Efficient Path Planning in Narrow Passages via Closed-Form Minkowski Operations

Sipu Ruan, Karen L. Poblete, Hongtao Wu, Qianli Ma and Gregory S. Chirikjian*

Abstract—Path planning has long been one of the major research areas in robotics, with PRM and RRT being two of the most effective classes of path planners. Though generally very efficient, these sampling-based planners can become computationally expensive in the important case of “narrow passages”. This paper develops a path planning paradigm specifically formulated for narrow passage problems. The core is based on planning for rigid-body robots encapsulated by unions of ellipsoids. The environmental features are enclosed geometrically using convex differentiable surfaces (e.g., superquadrics). The main benefit of doing this is that configuration-space obstacles can be parameterized explicitly in closed form, thereby allowing prior knowledge to be used to avoid sampling infeasible configurations. Then, by characterizing a tight volume bound for multiple ellipsoids, robot transitions involving rotations are guaranteed to be collision-free without traditional collision detection. Furthermore, combining the stochastic sampling strategy, the proposed planning framework can be extended to solving higher dimensional problems in which the robot has a moving base and articulated appendages. Benchmark results show that, remarkably, the proposed framework outperforms the popular sampling-based planners in terms of computational time and success rate in finding a path through narrow corridors and in higher dimensional configuration spaces.

Index Terms—Motion and path planning, computational geometry, Minkowski sums

I. INTRODUCTION

Sampling-based planners such as PRM [2] and RRT [3] (and a multitude of their extensions, e.g., [4], [5]) have demonstrated remarkable success in solving complex robot motion planning problems. These frameworks generate state samples randomly and perform explicit collision detection to assess their feasibility. These methods have had a profound impact both within robotics and across other fields such as molecular docking, urban planning, and assembly automation.

It is well known that despite the great success of these methods, the “narrow passage” problem remains a significant challenge. Generally speaking, when there is a narrow passage, an inordinate amount of computational time is spent on the random state samples and edges that eventually will be discarded. To increase the probability of sampling and connecting valid configurations in a narrow passage, various methods have been proposed such as [6]–[8] (Sec. II-A provides more detailed reviews on narrow passage problems). In this article, however, the narrow passage problem is addressed through an explicit closed-form characterization of the boundary between free and in-collision regions. The first goal of this paper is to:

1. Extend the previous methods of parameterizing the free space for single-body ellipsoidal robots avoiding ellipsoidal obstacles [9]. A more general case is studied where the obstacles are enclosed by arbitrary convex differentiable surfaces. By doing so, traditional collision checking computations can be avoided when generating collision-free configurations.

In our proposed path planning framework, the robot is encapsulated by a union of ellipsoids, while the environmental features are enclosed by superquadrics. Both rigid and articulated robots are considered in this article. The configuration spaces to be studied are $SE(d)$ and $SE(d) \times (S^1)^n$ for rigid-body and articulated robots respectively. Ellipsoids have a wide range of applications in encapsulating robots. For example, the projection contour of a humanoid robot can be tightly encapsulated by an ellipse since its shoulders are narrower than the head [10] (Fig. 1a). In computational crystallography, it is natural to enclose a protein molecule by a moment-of-inertia ellipsoid, which simplifies the complex geometries and maintains the physical information of the protein [11] (Fig. 1b). Moreover, superquadrics generalize ellipsoids by adding freedoms in choosing the power of the exponents rather than restricting to quadratics. The family of superquadrics represents a wider range of the complex shapes (e.g., cuboids, cylinders, etc) while still requiring only a few parameters [12].

When a robot is fixed at a certain orientation and internal joint angles, a “slice” of the configuration space (C-space) is defined by the Minkowski sums between the rigid body parts and the obstacles in the workspace [13], [14], denoted here as a “C-layer” [15]. (Sec. II-B reviews the literature in details on the computations of Minkowski sums). Once the configuration space obstacles (C-obstacles) are computed, the complement regions between the planning arena and the union of C-obstacles is the free space that allows the robot to travel through. Consequently, collision-free samples can be generated within this collision-free C-space. However, if one seeks to connect such samples using current sampling-based planning paradigms like PRM or RRT, explicit collision

1$SE(d), d = 2,3$ is the pose of the robot base frame and $(S^1)^n$ represents the configuration space of $n$ revolute joints.

2Here, the word “arena” denotes the bounded area in which the robot and obstacles are contained.
provides mathematical parameterizations of free space in each C-layer from HRM.

internal joint angles) and takes advantage of the explicit ples the rotational components (\(i\))

algorithm called “Highway RoadMap (HRM)” is proposed here to make planning in higher

carding configurations between different C-layers without

transition.

encloses all possible intermediate configurations within the transition.

All the above methods are combined into a path planning algorithm called “Highway RoadMap (HRM)”. This planner is deterministic and suitable for rigid-body planning problems. It is known that traditional deterministic path planners suffer from the curse of dimensionality burdens in the case of articulated robots. They might result in exponential combinatorial complexity with respect to the number of internal degrees of freedom. Therefore, the third goal of this article is to:

3. Develop an effective method to solve the exponential combinatorial complexity for the planning of articulated robots.

A hybrid algorithm called “Probabilistic Highway RoadMap (Prob-HRM)” is proposed here to make planning in higher dimensional configuration spaces tractable. It randomly samples the rotational components (i.e., the base orientation and internal joint angles) and takes advantage of the explicit parameterizations of free space in each C-layer from HRM.

This article extends the conference version [1] of the same topic, and has significant updates. Comparing to our conference paper, the key contributions of this article are:

- Extend the graph construction procedure in each C-layer to 3D multi-body case;
- Introduce a novel “bridge C-layer” method to connect vertices between adjacent C-layers;
- Propose a hybrid planner which integrates the advantages of sampling-based planners on higher dimensional articulated robot planning problems;
- Conduct rigorous benchmark simulations and physical experiments in challenging environments to evaluate the proposed planning framework.

These extensions are essential since more general 3D and articulated robot models are implemented. The benchmark and physical experimental settings are also more realistic. Thus, these improvements show that our proposed framework can be extendable to more complex and realistic higher dimensional planning problems.

The rest of this article is organized as follows. Section II reviews related literature. Section III provides mathematical foundations for the proposed path planning framework. Section IV extends our previously proposed Highway RoadMap (HRM) algorithm to the case of 3D multi-body robot with ellipsoidal components. The novel method of “bridge C-layer” is then introduced to connect different C-layers. Section V introduces the hybrid Prob-HRM planner which integrates the spirits of random sampling for robot shapes. In Section VI, benchmark studies are conducted with some popular and successful sampling-based planners from the well-known Open Motion Planning Library (OMPL). In Section VII, our planning framework is demonstrated by physical experiments in real world, which solve walking path planning problems for a humanoid robot in cluttered environments. We discuss the advantages and limitations of our framework in Section VIII. Finally, we conclude in Section IX.

II. LITERATURE REVIEW

This section reviews related work on narrow passages, Minkowski sum computations, and parametric surface representations.

A. The challenge of narrow passages

One of the key factors that affects the performance of sampling-based planners is the state sampling strategy. A simple one is to sample uniformly at random in the whole C-space. But if there are narrow corridors in the planning arena, it might take too much time to find a valid collision-free sample.

To tackle this so-called “narrow passage” challenge, various of biased samplers have been studied throughout these years, most of which try to capture the local features around obstacles. For example, UOBPRM [16] iteratively searches for collision-free samples from a configuration in collision by moving in different ray directions. The bridge test [6] tries to find a collision-free middle point between configurations that are in collision with the obstacles. In [8], a Bayesian learning scheme is used to model sampling distributions, which subsequently updates based on previous samples and maximizes the likelihood of sampling from high probability regions.

Other methods combine the advantages of different kinds of algorithms, which also show effectiveness in tackling narrow passage problems. For example, Toggle PRM [17] simultaneously maps both free space and obstacle space, enabling an augmentation from a failed connection attempt in one space to the other. Spark PRM [7] grows a tree inside the narrow passage region to connect different parts of the roadmap on...
different ends of the region. Retraction-based RRT [18] tries to retract initial samples into more difficult regions, so as to increase probability of sampling near narrow passages. More recently, a reinforcement learning method is applied to enhance the ability to explore local regions where the tree grows [19].

Hybrid planner [15] combines random sampling strategy with Minkowski sums computations, which increases the probability of identifying difficult regions. In this article, we use the same fashion to randomly sample the robot shapes. Nevertheless, the differences are significant. We propose a closed-form Minkowski sum expression for continuous bodies, as compared to point-based Minkowski sums for polyhedral objects. To generate valid vertices, they directly choose the points on C-obstacle boundary, but we compute vertices in the middle of free space in a more uniform way. And to connect among different C-layers, they add a new vertex and search for paths on the C-obstacle boundaries, but we instead generate a new layer based on an enlarged void.

Some of the successful sampling-based planners and state samples are implemented in the well-known Open Motion Planning Library (OMPL) [20], which provides us a standardized way to benchmark new algorithms. In this article, we evaluate the proposed planning framework with some famous sampling-based planners from this library.

B. Computations of Minkowski sums

The Minkowski sum is ubiquitous in many fields such as computational geometry [21], robot motion planning [13], control theory [22], etc. Despite its straightforward definition, which will be given in Sec. III, computing an exact boundary of Minkowski sum between two polytopes can be very expensive and has attracted extensive attentions for efficient implementations for decades.

The computational complexity of Minkowski sums between two general non-convex polytopes in \( \mathbb{R}^3 \) can be as high as \( O(N_1^3 N_2) \), where \( N_1 \) and \( N_2 \) are the complexities (i.e., the number of facets) of the two polytopes. Therefore, many efficient methods decompose the general polytopes into convex components [23], since the Minkowski sums between two convex polytopes can achieve \( O(N_1 N_2) \) complexity [24]. Another type of methods is based on convolutions of two bodies, with the fact that Minkowski sum of two solid bodies is the support of the convolution of their indicator functions [25]. A simple approximated algorithm [26] is proposed that avoids computing 3D arrangement and winding numbers, and reduces the trimming issue that many convolution-based methods might face via collision detection. [27] proposes an exact Minkowski sums for polytopes that contain holes using convolution. In addition, point-based methods avoid convex decomposition which is an expensive step of the decomposition-based methods. These algorithms improve the performance through approximations and are easy to implement [28]. The major advantages are the ease of generating points than meshes, and the possibility of parallelisms [29]. But the local properties rather cannot be expressed by individual points themselves. An exact closed-form Minkowski sums formula for \( d \)-dimensional ellipsoids was introduced [30]. And in [31], a parameterized ellipsoidal outer bound for the Minkowski sum of two ellipsoids is proposed. This article discovers that the Minkowski sums of an ellipsoid and an arbitrary convex differentiable shape embedded in \( d \)-dimensional Euclidean space can also be computed in closed form.

C. Ellipsoids and superquadrics for object representation

Apart from using polytopes or surface meshes for object representations, other geometric primitives such as ellipsoids and superquadrics play an important role due to their continuous and convex features and simple algebraic characterizations. A 3D ellipsoid in a general pose only needs 3 semi-axes lengths and 6 pose variables for a full definition. For superquadrics, two additional variables are needed, i.e., the exponential factors. Even with such a small set of parameters, a large range of shapes can be represented. Therefore, in many recent applications, ellipsoids or superquadrics are good candidates to encapsulate objects [32], [33].

Algorithms related to ellipsoids are studied extensively for efficient calculations [34], many of which are implemented and integrated in toolboxes [35]. Minimum volume enclosing ellipsoid (MVEE), which is characterized as a convex optimization problem [36], is widely applied. The studies of algebraic separation conditions for two ellipsoids provides very efficient algorithms to detect collisions in both static and dynamic cases [37], [38]. In addition, efficiently computations of distance between two ellipsoids is also attractive [39], [40]. In contrast to separation, once an ellipsoid is fully contained in another, the volume of its limited available motions is studied from both algebraic and geometric points of view [41].

Superquadrics can be seen as a high-order extension of ellipsoids, with the two additional exponents determining the sharpness and convexity [12]. They are able to represent a wide range of geometries such as cube, cylinder, octahedron, and ellipsoid. Using optimization or deep learning techniques, point cloud data can be segmented and fitted by unions of superquadrics [42], [43]. Proximity queries and contact detection are useful applications of this geometric model [44], [45]. In this article, body parts of the robots are enclosed by ellipsoids and obstacles are represented by superquadrics.

III. Mathematical Preliminaries

This section provides the mathematical preliminaries that are required for developing the new path planning paradigm in this article.

A. Minkowski sum and difference between two bodies

The Minkowski sum and difference of two point sets (or bodies) centered at the origin, i.e., \( P_1 \) and \( P_2 \) in \( \mathbb{R}^d \), are defined respectively as [46]

\[
P_1 \oplus P_2 \doteq \{p_1 + p_2 \mid p_1 \in P_1, p_2 \in P_2\}, \quad \text{and} \quad P_1 \ominus P_2 \doteq \{p \mid p + P_2 \subseteq P_1\}. \tag{1}
\]

When computing the set when the two bodies touch each other (i.e., their contact space), we refer to the calculation of \( P_1 \oplus \)}
for the 3D case. Note that the idea here is inspired by matrix with the semi-axis length $a$.

SO(d) denotes the orientation of $E$. The expression can be further extended when one ellipsoid is substituted by an arbitrary convex differentiable surface embedded in $\mathbb{R}^d$.

**B. Implicit and parametric surfaces**

Assume that $S_1$ is a convex and differentiable hyper-surface embedded in $\mathbb{R}^d$. Its implicit and parametric forms can be expressed as

$$
\Phi(x_1) = 1 \quad \text{and} \quad x_1 = f(\psi_1),
$$

where $\Phi(\cdot)$ is a real-valued differentiable function of $x_1 \in \mathbb{R}^d$ and $f$ is a differentiable $d$-dimensional vector-valued function of $\psi_1 = [\psi_1, \psi_2, ..., \psi_{d-1}]^T \in \mathbb{R}^{d-1}$. Let $E_2$ be an ellipsoid in $\mathbb{R}^d$ in general orientation, with semi-axis lengths $a_2 = [a_1, a_2, ..., a_n]^T$. Then, its implicit and explicit equations are of the form

$$
x_2^T A_2^{-2} x_2 = 1 \quad \text{and} \quad x_2 = A_2 u(\psi_2),
$$

where $A_2 = R_2 \Lambda(a_2) R_2^\top$ is the shape matrix of $E_2$, $R_2 \in \text{SO}(d)$ denotes the orientation of $E_2$, and $\Lambda(\cdot)$ is a diagonal matrix with the semi-axis length $a_i$ at the $(i, i)$ entry. Here, $u(\psi_2)$ is the standard parameterization of the $d$-dimensional unit hyper-sphere using $d-1$ angles.

**C. Closed-form Minkowski operations between an ellipsoid and a general convex differentiable surface**

It has been shown previously in [30] that the Minkowski sum and difference between two ellipsoids can be parameterized in closed-form. The expression can be further extended when one ellipsoid is substituted by an arbitrary convex differentiable surface embedded in $\mathbb{R}^d$ [1]. The general simplified form can be computed as

$$
x_{mb} = x_1 + R_2 \Lambda^2(a_2) R_2^\top \nabla_{x_1} \Phi(x_1) \frac{1}{\| A_2(a_2) R_2^\top \nabla_{x_1} \Phi(x_1) \|} ,
$$

where $\nabla_{x_1} \Phi(x_1)$ is the gradient of $S_1$ at $x_1$. Figure 2 illustrates the geometric interpretation of the computational process. The detailed derivations were presented in [1], and can be referred to in the supplementary material.

**D. The minimum volume concentric ellipsoid (MVCE) enclosing two ellipsoids with the same center**

When two ellipsoids are fixed at the same center, a “minimum volume concentric ellipsoid (MVCE)” can be computed in closed form. The computational procedure is visualized in Fig. 3 for the 3D case. Note that the idea here is inspired by [34], which provides equivalent computations for a maximum volume concentric ellipsoid covered by two ellipsoids.

Given two $d$-dimensional ellipsoids $E_a$ and $E_b$ with semi-axis lengths $a$ and $b$ respectively, their shape matrices can be expressed as $A = R_a \Lambda^{-2}(\alpha) R_a^\top$ and $B = R_b \Lambda^{-2}(\beta) R_b^\top$. One ellipsoid (e.g., $E_a$) can be shrunk into a sphere $(E_0)$ via the affine transformation $T = R_a \Lambda(r/b) R_b^\top$, where $r$ is the radius and $r/b = [r/b_1, r/b_2, ..., r/b_d]^T \in \mathbb{R}^d$. Then shape matrix for $E_a$ in shrunk space, i.e., $E_a'$, can be computed as $A' = T^{-1} R_a \Lambda^{-2}(\alpha) R_a^\top T^{-1}$. Then, using singular value decomposition (SVD), its semi-axis lengths and orientation, i.e., $\alpha'$ and $R_a'$, can be obtained respectively. Then, the shape matrix of their MVCE, $E_m$, is

$$
M = TR_a \Lambda^{-2}(\alpha', r) R_a^\top T,
$$

where $\max(\alpha', r) \geq \max(\alpha_1', r), ..., \max(\alpha_d', r)^T$ and $\alpha' \in [\alpha_1', \alpha_2', ..., \alpha_d']^T \in \mathbb{R}^d$.

Furthermore, this equation can be applied iteratively if there are multiple concentric ellipsoids. For example, the MVCE that enclose the previous two ellipsoids, along with the next ellipsoid, can be enclosed by a new MVCE. The final resulting ellipsoid encapsulates all the original set of ellipsoids, which is denoted as a tightly-fitted ellipsoid (TPE) and will be used extensively in the process of edge connections.

**IV. HIGHWAY ROADMAP PATH PLANNING ALGORITHM FOR RIGID-BODY ROBOTS WITH ELLIPSOIDAL COMPONENTS**

This section introduces the extended “Highway RoadMap (HRM)” algorithm. The previous work [9] only implemented for the 2D planning problems. Therefore, the detailed algorithmic processes about how to extend to 3D rigid-body path planning problems are explained. Then, a novel vertex connection strategy for configurations with different rotational components is proposed. This strategy can be applied when the robot is constructed by multiple ellipsoids, which extends the capabilities of the HRM planner for more complex robot models. In the following, we give a general overview of the whole algorithm and show details of each subroutine subsequently.

**A. Overview of the Highway Roadmap planner**

Firstly, the orientations of the robot are sampled and stored a priori. At each fixed orientation, a subset of the C-space that only contains translational motions is built, denoted here as a “C-layer”. Then, to detect the collision-free regions at each C-layer, a “sweep line” process is applied, which results in a set of collision-free line segments. And within each free segment, vertices are generated and connected among adjacent sweep lines within the same C-layer. Finally, vertices among adjacent C-layers are connected using a novel idea of “bridge C-layer”.

The general workflow to construct this graph-based roadmap system is illustrated in Algorithm 1. To visually demonstrate the concept, a fully connected graph obtained by running our algorithm in the planar case is shown in Figure 4. In particular, the input of the robot is a union of ellipsoids which contains the shape and kinematic data. The shape information of an ellipsoid includes semi-axes lengths, center point coordinates.
(a) Both bodies are rotated by $R_2$. (b) $E_2$ is shrunk as a sphere. (c) An offset surface is computed in shrunk space. (d) Stretch back and obtain $S_1 \oplus (-E_2)$.

Fig. 2: Process for the characterizations of the Minkowski sums between a superquadric surface $S_1$ and an ellipsoid $E_2$.

(a) Two concentric 3D ellipsoids, $E_a$ and $E_b$. (b) Shrink $E_b$ into a sphere, and enclose a larger ellipsoid $E'_m$. (c) Transform the whole space back to get MVCE $E_m$.

Fig. 3: Computational procedure for minimum volume concentric ellipsoid that covers two ellipsoids in 3D.

**Algorithm 1:** Highway RoadMap (HRM) Algorithm

| Inputs                        | Output                      |
|-------------------------------|-----------------------------|
| robot: a union of ellipsoidal objects; | roadmap: a graph structure; |
| obstacle: a set of superquadric objects; | path: an ordered list of configurations |
| arena: a set of superquadric objects; | orientation ← SampleOrientations($N_{Layer}$); |
| endpts: start and goal configurations | i = 1; |

for $i \leq N_{Layer}$ do

robot.Orientation = orientation[$i$] ;

$C_{Obstacle}, C_{Arena} \leftarrow$ MinkowskiOperations(robot, obstacle, arena);

$C_{Free,i} \leftarrow$ SweepLineProcess($C_{Obstacle}, C_{Arena}, N_{Line}$);

roadmap.vertex.Append(GenerateNewVertex($C_{Free,i}$));

roadmap.edge.Append(ConnectWithinOneLayer($C_{Free,i}$));

$C_{Free}.Append( C_{Free,i} );$

i += ;

end

roadmap.edge.Append(ConnectMultiAdjacentLayers($C_{Free}$));

path ← GraphSearch(roadmap, endpts)

Fig. 4: The fully connected graph structure, generated from one simulation trial. The vertical axis represents the rotational angle; dots are vertices and line segments are edges.

and orientation. The kinematic data of each body part stores the relative rigid-body transformation with respect to the base. The input environmental data includes a set of superquadric objects that enclose the obstacles and arena. And the endpts input indicates the start and goal configurations of the robot. There are two major user-defined parameters: the number of C-layers $N_{Layer}$ and the number of sweep lines at each C-layer $N_{Line}$. These two parameters determine the resolution of the resulting roadmap. They are fixed during the execution of the algorithm in order to make it deterministic and terminate in a finite time interval. The outputs of the algorithm are the roadmap and path. The roadmap is represented as a graph structure and can be reused to answer different path queries with the same environmental settings. And the path is represented as an ordered list of configurations from the start to the goal.
For the algorithm, Line 1 generates $N_{Layer}$ of discrete rotations in the special orthogonal group (SO(d)). Within each C-layer, the closed-form Minkowski sum and difference for the bodies of the robot are computed with the obstacles and arena, respectively (Line 5). Once the Minkowski operations are applied, $C_{Obstacle}$ and $C_{Arena}$ are generated. Then by sweeping lines throughout the C-layer with a certain resolution (indicated by $N_{Line}$), in Line 6, the free portion of the C-layer (C-free) is detected and represented as a set of line segments. Furthermore, the middle point of each collision-free line segment is appended to the vertex list (Line 7) in the roadmap. Then, edges are attempted to connect between two vertices in adjacent sweep lines (Line 8). The entire roadmap system can then be constructed by connecting vertices among adjacent C-layers in Line 12. Finally, in Line 13, a graph search technique is applied to find a path from the starting configuration to the goal. In this work, A* algorithm [13] is used. The following subsections discuss each major subroutine in details.

B. Discretization of the robot orientations

The orientation in Line 1 of Algorithm 1 is a pre-computed set of samples from the special orthogonal group. Specifically, in 2D case, uniformly distributed angles within the interval $[-\pi, \pi]$ are computed. In 3D, the icosahedral rotational symmetry group of the Platonic solid (consisting 60 elements) is used, which gives a finite and deterministic sampling of SO(3). The geodesic distances between two neighboring samples are almost uniformly distributed [47]. Using this set of orientation samples, the rotational difference between two adjacent C-layers is smaller compared to non-uniform sample sets. Note that more rotations can be sampled to construct a denser roadmap per the user’s choice. More details on sampling rotations on SO(3) can be found in [48], [49].

C. Minkowski operations for a multi-body robot at each C-layer

At each C-layer, the closed-form Minkowski operations are computed to generate C-obstacles (Line 5 of Algorithm 1). The robot is constructed by a finite union of $M$ rigidly connected ellipsoids $E_1, E_2, \ldots, E_M$. Without the loss of generality, $E_1$ is chosen as the base of the robot. The relative transformations between the base $E_1$ and other ellipsoidal parts $E_2, E_3, \ldots, E_M$ are defined as $g_i = (R_i, t_i)$ ($i = 2, \ldots, M$), respectively. For a multi-link rigid-body robot, these relative transformations can be computed via forward kinematics with all the internal joints being fixed. With this definition and the property from Eq. (2), the union of the Minkowski operations for all body parts can be expressed relative to one single reference point, which we choose as the center of the base ellipsoid $E_1$. In particular, for each ellipsoidal body $E_i$, a positional offset $t_i$ is added to Eq. (14). For practical computational purposes, each Minkowski sum and difference boundary is discretized as a convex polygon in 2D and polyhedral mesh in 3D. The vertices of the discrete boundary are generated using the parametric expression of Minkowski operations. The

free space is the region enclosed by C-arena and outside C-obstacle boundary. Figure 5 shows the Minkowski sums of a multi-body robot at a fixed orientation (Fig. 5a) and the collision-free C-space in the corresponding C-layer (Fig. 5b).

D. A sweep-line process to characterize free regions within one C-layer

This subsection introduces a sweep-line process to detect collision-free regions at each C-layer in both 2D and 3D cases. Three main parts are discussed: definitions of sweep lines, generations of collision-free vertices and vertex connections between adjacent sweep lines.

1) Definition of the sweep line: The general idea of this “sweep-line” process is analogous to raster scanning – a set of parallel lines is defined to sweep throughout the whole C-layer and intersect with C-arena and C-obstacles. Theoretically, these parallel lines can be defined along any direction in the whole space. But for simplicity of representation and storage, the lines are defined to be parallel to the basis axes of the coordinate system. Specifically, in the 2D case, the sweep lines are parallel to x-axis, which is therefore defined by a set of discrete y-coordinates under certain resolution within the upper and lower bounds. In 3D case, the sweep lines are parallel to z-axis, which is defined by a set of discrete pairs of (x, y)-coordinates. Note that the pre-defined number of sweep lines directly determines the size of the roadmap to be constructed. Meanwhile, the sweep lines in different C-layers are generated with the same set of coordinates. By doing
so, the adjacency of sweep lines in different C-layers can be searched easily via their indices.

2) Computations of collision-free configurations within each sweep line: To generate collision-free configurations, the intersecting points between each sweep line with all C-obstacles and C-arenas are computed. Since the C-obstacle and C-arena boundaries are discretized in the previous Minkowski operations, the intersections can be computed between line and polygonal boundaries (in 2D case) or surface meshes (in 3D case). The resulting intersections are then saved as line segment intervals. Denote the line segments within C-obstacles as $L_{Ox}$ and those within the C-arenas as $L_{A_e}$, then the collision-free line segment $L_{F\text{ree}}$ for each sweep line can be represented as \[ L_{F\text{ree}} = \bigcap_{k=2}^{M_A \times M} L_{A_k} - \bigcup_{k=1}^{M_O \times M} L_{O_k}, \] (7)

where $M_A$ and $M_O$ are numbers of arenas and obstacles respectively. Then, all these collision-free line segments are stored in $C_{F\text{ree},i}$ for the $i^{th}$ C-layer (Line 6 of Algorithm 1). Next, in Line 7 of Algorithm 1, collision-free vertices are generated from $C_{F\text{ree},i}$ as the middle point of each $L_{F\text{ree}}$.

3) Vertex connections among adjacent sweep lines: Once a list of collision-free configurations (“vertex” in graph) is generated, the next step is to connect them efficiently (Line 8 of Algorithm 1). In this work, only two vertices in adjacent sweep lines are tried to be connected with a straight line segment (“edge” in graph). In 2D, the adjacency is defined along y-direction; while in 3D, only the neighbors in x- and y-directions are considered as adjacent sweep lines. Each candidate edge should be inside the collision free region, i.e., within C-arena and outside C-obstacles. There might be multiple free segments for each sweep line, so a traversal through all possible segments is required. Assume the connection is attempted between $V_{j,k1}$ in $L_{F\text{ree},i,j}$ of $j^{th}$ line and $V_{j+1,k2}$ in $L_{F\text{ree},i+1,k}$ of $(j+1)^{th}$ line. The connection validity is checked by computing the intersections between the line segment $V_{j,k1}V_{j+1,k2}$ and all meshed C-obstacles. The computation of intersection between a line segment and meshed objects is similar to the one in the sweep-line process (i.e., in Line 6). If the segment is outside all C-obstacles, the whole edge is guaranteed to be collision-free. Figure 6 shows the decomposed C-space in one layer of a planar case. The horizontal raster lines indicate the collision-free line segments.

E. A local planner for vertex connections between adjacent C-layers

Since each C-layer only represents one orientation of the robot, rotational motions are required when connecting different C-layers. One solution was proposed to construct a “local C-space” by enlarging the ellipsoidal robot, and connecting two poses inside a convex polyhedral C-space [1]. This method works well when the two configurations are close to each other (i.e., the “small motion” assumption), especially in 2D. However, as shown in [41], when it comes to the 3D case, the local C-space occupies a much smaller portion of the total volume of the motion space, making it difficult to enclose two poses if the orientation difference is large. In this article, a “bridge C-layer” method is proposed (i.e., in Line 12 of Algorithm 1) to guarantee that the vertices at different C-layers can be safely connected without the small motion assumption.

1) Sweep volume for individual ellipsoidal part: Suppose that the $i^{th}$ ellipsoidal part $E_i$ is moving from vertex $V_1 = [t_1^T, \omega_1^T]$ to $V_2 = [t_2^T, \omega_2^T]$ while trying to stay in the free space. The idea here is to enclose $E_i$ at the two configurations by a tightly fitted sweep volume and ensure that $E_i$ at all the intermediate configurations stays fully inside the volume. If the sweep volume is within the collision-free space, all the intermediate configurations within the sweep volume will be guaranteed safe. The intermediate configurations between $V_1$ and $V_2$ can be computed using interpolation technique, which can be subjected to any motion primitive. In this work, linear interpolation between two vertices is considered.

Given a motion primitive connecting $V_1$ and $V_2$, the rotation of the intermediate steps can be retrieved. Firstly, $E_i$ at these orientations are encapsulated by a tightly-fitted concentric ellipsoid (TFE) via the iterative computations of MVCEs (Sec. III-D). Then, the computed TFE translates from $t_1$ to $t_2$ following the interpolated path of $E_i$’s center. The resulting sweep volume bounds the whole transition of $E_i$ between the two configurations. Figure 7a shows the procedure of constructing the sweep volume for an individual body part.

To ensure that the computed TFE stays inside the collision-free space, Minkowski operations are conducted for further validity check. But unlike the operations within one C-layer, no offset needs to be added to the C-obstacle and C-arena boundaries (the reason of doing so will be made clear in Sec. IV-E2). Then, if all the positions of the TFE center from $t_1$ to $t_2$ are within the collision-free regions, the sweep volume is guaranteed to be safe. Therefore, the whole transition for the ellipsoidal part $E_i$ is collision-free.

2) Concatenation of the sweep volumes for a union of ellipsoidal bodies: Once the sweep volumes of individual parts are calculated, the union of them encloses the whole robot. If $g_{i1}$ represents the transformation of base ellipsoid $E_1$ as seen in the world frame, the transformation for each individual part is then $g_{i} \circ g_{i}(i = 2, \ldots, M)$. Then, its translation component guides the sliding motion of the corresponding TFE, which results in the sweep volume enclosing the motion of $E_i$. Figure 7b illustrates the sweep volume union that encloses the whole multi-body robot in the planar case. The robot base...
follows a 2D straight line with rotations, and the TFEs of different body parts follow different paths (as show in white curves). In this process, TFE for each body part translates with respect to its own center individually. This is because that, as Fig. 7b shows, the motion of each robot part is no longer a pure translation. The reference points of Minkowski sums and difference for different body parts have different trajectories to follow. Therefore, the transition for the whole robot is guaranteed safe if all the individual reference points are within their own free space.

3) Vertex connections based on bridge C-layer calculations: With the C-obstacle and C-arena being computed for the TFE of each individual robot part, the next step is to check the validity of the translation motions of each TFE. To achieve this, a “bridge C-layer” is constructed, which is the union of all C-obstacles and C-arenas constructed from TFEs of all the individual robot body parts. When connecting two C-layers, this bridge C-layer only needs to be computed once before checking for validity of the interpolated configurations. This is because the interpolation of rotation parts between the same pair of orientations is the same. For each candidate connection, the robot is transformed according to the interpolated configurations between two vertices. Then, for each configuration, the TFE of each robot part is shrunk into its center point. The inclusion of this point with all the discrete C-obstacles are queried. If any of the center point is inside any C-obstacle, the validating process is terminated and the corresponding connection is discarded. Otherwise, further checks for other ellipsoidal parts are conducted until all the parts are checked. Note that all the existing vertices are tried to connect to their nearest neighbors within adjacent C-layers\(^5\). The nearest neighbors of a vertex are defined as having the same \(x\)-coordinate (in 2D) or \((x, y)\)-coordinate (in 3D), \(i.e.,\) located in the same sweep line. The sweep volume gives a conservative encapsulation of the robot transitions between two vertices. But if the orientation samplings are incremental and uniform, there will not be a large rotational difference between adjacent C-layers. Thus, the extra free space inside the sweep volume will be small.

V. Hybrid Probabilistic Variation of Highway Roadmap Planner for Articulated Robots with Ellipsoidal Components

The original HRM planner in Sec. IV only designs for the case when robot parts are rigidly connected to each other. This limits its ability to extend to higher dimensional configuration space, \(i.e.,\) \(SE(d) \times (S^1)^n\). To avoid the exponential computational complexity in concatenating all possible combinations of the base pose and joint angles, a hybrid algorithm is proposed here. The general idea is to combine with sampling-based planners, which are proved to be advantageous in dealing with the “curse of dimensionality” burden. Algorithm 2 shows the general workflow of the proposed hybrid probabilistic Highway Roadmap (Prob-HRM) planner.

Prob-HRM mainly differs from the original HRM algorithm in that it utilizes the idea of random sampling for rotational components of the robot, while HRM is a deterministic algorithm with fixed resolution during the run time. The robot with fixed rotational components is called a “shape” [15], and one C-layer is computed for each robot shape. Since for each shape, the internal joint angles are fixed, computations within the same C-layer in Prob-HRM stay the same with those in HRM, \(i.e.,\) Lines 5, 6, 7 and 8 in Alg. 2 are the same with the corresponding subroutines in Alg. 1. Other subroutines are also easy to be ported from the original HRM to Prob-HRM. In particular, the only difference for vertex connections among adjacent C-layers (Line 11) with that in HRM is that the connection attempts are made only for the new C-layer in the current loop. In HRM, as a comparison, the adjacent C-layers are connected in the end after all C-layers are generated. Also, the graph search process is conducted each time after the new C-layer is connected to the graph (as in Line 13). In contrast, for HRM, the graph search is conducted once after the whole graph is built. The new subroutines in Prob-HRM are the random sampling of robot shapes (Line 3) and the computations of forward kinematics (Line 4) in each loop. To sample a shape, the orientation of the robot base is uniformly and randomly sampled [48], followed by random sampling of joint angles within their ranges. After that, the forward kinematics is computed to get the poses of all the robot body parts with respect to the world frame. The algorithm terminates either when a path is found from start to the goal configuration, or the time limit is reached. Note that \(N_{Layer}\) is no longer a pre-defined parameter of Prob-HRM since the orientation of the robot base and joint angles are updated online.

\(^5\)The “adjacent C-layers” refer to the two C-layers with the nearest rotational distance.
Algorithm 2: Probabilistic Highway RoadMap (Prob-HRM) Algorithm

Inputs: robot: a list of ellipsoidal objects and robot kinematic information; obstacle: a set of superquadric objects; arena: a set of superquadric objects; endpoints: start and goal configurations;

Parameter: $N_{\text{Line}}$: number of sweep lines

Outputs: roadmap: a graph structure; path: an ordered list of configurations

1. $i = 1$
2. while !TerminationCondition do
3.   $configuration \leftarrow \text{RandomSampleRobotShape}()$
4.   $shape \leftarrow \text{robot.FowardKinematics}(configuration)$
5.   $C_{\text{Obstacle}}, C_{\text{Arena}} \leftarrow \text{MinkowskiOperations}(shape, obstacle, arena)$
6.   $C_{\text{Free},i} \leftarrow \text{SweepLineProcess}(C_{\text{Obstacle}}, C_{\text{Arena}}, N_{\text{Line}})$
7.   roadmap.vertex.Append($\text{GenerateNewVertex}(C_{\text{Free},i})$)
8.   roadmap.edge.Append($\text{ConnectWithinOneLayer}(C_{\text{Free},i})$)
9.   $C_{\text{Free}}.Append(C_{\text{Free},i})$
10. if $i \geq 2$ then
11.     roadmap.edge.Append($\text{ConnectAdjacentLayers}(C_{\text{Free}})$)
12. end
13. $path \leftarrow \text{GraphSearch}(\text{roadmap}, \text{endpts})$
14. $i++;$
15. end

TABLE I: Parameter symbols for the proposed HRM and Prob-HRM planners.

| Parameter symbol | Explanations |
|------------------|--------------|
| $N_{\text{Layer}}$ | Number of C-layers for HRM |
| $N_{\text{Line}}$ | Number of sweep lines for HRM and Prob-HRM |

VI. BENCHMARKS ON PATH PLANNING FOR ELLIPTIOAL ROBOTS IN SUPERQUADRIC ENVIRONMENTS

This section compares the performance of the proposed HRM and Prob-HRM planners with some famous sampling-based motion planners. The configuration spaces studied include $\text{SE}(3)$ and $\text{SE}(3) \times (S^3)^n$. The robots are enclosed by a union of ellipsoids and the obstacles are modeled as superquadrics. Both HRM and Prob-HRM planners are written in C++. The baseline sampling-based planners to be compared are sourced from the well-known “Open Motion Planning Library (OMPL)” [20]. All the benchmarks are conducted on Ubuntu 16.04 using an Intel Core i7 CPU at 3.40 GHz × 8.

TABLE II: Symbols of data structure details for planners.

| Result symbol | Explanations |
|---------------|--------------|
| $N_{\text{Vert}}$ | Number of valid vertices |
| $N_{\text{Edge}}$ | Number of valid edges |
| $N_{\text{Path}}$ | Number of vertices on the solved path |

A. Planning environment and robot type settings

The maps and robots used for benchmarks are summarized in this subsection. Figure 8 shows the planning environments and the solved paths for different robots using our proposed HRM or Prob-HRM planners. Both rigid-body and articulated robots are considered. The rigid-body robots include:
- tilted rabbit (Fig. 8c), with 3 body parts being rigidly and serially connected but not co-planar; and
- rigid object with 13 parts, resembling a common chair (Fig. 8d).

The articulated robots include:
- snake-like robot (Figs. 8b and 8e) which is serially configured with one movable base and 3 links (totally 9 degrees of freedom); and
- tree-like robot (Fig. 8a), which is a tree structure with one movable base in the middle and 3 branches of RRR-typed serial linkages (totally 15 degrees of freedom).

The planning environments being considered include:
- sparse map (Fig. 8a) with only two obstacles;
- cluttered map (Fig. 8b) with obstacles in arbitrary poses;
- spatial maze map (Fig. 8c) with more narrow corridors;
- home map (Fig. 8d) that is constructed as a two-floor house with walls, corridors, stairs and tables; and
- narrow window map (Fig. 8e) that includes one wall with a small window available for the robot to move through.

B. Parameter settings for planners

The benchmark studies are conducted between our proposed HRM-based planners and several widely-used sampling-based planners from OMPL, i.e., PRM [2], Lazy PRM [5], RRT [3], RRT Connect [4] and EST [51]. Moreover, different sampling methods that enhance the effectiveness for PRM are also compared, including uniform random sampling (Uniform), obstacle-based sampling (OB) [52], Gaussian sampling (Gaussian) [53], bridge test (Bridge) [6] and maximized clearance sampling (MC). We conduct 50 planning trials per planner per map. A time limit of 300 seconds is set for one planning trial for all planners. A planning trial is considered failure if the time exceeds this limit. The following describes the parameters used for both HRM-based and sampling-based planners.

1) Parameters for our proposed HRM-based planners:

Table III shows the parameters for the HRM-based planners. For both HRM and Prob-HRM, the numbers of sweep lines (i.e., $N_{\text{Line}}$) in each C-layer for the same map are pre-defined to be the same. In 3D case, $N_{\text{Line}}$ is a multiplication of the numbers of lines along $x$ and $y$ axes, i.e., $N_{\text{Line}}(N_x \times N_y)$ in the table. $N_{\text{Line}}$ is only defined for HRM planner, the number of which for all the maps is 60 – the number of discrete rotations in $\text{SO}(3)$ (as explained in Sec. IV-B). Note that only the Prob-HRM is benchmarked in the narrow map, therefore $N_{\text{Layer}}$ is not specified in the table.
Fig. 8: Demonstration of path planning solutions using our proposed HRM-based planners for different types of robots in different environments. Problems with rigid-body and articulated robots are planned using HRM and Prob-HRM respectively. Obstacles are 3D superquadrics and the robots are constructed by unions of 3D ellipsoids. The magenta curve represents the solved path of the robot base center that is projected from C-space to Euclidean space.

### TABLE III: Parameters for HRM-based planners

| Map            | $N_{Layer}$ | $N_{Line}$  |
|----------------|-------------|-------------|
| 3D Sparse      | 60          | 28 (7 x 4)  |
| 3D Cluttered   | 60          | 700 (35 x 20) |
| 3D Maze        | 60          | 700 (35 x 20) |
| 3D Home        | 60          | 400 (20 x 20) |
| 3D Narrow      | –           | 700 (35 x 20) |

2) Parameters for sampling-based planners: For sampling-based planners, since the majority of time is spent on collision checking, the choice of a relatively fast collision checker is essential. We choose the open-source and widely-used library, i.e., “Flexible Collision Library (FCL)” [54], as an external plug-in for the sampling-based planners. In particular, a special and efficient collision object from FCL is applied for ellipsoidal parts of the robot, where 12 vertices are pre-defined to bound the exact ellipsoidal surface. Furthermore, for superquadrics, their surfaces are discretized as triangular meshes based on the parametric expressions (the body that this discrete surface encloses can be seen as a convex polyhedron). The collision objects are generated as meshes a priori, and the collision queries are made online by only changing the transformations of each body part. Since the efficiency and accuracy of collision checking highly depend on the quality of discretization, we provide a statistical evaluation to determine the number of vertices for the discrete superquadric surface. The number of vertices on each superquadric surface is chosen as 100 (see supplementary materials for detailed statistical study). To make the comparison relatively fair, the same number of 100 vertices are chosen to discretize the closed-form Minkowski sums boundaries in each C-layer for HRM-based planners.

### C. Results

This subsection shows the benchmark results. We mainly compare the total running time and the success rate of solving a path planning problem. For the completeness, we also show the averaged numbers of vertices ($N_{Vertex}$) and edges ($N_{Edge}$) on the graph/tree structures and the number of vertices on the solution paths ($N_{Path}$) for different algorithms. Also, for the sake of clarity, we only display the data structure results for some selected scenes, and for the planners with uniform samplers (please refer to the supplementary document for a full list of comparisons).

1) Benchmark results for SE(3) rigid-body planning problems: The comparisons of total running time and success rate are shown in Fig. 9. The planning time for each algorithm is shown as a box plot, i.e., in Figs. 9a, 9b, 9c and 9d. The success rates are shown as bar plots, i.e., in Figs. 9e, 9f, 9g and 9h. Note that, for our proposed HRM planner and PRM-variants, the total running time at each trial includes both planning and searching phases. Table IV shows the comparisons of the graph/tree sizes between HRM and sampling-based planners in two typical scenes, i.e., rabbit robot in sparse map and chair object in home map. The resulting numbers of vertices and edges information are averaged and rounded to the nearest integer.

6The red line inside the each box is the median of the data, while the upper and lower edges of the box show the 25%-th and 75%-th percentile respectively. The dashed lines extend to the most extreme data points excluding the outliers. And the outliers are plotted as + signs.
Fig. 9: Running time and success rate comparisons between HRM and sampled-based motion planners. The running time includes both planning and searching phases for graph-based planners. PRM variants use different sampling strategies, which are formatted as “PRM (sampler name)”.

**TABLE IV: Comparisons of the size of data structures for HRM and sampling-based planners**

| Map (Robot)       | Planner | $N_{\text{Vertex}}$ | $N_{\text{Edge}}$ | $N_{\text{Path}}$ |
|-------------------|---------|---------------------|-------------------|-------------------|
| Sparse (Rabbit)   | HRM     | 3818                | 6258              | 16                |
| Sparse (Rabbit)   | PRM     | 13                  | 74                | 4                 |
| Sparse (Rabbit)   | Lazy PRM| 132                 | 1019              | 13                |
| Sparse (Rabbit)   | RRT     | 30                  | 29                | 9                 |
| Sparse (Rabbit)   | RRT Connect| 11          | 10                | 8                 |
| Sparse (Rabbit)   | EST     | 85                  | 84                | 10                |
| Home (Chair)      | HRM     | 32419               | 66575             | 95                |
| Home (Chair)      | PRM     | 2350                | 18977             | 35                |
| Home (Chair)      | Lazy PRM| 18311              | 167910            | 59                |
| Home (Chair)      | RRT     | 6374                | 6373              | 39                |
| Home (Chair)      | RRT Connect| 6285       | 6284              | 42                |
| Home (Chair)      | EST     | 2411                | 2410              | 51                |

**TABLE V: Comparisons of the size of data structures for Prob-HRM and sampling-based planners**

| Map (Robot)     | Planner       | $N_{\text{Vertex}}$ | $N_{\text{Edge}}$ | $N_{\text{Path}}$ |
|-----------------|---------------|---------------------|-------------------|-------------------|
| Cluttered (Snake)| Prob-HRM  | 7871                | 14159             | 49                |
| Cluttered (Snake)| PRM         | 76                  | 398               | 9                 |
| Cluttered (Snake)| Lazy PRM    | 257                 | 1992              | 14                |
| Cluttered (Snake)| RRT         | 53                  | 52                | 10                |
| Cluttered (Snake)| RRT Connect| 34                  | 33                | 10                |
| Cluttered (Snake)| EST         | 487                 | 486               | 21                |
| Narrow (Tree)   | Prob-HRM     | 46931               | 111150            | 21                |
| Narrow (Tree)   | PRM          | 1725                | 11698             | 16                |
| Narrow (Tree)   | Lazy PRM    | 31055               | 298000            | –                 |
| Narrow (Tree)   | RRT         | 4104                | 4103              | 10                |
| Narrow (Tree)   | RRT Connect | 3760                | 3759              | 14                |
| Narrow (Tree)   | EST         | 2822                | 2821              | –                 |

2) Benchmark results for higher dimensional articulated robot planning problems: Figures 10 and 11 show the computational time and success rate results for articulated robots in higher dimensional planning problems, respectively. Table V shows the comparisons of the graph/tree structures between Prob-HRM and sampling-based planners. Two types of environment (i.e., snake-like robot in cluttered map and tree-like robot in narrow map) with planners using uniform sampler are displayed. Since the shapes of the robot for Prob-HRM are sampled randomly, the averaged and rounded number of C-layers is computed. On average, the cluttered map results in 6 C-layers and the narrow map results in 84 C-layers.

**D. Analysis on the planning results**

Sampling-based planners are very efficient when the environments are sparse (such as in Figs. 9a, 10a, 10b, etc.). However, they become slower as the space occupied by obstacles increases. This is because most of the sampled configurations are discarded in the regions including narrow passages. Therefore, the rejection sampling process iterates much longer than in sparser environments. Also, the success rates of sampling-based planners decrease as the environment becomes denser. In cases like in Figs. 11f and 11h, some planners cannot even find any solutions within the assigned time limit of 300 seconds. For graph-based algorithms, even
with the help of effective biased samplers, they still take longer time to finally find a valid path. The tree-based planners are much more efficient in sparse and cluttered maps, which have higher chances of success and faster speed in single queries. And even in the maze map, when the dimensions of the problems increase, both RRT and RRT-connect planners can still search for a valid path efficiently. But when it comes to more complex maps like the home and narrow environments, both the speed and success rate for the tree-based probabilistic planners start to drop.

The graph information and running time show that the proposed HRM-based planners are able to generate more collision-free configurations in a relatively short period of time. The sizes of the resulting data structures are much larger than those from sampling-based planners (as in Tabs. IV and V). The main reason is that both HRM and Prob-HRM planners try to explore the whole C-layer by sweeping lines in a fixed resolution. Much more valid vertices are then generated and connected compared to the stochastic sampling methods. This might be inefficient for sparser environments. But it increases the chance of searching a valid path in difficult regions. From Figs. 9c, 9d, 10f and 10h, the planning speeds are faster than sampling-based planners in complicated environments. The success rates among multiple planning trials are also higher, as in Figs. 9g, 9h, 11f and 11h. These results show the advantages of the proposed HRM-based planners in solving narrow passage problems. From Fig. 9, HRM planner performs more stable among different trials in rigid-body planning problems, which is mainly due to its deterministic nature. Prob-HRM planner, on the other hand, has larger variances in planning time for articulated robots (such as in Fig. 10c). Furthermore, our proposed HRM and Prob-HRM are both graph-based planners, which have the properties of answering multiple queries in a fixed planning scene. They are competitive in solving complex problems with tree-based single-query planners (as in Figs. 9b, 10e and 10g), and outperforms all planners in environments with narrow corridors (as in Figs 9d, 10f and 10h). This is desirable since ours can not only build the roadmap efficiently but also answer planning queries multiple times when the environment does not change. The Prob-HRM can also solve higher dimensional problems via randomly sampling the robot shapes.

VII. PHYSICAL EXPERIMENTS ON WALKING PATH PLANNING FOR A HUMANOID ROBOT

In order to demonstrate the capabilities of our proposed planning framework in the real-world setting, in this section, physical experiments with a NAO humanoid robot [55] are conducted. The task is to guide the robot to walk through environments with several objects on the floor in random poses. The robot is required to avoid them in order to pass this cluttered space. Therefore, the problem is simplified into a
consists of three main modules: perception, planning, and the shape and pose of the fitted superellipse, respectively. The factor \( ab \) is added here in order to impose the smallest possible superellipse that fits the data [42]. The initial guess is chosen as a minimum volume enclosing ellipsoid [36] computed from the convex hull of the projected data points.

The environmental data, including the shapes and poses information of arena (pre-defined) and obstacles (computed from point cloud in the perception module), is given as inputs to the planning module. By manually selecting the start and goal poses of the robot, a valid SE(2) path is then solved by the proposed HRM planner. The parameters for solving the planning problems in three example experimental scenarios are summarized in Tab. VI.

Finally, given a list of SE(2) poses interpolated from the solved valid path, the robot follows the path via a simple proportional controller [59]. The robot pose is tracked by an ArUco tag attached to its head and is controlled to minimize the distance with the next way point on the trajectory until reaching the goal configuration.

### A. Experimental settings

The whole scene is firstly captured from a fixed RGB-D camera as point cloud data. The point cloud is transformed from the camera frame into the world frame (indicated by an ArUco marker [56] on the floor), and segmented into disjoint clusters using Point Cloud Library (PCL) [57], [58]. Each cluster is then projected onto the x-y plane and fitted into a superelliptical model. The superelliptical model can then be obtained by solving the optimization problem, i.e.,

\[
\min_{a,b,r,\theta,t} \quad ab \sum_{i=1}^{m} \left( \Phi^*(x'_i, y'_i) - 1 \right)^2 ,
\]

where \( \Phi^*(x'_i, y'_i) \) is the implicit expression for a superellipse, \((x'_i, y'_i)\) is the transformed data point as viewed in the body frame of the superellipse and \( m \) is the number of points of in the projected point cloud cluster. The optimizer \((a^*, b^*, c^*)\) and \((\theta^*, t^*)\) denotes the shape and pose of the fitted superellipse, respectively. The factor \( ab \) is added here in order to impose the smallest possible

### B. Results

Since the planning scene does not change during the whole trial of the experiment, the perception and planning modules...
computing the C-obstacle boundary). The resulting Minkowski one body (such as the superquadric obstacle body when closed-form expression only depends on the parameters of and difference that explicitly characterizes the C-space. The work is the closed-form parameterization of Minkowski sum

A. Advantageous properties of our proposed framework

One of the highlights of our proposed path planning framework is the closed-form parameterization of Minkowski sum and difference that explicitly characterizes the C-space. The closed-form expression only depends on the parameters of one body (such as the superquadric obstacle body when computing the C-obstacle boundary). The resulting Minkowski sum/difference boundary can be directly obtained. Therefore, the computational complexity is linear with respect to only one body, not both ones as the traditional polytope-based Minkowski sum [24].

In addition, the effectiveness of the sweep line method on a single C-layer provides a way to avoid traditional collision detection computation in generating collision-free samples. In the vertex generating process, vertices computed in each C-layer are automatically guaranteed to be safe. This is because they are all created outside the boundaries of C-obstacles by computing free intervals on each sweep line. Using this sweep line process, the whole space on each C-layer is explored. Moreover, when connecting an edge between two vertices within one C-layer, the whole edge is checked for intersections with C-obstacle boundaries (if a straight-line connection is considered). This is a continuous way of performing validity check, since no interpolation along the edge is required.

The idea of the “bridge C-layer” gives a connection strategy for vertices in adjacent C-layers. The process tries to sweep an enlarged tightly-fitted ellipsoidal (TFE) void to enclose all the possible intermediate configurations between the two vertices. The bridge C-layer for this TFE adds another C-layer, which doubles the total number of C-layers but reduces the dimension when validating the connections between two vertices. In other words, connecting path in the bridge C-layer can be viewed as a projection of the SE(3) (or SE(3) × (S^1)^n) sequence of the robot onto an \( \mathbb{R}^3 \) path of the void. When computing the TFE, interpolations between two SO(3) (or SO(3) × (S^1)^n) configurations are still required. But they are only computed once before connecting two C-layers. When the bridge C-layer is constructed, the validation of transition remains to check the validity of interpolated points an \( \mathbb{R}^3 \) space. The sweep volume generates a conservative safety margin for the possible motions of the robot. But its tightness makes the extra space as small as possible. The path in bridge C-layer can be subjected to any user-defined motion primitives a priori, making the layer connection method flexible for different types of robot motions.

The deterministic nature of HRM makes its speed stable over different benchmark trials on rigid-body geometric planning problems. The Prob-HRM planner, on the other hand, integrates the shining features of the probabilistic ideas in sampling-based algorithms. Comparing to HRM, the number of robot shapes sampled in Prob-HRM is unknown a priori. But as shown in the benchmark results, the final numbers of C-layers are within a tractable range. This is mainly because that Prob-HRM still preserves the deterministic nature when exploring each C-layer, which increases the chance of identifying difficult regions. The collaboration with sampling-based planners avoids the dimensionality explosions for higher degrees-of-freedom robots, making our framework extendable to wider and more complicated tasks.

both run offline. The control module runs as an on-board process to keep the robot on the solved path. Table VII shows the planning results in different example trials of experiments. Fig. 13 demonstrates the walking sequences of NAO for the three different planning scenarios.

VIII. DISCUSSIONS

From the benchmark results, our proposed Highway RoadMap (HRM) planner and its hybrid probabilistic variant Prob-HRM are able to efficiently solve geometric path planning problems with both rigid and articulated multi-body robots in different scenarios. Physical experiments further demonstrate the applicability of the proposed planning framework as a part of the robotic system in the real-world setting. This section discusses the advantages of using HRM-based planners in solving geometric path planning problems, especially in narrow passages, followed by some potential limitations.

A. Advantageous properties of our proposed framework

The deterministic nature of HRM makes its speed stable over different benchmark trials on rigid-body geometric planning problems. The Prob-HRM planner, on the other hand, integrates the shining features of the probabilistic ideas in sampling-based algorithms. Comparing to HRM, the number of robot shapes sampled in Prob-HRM is unknown a priori. But as shown in the benchmark results, the final numbers of C-layers are within a tractable range. This is mainly because that Prob-HRM still preserves the deterministic nature when exploring each C-layer, which increases the chance of identifying difficult regions. The collaboration with sampling-based planners avoids the dimensionality explosions for higher degrees-of-freedom robots, making our framework extendable to wider and more complicated tasks.

The idea of the “bridge C-layer” gives a connection strategy for vertices in adjacent C-layers. The process tries to sweep an enlarged tightly-fitted ellipsoidal (TFE) void to enclose all the possible intermediate configurations between the two vertices. The bridge C-layer for this TFE adds another C-layer, which doubles the total number of C-layers but reduces the dimension when validating the connections between two vertices. In other words, connecting path in the bridge C-layer can be viewed as a projection of the SE(3) (or SE(3) × (S^1)^n) sequence of the robot onto an \( \mathbb{R}^3 \) path of the void. When computing the TFE, interpolations between two SO(3) (or SO(3) × (S^1)^n) configurations are still required. But they are only computed once before connecting two C-layers. When the bridge C-layer is constructed, the validation of transition remains to check the validity of interpolated points an \( \mathbb{R}^3 \) space. The sweep volume generates a conservative safety margin for the possible motions of the robot. But its tightness makes the extra space as small as possible. The path in bridge C-layer can be subjected to any user-defined motion primitives a priori, making the layer connection method flexible for different types of robot motions.

The deterministic nature of HRM makes its speed stable over different benchmark trials on rigid-body geometric planning problems. The Prob-HRM planner, on the other hand, integrates the shining features of the probabilistic ideas in sampling-based algorithms. Comparing to HRM, the number of robot shapes sampled in Prob-HRM is unknown a priori. But as shown in the benchmark results, the final numbers of C-layers are within a tractable range. This is mainly because that Prob-HRM still preserves the deterministic nature when exploring each C-layer, which increases the chance of identifying difficult regions. The collaboration with sampling-based planners avoids the dimensionality explosions for higher degrees-of-freedom robots, making our framework extendable to wider and more complicated tasks.

The idea of the “bridge C-layer” gives a connection strategy for vertices in adjacent C-layers. The process tries to sweep an enlarged tightly-fitted ellipsoidal (TFE) void to enclose all the possible intermediate configurations between the two vertices. The bridge C-layer for this TFE adds another C-layer, which doubles the total number of C-layers but reduces the dimension when validating the connections between two vertices. In other words, connecting path in the bridge C-layer can be viewed as a projection of the SE(3) (or SE(3) × (S^1)^n) sequence of the robot onto an \( \mathbb{R}^3 \) path of the void. When computing the TFE, interpolations between two SO(3) (or SO(3) × (S^1)^n) configurations are still required. But they are only computed once before connecting two C-layers. When the bridge C-layer is constructed, the validation of transition remains to check the validity of interpolated points an \( \mathbb{R}^3 \) space. The sweep volume generates a conservative safety margin for the possible motions of the robot. But its tightness makes the extra space as small as possible. The path in bridge C-layer can be subjected to any user-defined motion primitives a priori, making the layer connection method flexible for different types of robot motions.

The deterministic nature of HRM makes its speed stable over different benchmark trials on rigid-body geometric planning problems. The Prob-HRM planner, on the other hand, integrates the shining features of the probabilistic ideas in sampling-based algorithms. Comparing to HRM, the number of robot shapes sampled in Prob-HRM is unknown a priori. But as shown in the benchmark results, the final numbers of C-layers are within a tractable range. This is mainly because that Prob-HRM still preserves the deterministic nature when exploring each C-layer, which increases the chance of identifying difficult regions. The collaboration with sampling-based planners avoids the dimensionality explosions for higher degrees-of-freedom robots, making our framework extendable to wider and more complicated tasks.
### TABLE VII: Planning results for HRM planner in different experiments

| Scene | \(N_{\text{Vertex}}\) | \(N_{\text{Edge}}\) | \(N_{\text{path}}\) | Graph time | Search time | Total time |
|-------|-----------------|-----------------|-----------------|------------|------------|------------|
| 1     | 4710            | 8459            | 37              | 55.71 ms   | 3.18 ms    | 58.90 ms   |
| 2     | 3246            | 5960            | 104             | 32.13 ms   | 1.96 ms    | 34.09 ms   |
| 3     | 2063            | 3562            | 53              | 17.62 ms   | 1.07 ms    | 18.69 ms   |

B. Limitations

The current HRM and Prob-HRM are both effective when the robot motions are dominated by translations. But they are not advantageous for robots with a fixed base such as manipulators. Prob-HRM can possibly be used to solve problems with pure rotational motions. But in this case, useful operations within a single C-layer might be very limited, since no translational connections can be made within one layer. Thus, the advantages of closed-form Minkowski operations might not be significant. When the robot base is fixed, Prob-HRM is equivalent to a pure sampling-based planner. In this case, the proposed closed-form Minkowski operations and the sweep line method can be used to generate valid vertices during the C-space exploration. And the “bridge C-layer” method can be applied as the transition validity checker between adjacent C-layers.

IX. Conclusion

This article proposes a path planning framework based on the closed-form characterization of Minkowski sum and difference. The important “narrow passage” problem can be solved efficiently by the proposed extended Highway RoadMap (HRM) planner and its hybrid probabilistic variant Prob-HRM. Collision-free configurations are generated directly by a “sweep line” process. And connections between two configurations with the same rotational components can be validated without interpolations. Configurations with different rotational components are connected through a novel “bridge C-layer” method using the sweep volume of enlarged ellipsoidal voids. Benchmarks are conducted among the proposed HRM-based planners and some famous sampling-based planners from the Open Motion Planning Library. The proposed HRM-based planners outperform the baseline sampling-based planners in complex environments, e.g., maze, home and narrow maps. Physical experiments on 2D path planning problems for a humanoid robot are also conducted to show the applicability of the proposed framework in some real-world settings.

The deterministic property of HRM makes it stable in solving complex rigid-body problems for multiple queries. Prob-HRM solves higher dimensional problems by combining the efficient explicit descriptions of C-space and the effectiveness of random sampling. This hybrid idea can thereby achieve better performance in higher dimensional (articulated robot) motion planning problems in cluttered environments with narrow passages.

Acknowledgement

The authors would like to thank Dr. Yan Yan and Mr. Yuanfeng Han for useful discussions. This work was performed under National University of Singapore Startup grants R-265-000-665-133 and R-265-000-665-731, National University of Singapore Faculty Board funds C-265-000-071-001, National Science Foundation of United States grant IIS-1619050 and United States Office of Naval Research award N00014-17-1-2142. The ideas expressed in this paper are solely those of the authors.

References

[1] S. Ruan, Q. Ma, K. L. Poblete, Y. Yan, and G. S. Chirikjian, “Path planning for ellipsoidal robots and general obstacles via closed-form
This section derives the closed-form Minkowski sums and difference between one ellipsoid and one arbitrary convex differentiable surface in details.

A. Derivations of the closed-form Minkowski operations

Assume that $S_1$ is a convex and differentiable hyper-surface embedded in $\mathbb{R}^d$, with implicit and parametric forms being

$$\Phi(\mathbf{x}) = 1 \quad \text{and} \quad \mathbf{x} = f(\mathbf{\psi}),$$

where $\Phi(\cdot)$ is a real-valued differentiable function of $\mathbf{x} \in \mathbb{R}^d$ and $f$ is a differentiable $d$-dimensional vector-valued functions of $\mathbf{\psi} = [\psi_1, \psi_2, \ldots, \psi_{d-1}]^T \in \mathbb{R}^{d-1}$. Let $E_2$ be an ellipsoid in general orientation in $\mathbb{R}^d$, with semi-axis lengths $a_2 = [a_1, a_2, \ldots, a_n]^T$. Then, the implicit and explicit equations are of the form

$$\mathbf{x}^T A_2^{-2} \mathbf{x} = 1 \quad \text{and} \quad \mathbf{x} = A_2 \mathbf{u}(\mathbf{\psi}),$$

where $A_2 = R_2 \Lambda(a_2) R_2^T$ is the shape matrix of $E_2$ where $R_2 \in SO(d)$ denotes the orientation of the ellipsoid, and $\Lambda(\cdot)$ is a diagonal matrix with the semi-axis length $a_i$ at the $(i, i)$ entry. Note that $A_2^{-2} = (A_2^T)^{-1} = (A_2^{-1})^2$ is used here for the sake of simplicity. Here, $\mathbf{u}(\mathbf{\psi})$ is the standard parameterization of the $d$-dimensional unit hyper-sphere using $d-1$ angles $\psi$, i.e., $\mathbf{u}^{2D}(\theta) = [\cos \theta, \sin \theta]^T$ in the 2D case and $\mathbf{u}^{3D}(\eta, \omega) = [\cos \eta \cos \omega, \cos \eta \sin \omega, \sin \eta]^T$ in the 3D case.

By applying an affine transformation, i.e.,

$$T = R_2 \Lambda(r/a_2) R_2^T,$$

the ellipsoid can be shrunk into a sphere with radius $r = \min\{a_1, a_2, \ldots, a_n\}$, and the surface $S_1$ can be transformed as $\mathbf{x}' = T \mathbf{x}$. The affine transformation matrix is symmetric and positive definite since $\Lambda(r/a_2)$ is diagonal and positive definite.

The implicit expression for the “shrunk” $S_1$, denoted as $S_1'$, is $\Phi(T^{-1} \mathbf{x}') = 1$. The Minkowski sum between $S_1'$ and $E_2'$ (now is a sphere), is obtained by computing the boundary of the offset surface with offset radius $r$ as

$$\mathbf{x}_{offs} = \mathbf{x}' + rn',$$

where $n' = \frac{\nabla \mathbf{x} \Phi(T^{-1} \mathbf{x}')}{\| \nabla \mathbf{x} \Phi(T^{-1} \mathbf{x}') \|}$ is the outward normal of the surface. With the help of the chain rule for derivatives, the numerator can be further simplified as

$$\nabla \mathbf{x} \Phi(T^{-1} \mathbf{x}') \bigg|_{\mathbf{x}' = T \mathbf{x}} = T^{-T} \nabla \mathbf{x} \Phi(\mathbf{x}).$$

(12)

In our special case, $T$ is a symmetric positive definite matrix, so we have $T^{-T} = (T^{-1})^T = (T^T)^{-1} = T^{-1}$. The Minkowski sum between the original surface $S_1$ and ellipsoid $E_2$ can be given by “stretching” the transformed space back, using inverse affine transformation, as

$$\mathbf{x}_{mb} = T^{-1} \mathbf{x}_{offs} = T^{-1} \left( T \mathbf{x} + r \frac{T^{-T} \nabla \mathbf{x} \Phi(\mathbf{x})}{\| T^{-T} \nabla \mathbf{x} \Phi(\mathbf{x}) \|} \right)$$

$$= \mathbf{x} + r \frac{T^{-2} \nabla \mathbf{x} \Phi(\mathbf{x})}{\| T^{-T} \nabla \mathbf{x} \Phi(\mathbf{x}) \|}.$$

(13)
Further simplification of Eq. (13) shows that the radius can be eliminated. And using the fact that the magnitude of a vector is preserved under rotation, the general form of the closed-form Minkowski sums between an ellipsoid and a general convex differentiable surface in $\mathbb{R}^d$ can be computed as

$$x_{mb} = x + R_2 \Lambda^2 (a_2) R_2^T \nabla_x \Phi(x) \| \Lambda (a_2) R_2 \nabla_x \Phi(x) \|. \tag{14}$$

The Minkowski sum boundary $x_{mb}$ is characterized directly from the closed-form expression, which is an one-to-one mapping with the parameterization of $S_1$. Therefore, the complexity of our proposed closed-form Minkowski sums only depends on the complexity of $S_1$. On the other hand, the Minkowski difference $S_1 \ominus E_2$ can thus be obtained by switching the plus signs in Eqs. (13) and (14) to minus.

From Eq. (14), when the pose of superquadric $S_1$ and the orientation of ellipsoid $E_2$ are fixed, the parameters derived from unknown variables are the parametric surface of $S_1$, i.e., $x$ and its gradient $\nabla_x \Phi(x)$. Therefore, in the following, we provide necessary and explicit calculations of these two parameters that defines the closed-form Minkowski sums between an ellipsoid and a superquadric surface in both 2D and 3D cases.

**B. The 2D case**

The implicit and explicit equations for a superellipse $S_1$ in $\mathbb{R}^2$ are defined as

$$\Phi(x_1, y_1) = \left( \frac{x_1}{a_1} \right)^{\frac{2}{\epsilon_1}} + \left( \frac{y_1}{b_1} \right)^{\frac{2}{\epsilon_1}} = 1, \quad \text{and} \quad x = \left( \frac{x_1}{a_1}, \frac{y_1}{b_1} \right) = \left( \frac{a_1 \cos \theta}{b_1 \sin \theta}, \frac{\omega}{\epsilon_1} \right), \quad -\pi \leq \theta \leq \pi,$$

respectively. The shape described by the above function changes with $\epsilon$. We only consider the case of $0 < \epsilon < 2$ to ensure the convexity of the corresponding shape. The gradient of $\Phi(x_1(\theta), y_1(\theta))$ with respect to the parameter $\theta$ can be computed as

$$\nabla \Phi(x_1(\theta), y_1(\theta)) = \frac{2}{\epsilon_1} \left( \cos^{2-\epsilon_1} \theta / a_1 \sin^{2-\epsilon_1} \theta / b_1 \right). \tag{15}$$

**C. The 3D case**

The implicit and explicit equations for a superquadric surface $S_1$ in $\mathbb{R}^3$ are defined as

$$\Phi(x) = \left( \frac{x_1}{a_1} \right)^{\frac{2}{\epsilon_1}} + \left( \frac{y_1}{b_1} \right)^{\frac{2}{\epsilon_2}} + \left( \frac{z_1}{c_1} \right)^{\frac{2}{\epsilon_3}} = 1, \quad \text{and} \quad x = \left( \frac{x_1}{a_1}, \frac{y_1}{b_1}, \frac{z_1}{c_1} \right), \quad \eta \in [-\pi/2, \pi/2], \quad \omega \in [-\pi, \pi],$$

respectively, where $x = [x_1, y_1, z_1]^T$. The surface described by the above function changes with $\epsilon_1$, $\epsilon_2$, and $\epsilon_3$, and we only consider the case of $0 < \epsilon_1, \epsilon_2 < 2$ to ensure the convexity of the superquadric surface. The gradient of $\Phi(x_1(\eta, \omega), y_1(\eta, \omega), z_1(\eta, \omega))$ with respect to the parameters $\eta$ and $\omega$ can be computed as

$$\nabla \Phi(\eta, \omega) = \frac{2}{\epsilon_1} \left( \cos^{2-\epsilon_1} \eta \cos 2 - \epsilon_2 \omega / a_1 \right. \sin^{2-\epsilon_1} \eta / c_1 \right). \tag{16}$$

**D. Derivations of the minimum volume concentric ellipsoid (MVCE)**

Suppose the two ellipsoids, $E_a$ and $E_b$, have semi-axis lengths $a$ and $b$, respectively. The shape matrices can be expressed as $A = R_a \Lambda^2 (a) R_a^T$ and $B = R_b \Lambda^2 (b) R_b^T$, and the parametric expressions are $x_a = R_a \Lambda (a) u$ and $x_b = R_b \Lambda (b) u$, respectively.

It has been shown in the derivations of the closed-form Minkowski operations that one ellipsoid (i.e., $E_b$) can be shrunk into a sphere ($E_b'$) via the affine transformation $T = R_b \Lambda (r / b) R_b^T$, where $r$ is the radius. Then shape matrix for $E_a$ in shrunk space, i.e., $E_a'$, can be computed as

$$A' = T^{-1} R_a \Lambda^2 (a) R_a^T T^{-1}. \tag{17}$$

Then, using singular value decomposition (SVD), its semi-axis lengths and orientation, i.e., $a'$ and $R_a'$, can be obtained respectively.

Now, the problem becomes finding an ellipsoid $E_m'$ that fully encloses an ellipsoid $E_a'$ and a sphere $E_b'$ with the same center. The semi-axes are aligned with those of $E_a'$ and $E_b'$, and their lengths are set as the maximum values between elements in $a'$ and the radius $r$ of $E_b'$. The shape matrix for $E_m'$ can be obtained as

$$M' = R_a' \Lambda^2 (\text{max}(a', r)) R_a'^T, \tag{18}$$

where $\text{max}(a', r, \ldots, \text{max}(a_d', r))^T$ and $a' \doteq [a_1', a_2', \ldots, a_d']^T \in \mathbb{R}^d$.

Finally, we stretch the space back by applying the inverse affine transformation $T^{-1}$, and get the shape matrix of the MVCE, $E_m$, as

$$M = T M' T^{-1} = R_b \Lambda (r / b) R_b^T R_a' \Lambda^2 (\text{max}(a', r)) R_a'^T R_b \Lambda (r / b) R_b^T \tag{19}$$

The semi-axis lengths and the orientation can also be obtained by SVD of $M$. All the above derivations are valid in any dimension.

Furthermore, this computational procedure can be performed iteratively if there are multiple concentric ellipsoids. For example, the MVCE that enclose the previous two ellipsoids, along with the next ellipsoid, can be enclosed by a new MVCE. The final resulting ellipsoid encapsulates all the original set of ellipsoids.

**E. Proof of volume minimality of MVCE**

This subsection proves the volume minimality of the concentric ellipsoid $E_m$ that contains both $E_a$ and $E_b$ via Eq. (20).

In the shrunk space, it is clear that $E_m'$ reaches the minimum volume when its semi-axes are aligned with those of $E_a'$.
because of the symmetry of $E'_a \cup E'_b$. Then, the implicit expressions of $E'_m$ can be written as

$$\Phi_{E'_m}(x) = x^\top R'_a \Lambda^{-2}(m') R'_a^\top x,$$

(21)
because of the alignment of the semi-axes. Now we apply a change of variables as $y = R'_a^\top x$, then Eq. (21) becomes

$$\Phi_{E'_m}(y) = y^\top \Lambda^{-2}(m') y = \sum_{i=1}^d \frac{y_i^2}{m_i^2}.$$  

(22)
The implicit expressions for $E'_a$ and $E'_b$, after the change of variables, are

$$\Phi_{E'_a}(y) = \sum_{i=1}^d \frac{y_i^2}{a_i^2} \text{ and } \Phi_{E'_b}(y) = \sum_{i=1}^d \frac{y_i^2}{b_i^2},$$

(23)
respectively.

Firstly we show that when $m_i' = \max(a'_i, r)$ ($i = 1, \ldots, d$), $E'_m$ contains both $E'_a$ and $E'_b$. Suppose an arbitrary point $y_0$ lies inside or on the boundary of $E'_a$, then $\Phi_{E'_m}(y_0) = \sum_{i=1}^d \frac{y_i^2}{a_i'^2} \leq 1$. Then, since $a'_i \leq \max(a'_i, r) = m_i'$, we have $\Phi_{E'_m}(y_0) = \sum_{i=1}^d \frac{y_i^2}{m_i'^2} \leq \sum_{i=1}^d \frac{y_i^2}{a_i'^2} \leq 1$, which means that the point $y_0$ is also inside or on the boundary of $E'_m$. The same procedure can be apply when $y_0$ is contained in $E'_b$.

We further show, by contradiction, that to ensure the containment, $m_i'$ cannot be smaller than $\max(a'_i, r)$ ($i = 1, \ldots, d$). We assume that an arbitrary $k^{th}$ semi-axis of $E'_m$ satisfies $m_k' < \max(a_k', r)$. Now, without loss of generality, we suppose $a_k' \geq r$, then $m_k' < \max(a_k', r) = a_k'$. And if $y_k = a_k' e_k$ ($e_k$ is the $k^{th}$ basis vector of $\mathbb{R}^d$) is on the boundary and the $k^{th}$ semi-axis of $E'_a$, then $\Phi_{E'_a}(y_k) = \frac{a_k'^2}{a_k'^2} = 1$. Substituting $y_k$ into Eq. (21) gives $\Phi_{E'_m}(y_k) = \frac{a_k'^2}{m_k'^2} > 1$, which implies that $y_k$ is outside $E'_m$, and therefore contradicts that $E'_m$ contains both $E'_a$ and $E'_b$.

Combining the above statements, we conclude that the minimum values of the semi-axes lengths that make $E'_m$, containing both $E'_a$ and $E'_b$, is $m_i'^* = \max(a_i', r)$ ($i = 1, \ldots, d$). According to the volume expressions for $d$-dimensional ellipsoids, we have

$$Vol^*(E'_m) = \eta \prod_{i=1}^d \max(a_i'^*, r),$$

(24)
where $\eta$ is the volume of a unit sphere in $\mathbb{R}^d$. From the property of affine transformation, the corresponding volume of $E'_m$, with the equal sign holds, in the original space is

$$Vol(E'_m) = \eta | \det(T)| \prod_{i=1}^d \max(a_i', r) \quad \text{where } \eta = \frac{\eta}{\prod_{i=1}^d b_i \max(a_i', r)}.$$

(25)
Since $\det(T)$ is constant, the minimality is preserved, thus $Vol(E'_m^*)$ is minimal, which concludes the proof.

Note that, no matter which ellipsoid to be shrunk into a sphere, the results will be the same: even though the shrunk spaces are generated by different affine transformations, i.e., $T_a$ and $T_b$, the covering ellipsoids $E_{ma}$ and $E_{mb}$ always have the minimum volume in the corresponding space. Therefore, after the inverse affine transformation, the resulting $E_{ma}$ and $E_{mb}$ both have minimum volume in the original space, which indicates the uniqueness of the result, i.e., $Vol(E'_m) = Vol(E_{ma}) = Vol(E_{mb})$.

To quantify the quality of discretization, relative values between the volume of the discrete polyhedron approximation and the exact volume are computed, i.e.,

$$\kappa = \frac{Vol_{poly}}{Vol_{SQ}},$$

(26)
where,

$$Vol_{SQ} = 2a_1a_2a_3\epsilon_1\epsilon_2\beta(\frac{\epsilon_1}{2} + 1, \epsilon_1)\beta(\frac{\epsilon_2}{2} + \frac{\epsilon_2}{2}),$$

(27)
where $a_1, a_2, a_3$ are the semi-axes lengths, $\epsilon_1, \epsilon_2$ are the exponents and $\beta(\cdot)$ is the Beta function, i.e., $\beta(x, y) = 2 \int_0^{\pi/2} \sin^{2x-1}\phi \cos^{2y-1}\phi \, d\phi$. We compute the relative volume for different numbers of vertices on the surface, i.e., from 16 to 400. Then for each discretization number, we randomly generate 50 different superquadric shapes by changing the parameters such as semi-axes lengths and exponents. Figure 14 shows the statistical plot of the discretization quality for different vertex resolutions. After around 100 vertices, the relative volume starts to be plateaued and above 90%, so for the experiments, we choose 100 as the number of vertices for the superquadric surface.

This section summarizes the sizes of data structures for different planners in different benchmark scenarios. Table VIII shows the comparisons for $\text{SE}(3)$ rigid-body planning scenarios using HRM planner. Table IX shows the comparisons for $\text{SE}(3) \times (S^1)^3$ planning scenarios for snake-like robot using Prob-HRM planner. Table X shows the comparisons for $\text{SE}(3) \times (S^1)^3$ planning scenarios for tree-like robot using Prob-HRM planner. The numbers in the tables are ordered as “$N_{\text{Vertex}}, N_{\text{Edge}}, N_{\text{Path}}$.”
### TABLE VIII: Full list of data structure size comparisons between HRM and sampling-based planners for SE(3) rigid-body planning problems

| Planner         | Sparse (Rabbit) | Cluttered (Rabbit) | Maze (Rabbit) | Home (Chair) |
|-----------------|-----------------|--------------------|---------------|--------------|
| HRM             | 3818, 6258, 16  | 11627, 18279, 48   | 33288, 57406, 207 | 32419, 66575, 95 |
| PRM (Uniform)   | 13, 74, 4       | 2724, 21279, 20    | 12807, 103780, 30 | 2350, 18977, 35 |
| PRM (OB)        | 126, 596, 9     | 2942, 15783, 20    | 13016, 83352, 57 | 3085, 18219, 44 |
| PRM (Gaussian)  | 23, 114, 6      | 943, 6535, 16      | 6247, 48223, 29  | 2381, 18176, 33 |
| PRM (MC)        | 12, 59, 5       | 2263, 17474, 19    | 13247, 107440, 33 | 2352, 18950, 38 |
| PRM (Bridge)    | 55, 242, 8      | 748, 4042, 17      | 10882, 76652, 33 | 2539, 16881, 43 |
| Lazy PRM        | 132, 1019, 13   | 22366, 202260, 29  | 15751, 135320, – | 18311, 167910, 59 |
| RRT             | 30, 29, 9       | 710, 709, 14       | 5381, 5380, 24   | 6374, 6373, 39 |
| RRT Connect     | 11, 10, 8       | 311, 310, 15       | 2411, 2410, 28   | 6285, 6284, 42 |
| EST             | 85, 84, 10      | 464, 463, 18       | 1572, 1571, 31   | 2411, 2410, 51 |

### TABLE IX: Full list of data structure size comparisons between Prob-HRM and sampling-based planners of SE(3) \(\times (S^1)^3\) planning problems for snake-like articulated robot

| Planner         | Sparse | Cluttered | Maze | Home | Narrow |
|-----------------|--------|-----------|------|------|--------|
| Prob-HRM        | 299, 564, 20 | 7871, 14159, 49 | 2814, 5047, 87 | 19902, 43355, 157 | 4454, 10586, 38 |
| PRM (Uniform)   | 3, 5, 2 | 76, 398, 9 | 515, 3341, 18 | 2902, 21328, 36 | 295, 2502, 11 |
| PRM (OB)        | 3, 4, 2 | 78, 346, 9 | 506, 2996, 18 | 2654, 17633, 33 | 319, 2344, 11 |
| PRM (Gaussian)  | 3, 4, 2 | 129, 519, 13 | 1096, 5884, 26 | 3658, 21011, 40 | 261, 1449, 12 |
| PRM (MC)        | 3, 4, 2 | 66, 325, 9 | 456, 2941, 18 | 2742, 20192, 33 | 243, 2025, 10 |
| PRM (Bridge)    | 3, 4, 2 | 100, 404, 12 | 596, 3261, 20 | 2192, 13135, 36 | 104, 532, 9 |
| Lazy PRM        | 49, 277, 10 | 257, 1992, 14 | 2800, 22119, 41 | 21187, 190240, – | 28156, 272910, 18 |
| RRT             | 26, 25, 8 | 53, 52, 10 | 181, 180, 17 | 4215, 4214, 37 | 307, 306, 10 |
| RRT Connect     | 8, 7, 6 | 34, 33, 10 | 143, 142, 18 | 3548, 3547, 37 | 146, 145, 10 |
| EST             | 3, 2, 2 | 487, 486, 21 | 1209, 1208, 46 | 3486, 3485, 74 | 381, 380, 9 |

### TABLE X: Full list of data structure size comparisons between Prob-HRM and sampling-based planners of SE(3) \(\times (S^1)^9\) planning problems for tree-like articulated robot

| Planner         | Sparse | Cluttered | Maze | Home | Narrow |
|-----------------|--------|-----------|------|------|--------|
| Prob-HRM        | 315, 557, 20 | 6651, 10757, 51 | 46931, 111150, 21 |
| PRM (Uniform)   | 8, 24, 4 | 290, 1360, 13 | 1725, 11698, 16 |
| PRM (OB)        | 11, 32, 4 | 258, 1072, 14 | 2139, 11433, 16 |
| PRM (Gaussian)  | 15, 37, 5 | 697, 2577, 23 | 2472, 10528, 18 |
| PRM (MC)        | 9, 27, 3 | 269, 1259, 13 | 1852, 12553, 18 |
| PRM (Bridge)    | 14, 37, 4 | 377, 1462, 16 | 1921, 9203, 17 |
| Lazy PRM        | 75, 459, 11 | 1008, 7735, 21 | 31055, 290000, – |
| RRT             | 26, 25, 8 | 59, 58, 11 | 4104, 4103, 10 |
| RRT Connect     | 10, 9, 7 | 49, 48, 11 | 3760, 3759, 14 |
| EST             | 17, 16, 5 | 893, 892, 26 | 2822, 2821, – |