi(q) = A_q$ the corresponding 4×4 matrix of the configuration $q \in SE(3)$.

Notice that when $F^p_{M_1}(q) = 0$, the point $p \in M_2$ touches the boundary of the obstacles M_1 when placing M_2 at the configuration q.

Definition 5.2: Let $U(p) \subseteq M_2$ be a small open neighborhood of $p \in M_2$. Then we define the *weak-contact-SDF* (of that point p as $F^{U(p)}_{M_1} : SE(3) \to \mathbb{R}$:

$$F^{U(p)}_{M_1}(q) \coloneqq \min_{p' \in U(p)} F^{p'}_{M_1}(q) \ . \tag{7}$$

Given k-points $p_1, \ldots, p_k \in M_2$ for some $k \in \mathbb{N}$, we define the *multiple weak-contact-SDF* as $F^{U(p_1),\ldots,U(p_k)}_{M_1} : SE(3) \to \mathbb{R}^k$,

$$F^{U(p_1),\ldots,U(p_k)}_{M_1}(q) = \left(F^{U(p_1)}_{M_1}(q), \ldots, F^{U(p_k)}_{M_1}(q)\right) \ . \tag{8}$$

We refer to those $p_1, \ldots, p_k$ as the *contact points*. Assume that there is only a finite number of contact points. Then we note that when a configuration $q \in SE(3)$ has $F^{U(p_1),\ldots,U(p_k)}_{M_1}(q) = 0$, for all the original contact points $p_1, \ldots, p_k$ there is some point in their open neighborhood (on the robot M_2) which lies on the boundary of the obstacles M_1, and (since $F^{U(p)}_{M_1}$ is defined as $\min$) no other point in any of these neighborhoods touches or penetrates the obstacles.

Lemma 5.1: Assume the following holds:

- There exists $1 \leq k \leq 6$ contact points $p_1, \ldots, p_k$ on the robot M_2 .
- The function $G_{M_1} : \mathbb{R}^3 \times SE(3) \to \mathbb{R}$ defined by

$$G_{M_1}(p, q) = F_{M_1}(\varphi(q) \cdot p) \tag{9}$$

  is continuously-differentiable, and $\nabla_{(p,q)} G_{M_1} \neq 0$ for all $(p, q) \in \mathbb{R}^3 \times SE(3)$.
- For all points $p \in M_2$, the function $F^{U(p)}_{M_1}$ is continuously-differentiable.

Then the set

$$M^F_{U(p_1),\ldots,U(p_k)} \coloneqq (F^{U(p_1),\ldots,U(p_k)}_{M_1})^{-1} \left(\{\vec{0}\}\right) \tag{10}$$

is a $(6-k)$-manifold, which we refer to as the *(weak)-contact critical manifold*.

Note that the assumptions above need not necessarily hold. Still, we use the following analysis to derive an algorithm that works well in practice.

Assuming we are able to compute the gradient $\nabla_p F_{M_1}$, it is straightforward to compute also the gradient of the contact-SDF $\nabla_q F^p_{M_1}$. In order to compute the gradient of the weak-contact-SDF, we notice the following:

Lemma 5.2: Let $q \in SE(3)$ and some $p \in M_2$, and assume that $p'_* = \arg\min_{p' \in U(p)} F^{p'}_{M_1}(q)$. Then:

$$\nabla_q F^{U(p)}_{M_1} = \nabla_q F^{p'_*}_{M_1} \ . \tag{11}$$

We would also like to *retract* a configuration $q \in SE(3)$ onto the weak-contact critical manifold. There are many algorithms for retraction onto a manifold [41]. In this work, we do so by minimizing the function $\mathcal{L}_{U(p_1),\ldots,U(p_k)} : SE(3) \to \mathbb{R}$,

$$\mathcal{L}_{U(p_1),\ldots,U(p_k)}(q) = \sum_{i=1}^{k} \left(F^{U(p_i)}_{M_1}(q)\right)^2 \ . \tag{12}$$

Non-convex optimization is an extensively researched area, e.g., [42]–[45], and it is beyond the scope of this work to give a comprehensive review. In our work, we use the Adam optimizer [46], which is a common choice when optimizing with neural networks, and performs well empirically [47]. We note that minimizing this function $\mathcal{L}_{U(p_1),\ldots,U(p_k)}$ yields a configuration in $SE(3)$ for which all

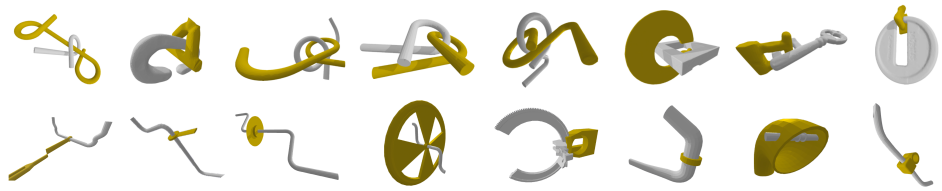

Fig. 3. All instances from the data-set presented in [19] that our method is able to solve consistently.

points $p_1, \ldots, p_k$ will have a neighboring point on M_2 that lies in very close proximity to the boundary of the obstacles. We denote this *retraction* operator by $\mathcal{R}_{M^F_{U(p_1),\ldots,U(p_k)}}$.

We are now ready to present the slide step[6], which generates a sequence of configurations $q_1, \ldots, q_T$ such that for all $t = 1, \ldots, T$, each q_t lies in close proximity to the weak-contact critical manifold. The step size $\eta > 0$ is predefined, hyper-parameter.

For each iteration, assume we have a configuration q_t. We take the direction $v \in SE(3) \simeq \mathbb{R}^6$, which is the steering direction from q_{near} towards a randomly sampled s (see Algorithm 1), and project it onto the tangent space $T_{q_t} M^F_{U(p_1),\ldots,U(p_k)}$, which is the linear space that is orthogonal to the gradients $\nabla_q F^{U(p_i)}_{M_1}$. Denote this projection by v_t. We then take a small (η) step from q_t in direction v_t, to get $q'_{t+1} = q_t + \eta \cdot v_t$. We then retract q'_{t+1} back towards the critical manifold, to get the next point in our sequence, q_{t+1}. The pseudo-code, analysis and further details for this section are presented in the Supplementary Material.

VI. Experiments and Results

In this section, after providing details on our experimental setup, we present comparisons with state-of-the-art benchmarks and results, on a subset of 16 instances from the benchmark presented in [19].

A. Implementation Details

Our open source software is written in Python and is available at `https://github.com/TAU-CGL/tr-rrt-public`. The code and all evaluations were run on a Linux machine with Intel Core i7-13700K CPU and NVIDIA RTX 3090 GPU.

The SDF for approximating distances from each obstacle is implemented as a neural network with $N = 12$ hidden layers with $H = 64$ neurons in each layer, with ReLU activation, and with $\tanh$ activation for the final layer. Our network structure, data-set and training method are performed as described in [32]. All models are scaled beforehand such that they fit in the unit sphere $\mathbb{S}^2 \subseteq \mathbb{R}^3$. The choice to implement our SDF as a neural network rather than a grid is based on the accuracy-size trade-off. In order to achieve the 0.0005 accuracy which is achieved by the network, a grid would require about 13 GB, which seems too large. The network weights are about 204 KB.

To perform collision detection and to find contact points, we do as follows. In the beginning of the algorithm we sample $2,500$ points on the boundary of the robot M_2. Then, for a configuration $q \in SE(3)$, we transform those sampled points by q and evaluate the SDF of M_1. Points that have value of signed distance less than a threshold $\delta = 0.002$ are considered contact points. Points with signed distance that is less then $-\delta$ are considered in penetration, and we report that there is a collision. When checking if an edge $[q_1, q_2] \subseteq SE(3)$ has collisions, we split it in increments of δ, and perform collision detection for each point along the edge. Note that since our SDF is represented as a neural network, we are able to evaluate all sample points in parallel. When computing a gradient for a contact point, as described in Section V, we use Autograd [48].

Since there are many contact points that are in close proximity to each other, yielding similar gradients, we might lose degrees of freedom which we have not intended. That is, when projecting a direction v onto the tangent space of a critical manifold, we might instead project to a linear sub-space of that tangent space. Hence after computing the gradients for each contact point, we cluster together similar gradients using K-means algorithm [49, 50]. Note that by clustering based on the gradients proximity rather than contact points proximity the direction v' calculation becomes much more accurate. This is due to the fact that close contact points may require significantly different gradients on the one hand, while far away contact points, even on two side of a tunnel, may generate (almost) same (or antipodal) gradients.

B. Evaluation Against Baseline Methods

We evaluated our method on the data-set provided by Tian et al. [19]. More specifically, we deal with their *rotational assemblies*, which are problems that require a combination of simultaneous translation and rotation in order to disassemble the parts. This data-set is comprised of overall

TABLE I

Success rate (%) of our method against the baseline methods. Overall, we achieve the best success rate. The `Puzzles` and `Others` benchmarks have 8 instances each.

Method	`Puzzles`	`Others`	Overall
BK-RRT	75.0	87.5	81.25
TIAN-ET-AL	87.5	87.5	87.5
TR-RRT (Ours)	**100**	**100**	**100**

[6]This method can be described as an implementation of gradient descent optimization on Riemannian manifolds. More details can be found in the Supplementary Material.

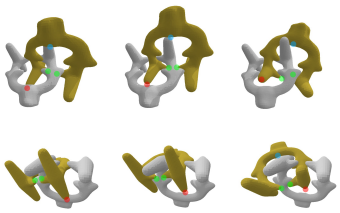

Fig. 4. Illustration of one of the tunnels required to solve the Elk puzzle. The robot is in yellow, while the obstacle is white. In red, green and blue are different contact points, throughout the motion. In the top row, from left to right, are the configurations of the robot before, during and after the tunnel, respectively. In the bottom row are the same configurations, from a different viewpoint.

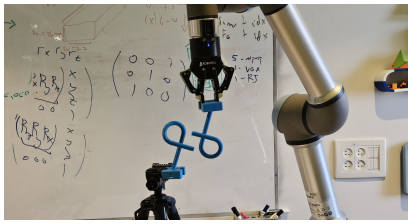

Fig. 5. Preliminary attempts at transferring the simulation plans to UR5.

24 instances, divided to three categories (of eight instances each): `screws`, `puzzles` and `others`. Since our method deals with cases for which there is only a discrete number of contact points at all times, we have not evaluated our method on the screws category, which have contact surfaces (i.e., the set of all contact points in the workspace is a 2-manifold).

We compare our results against the method presented in [19], which we refer to as TIAN-ET-AL, as well as the BK-RRT algorithm [20]. We ran each method on each instance twice, with a timeout of 15000 seconds. If before that timeout, a valid path of motion was found we consider this a success. In Table I we show the success rate of the methods on each subset. We outperform both BK-RRT and TIAN-ET-AL, by being able to solve all problems in the benchmark consistently.

C. Towards Solving the Elk Puzzle

Another state-of-the-art work we have compared our work to is by Zhang et al. [13]. In their work, they identify geometric features which may be found on individual pieces and could hint at the location of the tunnels. They present their results on a series of disentanglement puzzles. Here we show how the fact that our algorithm does not rely on the existence and identification of local features, enables it to perform better in cases where these features do not exist, or those which exist - do not lead to an actual tunnel.

As many of the puzzles in [13] also appear in [19], while others are (interesting) variants of each other, we skip the quantitative comparison and refer to a few representative puzzles. First is the classic Alpha Puzzle 1.0, which purely represents one of the features Zhang et al. are using. Their success rate is 20% and 50% (for two variants of the algorithm), while our algorithm achieves a consistent 100%. Another puzzle, of a different type, which uncovers a second type of local feature Zhang et al. are leveraging is the duet puzzle. Here their two algorithms achieve 60% and 90%, while ours achieves consistent 100% success rate. The third, and probably the most interesting one, is the *Elk* puzzle, which seems highly non-trivial to solve [13]. As discussed in [13, Section 8.1], the features needed by their algorithm on individual pieces are simply not there. What makes it even harder is that along the solution, there are multiple tunnels for which there are more than three contact points, which are not in close proximity to each other. As shown in Figure 4, our algorithm is able to traverse such tunnels. We place the robot in a random configuration, and let the algorithm explore the critical manifold. Indeed, the robot slides and rolls with the contact points, and passes some of the tunnels, although it has not yet obtained the entire solution.

VII. Limitations and Future Work

A. Occlusions in Sampled Configurations

In order to guide the search tree towards tunnels and along them, samples (not necessarily in $\mathcal{C}_{\text{free}}$) in the area of the tunnel are needed. There are situations where branches of the tree are close to the tunnel but unreachable through the tunnel. Consequently, they create an occlusion in the sampling process for growing the tree towards the tunnel, and may increase the search time dramatically. We intend to tackle this limitation in future work, probably by breaking the search tree into a forest.

B. Running Time and Efficiently Computing the SDF

Currently the SDF is implemented using a neural network. As such, computing the signed distances using a forward pass is relatively fast, but using a backward pass for extracting the gradients is time consuming and influences the run time. In future work we intend to replace the neural network with a more efficient means of direct gradient calculation.

C. Multi-Part Assemblies

The algorithm presented solves only two parts problem. We intend to expand it to k-handed problems (see, e.g., [51]) using similar principles.

D. Solving Puzzles in the Real World

Executing tight motion planning solutions in the real world is the ultimate goal. We have not shown that yet. We have put together the experimental setup (see Figure 5), which currently solves wider 3D puzzles. Our immediate next goal is to complete the path conversion of the tight puzzles to the system and physically demonstrate the solutions. We plan on starting with a pre-defined grasping, and continue to automating the grasping decision similar to [52].

Acknowledgement The authors are grateful to Steven M. Lavalle for insightful discussions on the contents of the paper, as well as to James Kuffner for sharing his original code solving the Alpha Puzzle. In addition, we thank Tomer Buber and Tal Geller for their assistance in setting up the environment for the experiments.

REFERENCES

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

[2] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[3] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, "A fast RRT algorithm for motion planning of autonomous road vehicles," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 1033–1038.

[4] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "RRT*-connect: Faster, asymptotically optimal motion planning," in *2015 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2015, pp. 1670–1677.

[5] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 473–483, 2016.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[7] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1478–1483.

[8] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *2003 IEEE international conference on robotics and automation (cat. no. 03CH37422)*, vol. 3. IEEE, 2003, pp. 4420–4426.

[9] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105–1115, 2005.

[10] 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, pp. 81–100, 2018.

[11] H. Ha, J. Xu, and S. Song, "Learning a decentralized multi-arm motion planner," in *Conference on Robotic Learning (CoRL)*, 2020.

[12] S. Ruan, K. L. Poblete, H. Wu, Q. Ma, and G. S. Chirikjian, "Efficient path planning in narrow passages for robots with ellipsoidal components," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 110–127, 2022.

[13] X. Zhang, R. Belfer, P. G. Kry, and E. Vouga, "C-space tunnel discovery for puzzle path planning," *ACM Trans. Graph.*, vol. 39, no. 4, aug 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392468

[14] T. Lefebvre, J. Xiao, H. Bruyninckx, and G. De Gersem, "Active compliant motion: a survey," *Advanced Robotics*, vol. 19, no. 5, pp. 479–499, 2005.

[15] F. Dai, A. Wahrburg, B. Matthias, and H. Ding, "Robot assembly skills based on compliant motion," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*. VDE, 2016, pp. 1–6.

[16] J. Friedman, J. Hershberger, and J. Snoeyink, "Efficiently planning compliant motion in the plane," *SIAM Journal on Computing*, vol. 25, no. 3, pp. 562–599, 1996.

[17] J. De Schutter and H. Van Brussel, "Compliant robot motion I. a formalism for specifying compliant motion tasks," *The International Journal of Robotics Research*, vol. 7, no. 4, pp. 3–17, 1988.

[18] M. A. Peshkin, "Programmed compliance for error corrective assembly," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 4, pp. 473–482, 1990.

[19] Y. Tian, J. Xu, Y. Li, J. Luo, S. Sueda, H. Li, K. D. Willis, and W. Matusik, "Assemble them all: Physics-based planning for generalizable assembly by disassembly," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 6, pp. 1–11, 2022.

[20] S. Zickler and M. M. Veloso, "Efficient physics-based planning: Sampling search via non-deterministic tactics and skills." in *AAMAS (1)*, 2009, pp. 27–33.

[21] C. Suh, T. T. Um, B. Kim, H. Noh, M. Kim, and F. C. Park, "Tangent space RRT: A randomized planning algorithm on constraint manifolds," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4968–4973.

[22] 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.

[23] L. Jaillet and J. M. Porta, "Path planning with loop closure constraints using an atlas-based RRT," in *Robotics Research: The 15th International Symposium ISRR*. Springer, 2017, pp. 345–362.

[24] C. Voss, M. Moll, and L. E. Kavraki, "Atlas+ x: Sampling-based planners on constraint manifolds," Department of Computer Science, Rice University, Houston, TX, Tech. Rep., 2017.

[25] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 625–632.

[26] M. Jones, J. Baerentzen, and M. Sramek, "3D distance fields: a survey of techniques and applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 581–599, 2006.

[27] H. Zhao, "A fast sweeping method for eikonal equations," *Mathematics of computation*, vol. 74, no. 250, pp. 603–627, 2005.

[28] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.

[29] J. Zhang, Y. Yao, and L. Quan, "Learning signed distance field for multi-view surface reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6525–6534.

[30] D. Driess, J.-S. Ha, M. Toussaint, and R. Tedrake, "Learning models as functionals of signed-distance fields for manipulation planning," in *Conference on Robot Learning*, 2022, pp. 245–255.

[31] J. Shim, C. Kang, and K. Joo, "Diffusion-based signed distance fields for 3d shape generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20 887–20 897.

[32] T. Davies, D. Nowrouzezahrai, and A. Jacobson, "On the effectiveness of weight-encoded neural implicit 3d shapes," *arXiv preprint arXiv:2009.09808*, 2020.

[33] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[34] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li, "Nerf: Neural radiance field in 3d vision, a comprehensive review," 2023.

[35] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and Z. Corse, "Local optimization for robust signed distance field collision," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 3, no. 1, pp. 1–17, 2020.

[36] H. Xu and J. Barbič, "6-dof haptic rendering using continuous collision detection between points and signed distance fields," *IEEE transactions on haptics*, vol. 10, no. 2, pp. 151–161, 2016.

[37] D. Koschier, C. Deul, and J. Bender, "Hierarchical hp-adaptive signed distance fields." in *Symposium on Computer Animation*, 2016, pp. 189–198.

[38] J. Bender, C. Duriez, F. Jaillet, and G. Zachmann, "Continuous collision detection between points and signed distance fields," in *Workshop on Virtual Reality Interaction and Physical Simulation*, vol. 8, 2014.

[39] H. Bertiche, M. Madadi, and S. Escalera, "Neural implicit surfaces for efficient and accurate collisions in physically based simulations," *arXiv preprint arXiv:2110.01614*, 2021.

[40] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[41] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008. [Online]. Available: https://doi.org/10.1515/9781400830244

[42] P. Jain, P. Kar *et al.*, "Non-convex optimization for machine learning," *Foundations and Trends® in Machine Learning*, vol. 10, no. 3-4, pp. 142–363, 2017.

[43] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, "Accelerated methods for nonconvex optimization," *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1751–1772, 2018.

[44] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[45] M. Danilova, P. Dvurechensky, A. Gasnikov, E. Gorbunov, S. Guminov, D. Kamzolov, and I. Shibaev, "Recent theoretical advances in non-convex optimization," in *High-Dimensional Optimization and Probability: With a View Towards Data Science*. Springer, 2022, pp. 79–163.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[47] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," 2020.

[48] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS 2017 Workshop on Autodiff*, 2017. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ

[49] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[50] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[51] D. Halperin, J. Latombe, and R. H. Wilson, "A general framework for assembly planning: The motion space approach," *Algorithmica*, vol. 26, no. 3-4, pp. 577–601, 2000. [Online]. Available: https://doi.org/10.1007/s004539910025

[52] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004.