Efficient Path Planning in Narrow Passages for Robots With Ellipsoidal Components

Sipu Ruan, *Member, IEEE*, Karen L. Poblete, Hongtao Wu, *Graduate Student Member, IEEE*, Qianli Ma, and Gregory S. Chirikjian, *Fellow, IEEE*

***Abstract*—Path planning has long been one of the major research areas in robotics, with probabilistic roadmap (PRM) and rapidly-exploring random trees (RRT) being two of the most effective classes of planners. Though generally very efficient, these sampling-based planners can become computationally expensive in the important case of "narrow passages." This article 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. Each environmental feature is represented geometrically using a strictly convex body with a $\mathcal{C}^1$ boundary (e.g., superquadric). 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 needing to perform traditional collision detection. Furthermore, by combining with a 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 the proposed framework often outperforms the sampling-based planners in terms of computational time and success rate in finding a path through narrow corridors for both single-body robots and those with higher dimensional configuration spaces. Physical experiments using the proposed framework are further demonstrated on a humanoid robot that walks in several cluttered environments with narrow passages.**

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

Manuscript received 28 November 2021; revised 28 April 2022; accepted 13 June 2022. Date of publication 27 July 2022; date of current version 8 February 2023. This work was supported in part by the National Research Foundation, Singapore, through its Medium Sized Centre Program—Centre for Advanced Robotics Technology Innovation—under Subaward R-261-521-002-592, in part by the Singapore Ministry of Education Tier 1 Grant R-265-000-655-114, in part by the National University of Singapore (NUS) Startup under Grant R-265-000-665-133 and Grant R-265-000-665-731, in part by NUS Faculty Board funds under Grant C-265-000-071-001, in part by the U.S. National Science Foundation under Grant IIS-1619050, and in part by the U.S. Office of Naval Research under Award N00014-17-1-2142. An earlier version of this article was presented in part at the 2018 International Workshop on Algorithmic Foundations of Robotics (WAFR) [DOI: 10.1007/978-3-030-44051-0_1] [1]. This article was recommended for publication by Associate Editor J. O'Kane and Editor W. Burgard upon evaluation of the reviewers' comments. *(Corresponding author: Gregory S. Chirikjian.)*

Sipu Ruan and Gregory S. Chirikjian are with the Department of Mechanical Engineering, National University of Singapore, Singapore 119077 (e-mail: ruansp@nus.edu.sg; mpegre@nus.edu.sg).

Hongtao Wu is with the Laboratory for Computational Sensing and Robotics, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: hwu67@jhu.edu).

Karen L. Poblete is with the Epic Systems, Verona, WI 53593 USA (e-mail: karenpoblete2002@msn.com).

Qianli Ma is with the Motional, Inc., Pittsburgh, PA 15207 USA (e-mail: qianli.ma622@gmail.com).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2022.3187818.

Digital Object Identifier 10.1109/TRO.2022.3187818

I. Introduction

SAMPLING-BASED planners, such as probabilistic roadmap (PRM) [2] and rapidly-exploring random trees (RRT) [3] (and a multitude of their extensions, e.g., [4] and [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] (Section 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 article is to *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 represented by unions of strictly convex bodies with $\mathcal{C}^1$ boundaries.*

In our proposed path planning framework, the robot is encapsulated by a union of ellipsoids. The configuration spaces (C-spaces) to be considered are $\mathrm{SE}(d)$ and $\mathrm{SE}(d) \times (S^1)^n$ for rigid-body and articulated robots, respectively.[1] 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] [see Fig. 1(a)]. In computational crystallography, it is natural to approximate a protein molecule

[1]$\mathrm{SE}(d)$, $d = 2, 3$ is the pose of the robot base frame and $(S^1)^n$ represents the C-space of n revolute joints.

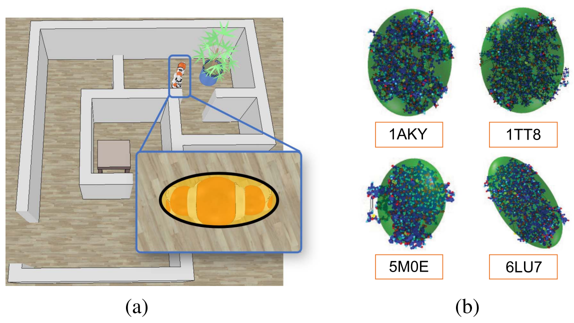


Fig. 1. Examples of robots and protein molecules represented by ellipsoids. (a) Projection contour of a NAO humanoid robot is enclosed by an ellipse (yellow). (b) Protein atom elements (small balls) are approximated by ellipsoids (green).

by a moment-of-inertia ellipsoid, which simplifies the complex geometric models and maintains the physical information of the protein [11] [see Fig. 1(b)]. Moreover, superquadrics are chosen as examples to represent environmental features. This family of shapes generalizes ellipsoids by adding freedoms in choosing the power of the exponents rather than restricting to quadratics. It represents a wider range of the complex shapes (e.g., cuboids, cylinders, etc.) while also requiring only a few parameters [12].

When a robot is fixed at a certain orientation and internal joint angles, a "slice" of the 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-slice" [15]. (Section II-B reviews the literature in detail on the computations of Minkowski sums). Once the C-space obstacles (C-obstacles) are computed, the complement region between the planning arena[2] and the union of C-obstacles is the free space that allows the robot to travel through safely. 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 planners like PRM or RRT, explicit collision checking is still required. Therefore, the second goal of this article is to *develop guaranteed safe and efficient methods for connecting configurations between different C-slices without performing explicit collision checking between pairwise bodies.*

A "bridge C-slice" idea is proposed as a local planner to guarantee safe transitions between different C-slices. The name suggests that a new C-slice is built as a bridge between two adjacent C-slices. To efficiently construct a bridge C-slice, an enlarged void for each ellipsoidal robot part is computed in closed form. Here, a "void" is the free space that fully contains the robot part, ensuring that it moves without collisions. A sweep volume is then constructed to enclose the robot at all the possible intermediate configurations during 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 planners suffer from the curse of dimensionality burden in the case of articulated robots. Therefore, the third goal of this article is to *develop an effective method to tackle the exponential computational complexity for the planning of articulated robots.*

[2]Here, the word "arena" denotes the bounded area in which the robot and obstacles are contained.

A hybrid algorithm called "Probabilistic Highway RoadMap (Prob-HRM)" is proposed here to make planning in higher dimensional C-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-slice from HRM.

This article extends the conference version [1] on the same topic and has significant updates. Compared to the conference paper, the key contributions of this article are as follows.

1) This article extends the graph construction procedure in each C-slice to the 3-D multibody case.
2) It introduces a novel "bridge C-slice" method to connect vertices between adjacent C-slices.
3) It proposes a hybrid planner, which integrates the advantages of sampling-based planners on higher dimensional articulated robot planning problems.
4) It conducts rigorous benchmark simulations and physical experiments in challenging environments to evaluate the proposed planning framework.

These extensions are essential since more general 3-D and articulated robot models are implemented. The benchmark and physical experimental settings are also more realistic.

The rest of this article is organized as follows. Section II reviews related literature. Section III provides mathematical foundations. Section IV extends our previously proposed HRM planner to the case of 3-D multibody robot with ellipsoidal components. The novel "bridge C-slice" method is then introduced. Section V introduces the hybrid Prob-HRM planner. Section VI conducts extensive benchmarks with some popular and successful sampling-based planners. 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 proposed framework in Section VIII. Finally, Section IX concludes this article.

II. Literature Review

This section reviews related work on the key topics that this article addresses.

A. Challenge of Narrow Passages

One of the key factors that affect the performance of sampling-based planners is the random state sampling strategy. To tackle the "narrow passage" challenge, various sampling strategies have been studied throughout these years, many of which try to capture the local features around obstacles.

The bridge test [6] finds a collision-free middle point between configurations that are in collision with the obstacles. UOBPRM [16] searches for collision-free samples from a configuration in collision by moving in different ray directions. In [8], a Bayesian learning scheme is used to model sampling distributions. It subsequently updates the previous samples by maximizing the likelihood from the region that has higher probability in forming a valid path within the narrow passage. Ideas about generating samples on the "medial axis" were proposed in [17] and [18]. Each sampled state, regardless of free or in-collision, is retracted to the medial axis of free space. The retraction direction is selected between the sampled state and its nearest neighbor on the boundary of free space. The resulting samples stay far from obstacles. And the usage of in-collision samples is able to detect regions close to narrow passages. The proposed framework in this article also attempts to generate vertices that stay away from obstacles as far as possible. A similar idea is used in the "maximize clearance" sampler, i.e., PRM(MC), in the benchmark studies of this article. For each valid sample, the sampler searches a new sample close-by but with larger distance to the obstacles. We use PRM(MC) for comparisons since it is implemented on the well-known Open Motion Planning Library (OMPL) [19]. This provides a standardized way to benchmark with other sampling-based planning algorithms as well as samplers.

Other methods combine the advantages of different kinds of algorithms. For example, Toggle PRM [20] 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 [21] tries to retract initial samples into more difficult regions, so as to increase the probability of sampling near narrow passages. More recently, a reinforcement learning method has been applied to enhance the ability to explore local regions where the tree grows [22].

Hybrid planner [15] combines a random sampling strategy with Minkowski sum computations, which increases the probability of identifying narrow regions. In this article, we use an approach with some similarities to the Prob-HRM planner 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 generate in the middle of free space in a more uniform way. And to connect different C-slices, they add a new vertex and search for paths on the C-obstacle boundaries, but we instead generate a new slice based on an enlarged void.

B. *Computations of Minkowski Sums*

The Minkowski sum is ubiquitous in many fields such as computational geometry [23], robot motion planning [13], control theory [24], etc. Despite its straightforward definition, which will be given in Section III, computing an exact boundary of Minkowski sum between two general nonconvex polytopes in $\mathbb{R}^3$ can be as high as $O(N_1^3 N_2^3)$, 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 [25], since the Minkowski sums between two convex polytopes can achieve $O(N_1 N_2)$ complexity [26]. Another type of methods is based on convolutions of two bodies, since Minkowski sum of two solid bodies is the support of the convolution of their indicator functions [27]. A simple approximated algorithm [28] is proposed that avoids computing 3-D arrangement and winding numbers via collision detection. An exact Minkowski sum for polytopes containing holes is proposed using convolution [29]. In addition, point-based methods avoid convex decomposition [30]. The major advantages are the ease of generating points than meshes and the possibility of parallelisms [31]. An exact closed-form Minkowski sum formula for d-dimensional ellipsoids was introduced [32]. And in [33], a parameterized ellipsoidal outer boundary for the Minkowski sum of two ellipsoids is proposed. This article studies a more general case when one body is an ellipsoid and the other is a strictly convex body with a $\mathcal{C}^1$ boundary (e.g., superquadric).

C. *Ellipsoids and Superquadrics for Object Representation*

Besides using polyhedra for object representations, other geometric primitives, such as ellipsoids and superquadrics, also play an important role due to their simple algebraic characterizations. Recently, in many robotic applications, they have been good candidates to encapsulate objects [34], [35].

A 3-D ellipsoid in a general pose only needs nine parameters: three for the shape (i.e., semiaxis lengths) and six for the pose. Algorithms related to ellipsoids are studied extensively [36], [37]. The minimum volume enclosing ellipsoid (MVEE), which is characterized as a convex optimization problem [38], is widely used to encapsulate a point cloud. The studies of algebraic separation conditions for two ellipsoids provide very efficient algorithms to detect collisions in both the static and dynamic cases [39], [40]. Another attractive attribute of the representation using ellipsoids is the existence of efficient procedures of computing their distance [41], [42]. Once an ellipsoid is fully contained in another, the volume of its limited available motions is computed explicitly [43].

Superquadrics can be seen as an extension of ellipsoids, with the two additional exponents determining the sharpness and convexity [12]. They are able to represent a wider range of geometries such as cube, cylinder, octahedron, etc. Using optimization or deep learning techniques, point cloud data can be segmented and fitted by unions of superquadrics [44], [45]. Proximity queries and contact detection are useful applications of this geometric model [46], [47].

III. Mathematical Preliminaries

This section provides the mathematical preliminaries 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 [48]

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

When computing the boundary in which the two bodies touch each other externally (i.e., their contact space), we refer to the calculation of $\partial[P_1 \oplus (-P_2)]$, where $-P_2$ is the reflection of P_2

as viewed in its body frame [28]. Note that when P_2 is centrally symmetric, such as ellipsoids and superquadrics that this article focuses on, the Minkowski sum boundary and contact space are equivalent. Moreover, when the bodies are nonconvex, using the fact that

$$\text{if } P_1 = Q_1 \cup Q_2, \text{ then, } P_1 \oplus P_2 = (Q_1 \oplus P_2) \cup (Q_2 \oplus P_2) \tag{2}$$

their Minkowski sums can be obtained via convex decomposition.

B. Implicit and Parametric Surfaces

Assume that S_1 is a strictly convex body bounded by a $\mathcal{C}^1$ hypersurface embedded in $\mathbb{R}^d$. The implicit and parametric forms of its surface can be expressed as

$$\Phi(\mathbf{x}_1) = 1 \text{ and } \mathbf{x}_1 = \mathbf{f}(\boldsymbol{\psi}_1) \tag{3}$$

where $\Phi(\cdot)$ is a real-valued differentiable function of $\mathbf{x}_1 \in \mathbb{R}^d$ and $\mathbf{f}$ is a differentiable d-dimensional vector-valued function of surface parameters $\boldsymbol{\psi}_1 = [\psi_1, \psi_2, \ldots, \psi_{d-1}]^\top \in \mathbb{R}^{d-1}$.

Let E_2 be an ellipsoid in $\mathbb{R}^d$ in general orientation, with semiaxis lengths $\mathbf{a}_2 = [a_1, a_2, \ldots, a_n]^\top$. Its implicit and explicit expressions are

$$\mathbf{x}_2^\top A_2^{-2} \mathbf{x}_2 = 1 \text{ and } \mathbf{x}_2 = A_2 \mathbf{u}(\boldsymbol{\psi}_2) \tag{4}$$

where $A_2 = R_2 \Lambda(\mathbf{a}_2) R_2^\top$ 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 semiaxis length a_i at the (i, i) entry. $A_2^{-2} \doteq (A_2^2)^{-1} = (A_2^{-1})^2$ is used here for the sake of simplicity. $\mathbf{u}(\boldsymbol{\psi}_2)$ is the standard parameterization of the d-dimensional unit hypersphere using $d-1$ angles. Specifically, in 2-D, $\mathbf{u}(\theta) = [\cos\theta, \sin\theta]^\top$, and in 3-D, $\mathbf{u}(\eta, \omega) = [\cos\eta \cos\omega, \cos\eta \sin\omega, \sin\eta]^\top$.

One class of strictly convex bodies meeting the conditions stated earlier includes those with specific kinds of superquadric boundaries. The implicit equations in the 2-D and 3-D cases are given by

$$\Phi(x, y) = \left(\frac{x}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\epsilon}} \text{ and} \tag{5}$$

$$\Phi(x, y, z) = \left(\left(\frac{x}{a}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{b}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{c}\right)^{\frac{2}{\epsilon_1}} \tag{6}$$

where a, b, and c are the semiaxis lengths, and $\epsilon, \epsilon_1, \epsilon_2 \in (0, 2)$ are the exponents that ensure strict convexity.

C. Closed-Form Minkowski Operations Between an Ellipsoid and a General Convex Differentiable Surface

It has been shown previously in [32] that the Minkowski sum and difference between two ellipsoids can be parameterized in closed form. The expression can be extended when one ellipsoid is substituted by S_1 [1]. The general simplified form for the Minkowski sum can be computed as

$$\mathbf{x}_{mb} = \mathbf{x}_1 + \frac{R_2 \Lambda^2(\mathbf{a}_2) R_2^\top \nabla_{\mathbf{x}_1} \Phi(\mathbf{x}_1)}{\|\Lambda(\mathbf{a}_2) R_2^\top \nabla_{\mathbf{x}_1} \Phi(\mathbf{x}_1)\|} \tag{7}$$

where $\nabla_{\mathbf{x}_1} \Phi(\mathbf{x}_1)$ is the gradient of S_1 at $\mathbf{x}_1$. The conditions that S_1 is strictly convex and its boundary is $\mathcal{C}^1$ ensure that the gradient exists and that there is never division by zero when using (7). Fig. 2 illustrates the geometric interpretation of the computational process. Detailed derivations were presented in [1].

D. Minimum Volume Concentric Ellipsoid (MVCE) Enclosing Two Ellipsoids With the Same Center

When two ellipsoids are fixed at the same center, an "MVCE" can be computed in closed form as follows.

Given two d-dimensional ellipsoids E_a and E_b with semiaxis lengths $\mathbf{a}$ and $\mathbf{b}$, respectively. One ellipsoid (e.g., E_b) can be shrunk into a sphere (E_b') via the affine transformation $T = R_b \Lambda(r/\mathbf{b}) R_b^\top$, where r is the radius and $r/\mathbf{b} \doteq [r/b_1, r/b_2, \ldots, r/b_d]^\top \in \mathbb{R}^d$. Then, the shape matrix of E_a in shrunk space, i.e., E_a', can be computed as $A' = T^{-1} R_a \Lambda^{-2}(\mathbf{a}) R_a^\top T^{-1}$. Using singular value decomposition, its semiaxis lengths and orientation, i.e., $\mathbf{a}'$ and R_a', can be obtained, respectively. The shape matrix of their MVCE, i.e., E_m, is obtained as $M = T R_a' \Lambda^{-2}(\max(\mathbf{a},' r)) R_a'^\top T$, where $\max(\mathbf{a},' r) \doteq [\max(a_1', r), \ldots, \max(a_d', r)]^\top$ and $\mathbf{a}' \doteq [a_1', a_2', \ldots, a_d']^\top \in \mathbb{R}^d$. The computational procedure is visualized in Fig. 3 for the 3-D case. The idea here is inspired by [36], which provides equivalent computations for a maximum volume concentric ellipsoid covered by two ellipsoids.

Furthermore, this process can be applied iteratively if there are multiple concentric ellipsoids. For example, the MVCE that encloses the previous two ellipsoids, along with the next ellipsoid, can be enclosed by a new MVCE. The final resulting ellipsoid encapsulates all the original sets of ellipsoids, which is denoted as a tightly fitted ellipsoid (TFE).

E. Superquadric Model Fitting to Point Cloud Data

Given a set of m 3-D points $\{\mathbf{x}_i = [x_i, y_i, z_i]^\top, i = 1, \ldots, m\}$, a superquadric model can be approximated by minimizing [45], [49]

$$\min_{a,b,c,\epsilon_1,\epsilon_2,R,\mathbf{t}} abc \sum_{i=1}^{m} \left(\Phi^{\epsilon_1}(x_i', y_i', z_i') - 1\right)^2 \tag{8}$$

where $\Phi(\cdot)$ is shown in (6), $\mathbf{x}_i' = R^\top(\mathbf{x}_i - \mathbf{t})$ is the transformed data point as viewed in the body frame of the superquadric, and $R \in \text{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$ are the orientation and center of the superquadric, respectively. The factor abc is added here in order to minimize the volume of the fitted superquadric body. Similarly, for the 2-D case, the corresponding nonlinear optimization problem can be formulated as

$$\min_{a,b,\epsilon,\theta,\mathbf{t}} ab \sum_{i=1}^{m} \left(\Phi^{\epsilon}(x_i', y_i') - 1\right)^2 \tag{9}$$

where $\Phi(\cdot)$ is now referred to (5) and θ is the rotational angle of the 2-D superellipse.

Solving the above optimizations requires good initial conditions. The parameters from MVEE are used, which can be

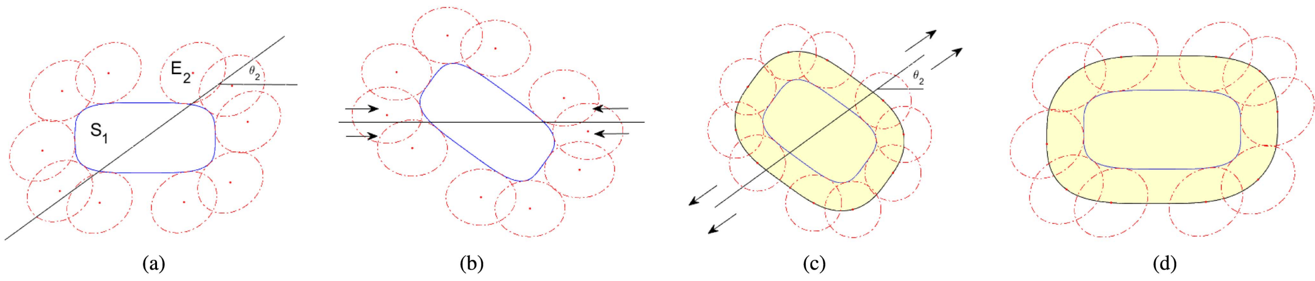


Fig. 2. Process for the characterizations of the Minkowski sums between a superquadric S_1 and an ellipsoid E_2. (a) Original space with S_1 in the center and E_2 translating around. (b) Both the bodies are rotated by the inverse orientation of E_2. (c) E_2 is shrunk into a sphere, and an offset surface is computed. (d) Stretch back and obtain $S_1 \oplus (-E_2)$ (the yellow region).

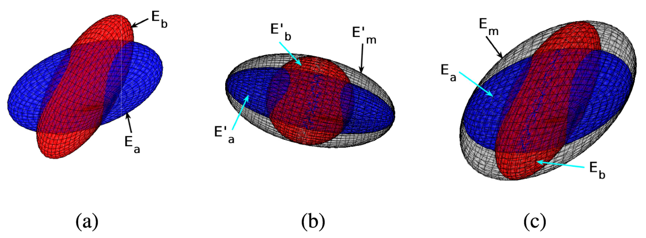


Fig. 3. Computational procedure for MVCE that covers two ellipsoids in 3-D. (a) Two concentric 3-D ellipsoids, E_a and E_b. (b) Shrink E_b into a sphere E'_b, and enclose both ellipsoids by E'_m. (c) Transform back to get MVCE E_m in the original space.

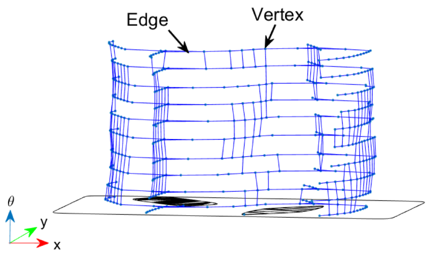


Fig. 4. Fully connected graph structure, generated from one simulation trial. The vertical axis represents the rotational angle; dots are vertices, and line segments are edges.

computed using convex optimization as [38]

$$\min_{A,\mathbf{t}} \log\det A$$
$$\text{s.t. } A \succ 0\,,$$
$$(\mathbf{x}_i - \mathbf{t})^\top A^{-2}(\mathbf{x}_i - \mathbf{t}) \le 1 \;(i = 1,\ldots,m) \qquad (10)$$

where A is the shape matrix of an ellipsoid as in (4). This convex optimization process can also be used to bound the robot parts if they are originally modeled by surface meshes.

IV. HRM Planning Algorithm for Rigid-Body Robots With Ellipsoidal Components

This section introduces the extended "HRM" algorithm. The extension to the previous work [9] from 2-D to 3-D rigid-body path planning problems is explained here. 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 a union of ellipsoids. Also, a procedure to iteratively refine the roadmap is introduced.

A. Overview of the HRM Planner

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 Fig. 4.

The input of the *robot* is a union of ellipsoids, including the body shapes and kinematic data. The kinematic data of each body part store the relative rigid-body transformation with respect to the base. The input environmental data include a set of superquadric objects that represent the *obstacles* and *arena*.

Algorithm 1: Highway RoadMap (HRM) Algorithm.

```
Inputs    : $robot$: a union of ellipsoidal objects;
            $obstacle$: a set of superquadric objects;
            $arena$: a set of superquadric objects;
            $endpts$: start and goal configurations
Parameter: $N_{\text{slice}}$: number of C-slices;
            $N_{\text{line}}$: number of sweep lines;
            $N_{\text{point}}$: number of points for interpolation
Outputs   : $roadmap$: a graph structure;
            $path$: an ordered list of configurations
1  $R \leftarrow$ SampleOrientations($N_{\text{slice}}$);
2  foreach $i < N_{\text{slice}}$ do
3      $robot$.ForwardKinematics($R_i$) ;
4      $roadmap \leftarrow$ ConstructOneSlice($robot$, $obstacle$,
         $arena$, $N_{\text{line}}$) ;
5  end
6  foreach $i < N_{\text{slice}}$ do
7      $roadmap \leftarrow$ ConnectAdjacentSlice($i$, $robot$, $R$,
         $N_{\text{point}}$);
8  end
9  $path \leftarrow$ GraphSearch($roadmap$, $endpts$);
10 while Not TerminationCondition do
11     $roadmap, path \leftarrow$ RefineExistRoadMap($robot$,
         $obstacle$, $arena$, $N_{\text{line}}$, $R$) ;
12 end
```

And the *endpts* input indicates the start and goal configurations of the robot. There are two major input parameters: the number of C-slices N_{slice} and the initial number of sweep lines at each C-slice N_{line}. These two parameters determine the initial resolution of the roadmap. N_{line} will be increased after the initial roadmap is built, but the termination condition is not reached. The outputs of the algorithm are the *roadmap* and *path*. The *roadmap* is represented as a graph structure and the *path* stores an ordered list of valid configurations from the start to the goal. This algorithm will terminate when any of the following conditions is satisfied: a valid path is found, the maximum planning time is reached, or the maximum number of sweep lines is generated.

In Algorithm 1, Line 1 generates N_{slice} of discrete rotations in SO(d) and stored *a priori*. Then, the forward kinematics is computed in Line 3 to rotate the rigid-body robot. At each fixed orientation, a subset of the C-space that only contains translation is built, denoted here as a "C-slice," in Line 4. Once all the C-slices are constructed, the vertices among adjacent C-slices are connected via a novel idea of "bridge C-slice" in Line 7. Each constructed C-slice only connects to its most adjacent C-slice. In Line 9, a graph search technique is applied to find a path from the starting configuration to the goal. In this article, A^* algorithm [50] is used. When the termination condition is not satisfied, the roadmap is refined in an iterative way in Line 11.

B. Discretization of the Robot Orientations

Line 1 of Algorithm 1 precomputes a set of orientation samples from SO(d). In the 2-D case, uniformly distributed angles within the interval $[-\pi, \pi]$ are computed. In 3-D, 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 [51]. Using this set of orientation samples, the rotational difference between two adjacent C-slices is smaller compared to nonuniform sample sets. Note that more rotations can be sampled to construct a denser roadmap per the user's choice, i.e., using the strategies proposed in [52] and [53].

C. Construction of One C-Slice

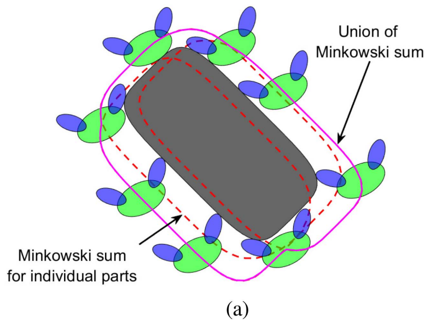
The detailed procedure to construct one single C-slice (i.e., Line 4 of Algorithm 1) is outlined in Algorithm 2. Within each C-slice, the closed-form Minkowski sum and difference for the bodies of robot are computed with the obstacles and arena, which results in C_{obstacle} and C_{arena}, respectively (Line 1). By sweeping parallel lines throughout the C-slice with a certain resolution (indicated by N_{line}), the free portion of the C-slice (C_{free}) is detected and represented as a set of line segments (Line 2). Furthermore, the middle point of each collision-free line segment is generated as the sampled vertices in the roadmap (Line 3). Then, two vertices in adjacent sweep lines attempt to be connected by collision-free edges (Line 4).

1) Minkowski Operations for a Multibody Robot: At each C-slice, the closed-form Minkowski operations are computed to generate C-obstacles (i.e., in Line 1 of Algorithm 2). The robot is constructed by a finite union of M rigidly connected ellipsoids $E_1, E_2, \ldots, E_M$. Without 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 $g_i = (R_i, \mathbf{t}_i)$ $(i = 2, \ldots, M)$, respectively. For a multilink 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 (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 $\mathbf{t}_i$ is added to (7). For practical computational purposes, each Minkowski sum and difference boundary is discretized as a convex polygon in 2-D and polyhedral mesh in 3-D. The vertices of the discrete boundary are generated using the parametric expression of Minkowski operations. Fig. 5 shows the Minkowski sums of a multibody robot at a fixed orientation [see Fig. 5(a)] and the collision-free C-space in the corresponding C-slice [see Fig. 5(b)].

2) Sweep-Line Process to Characterize Free Regions Within One C-Slice: The general idea of the "sweep-line" process (i.e., Line 2 of Algorithm 2) is analogous to raster scanning – a set of parallel lines is defined to sweep throughout the whole C-slice. Theoretically, these parallel lines can be defined along any direction. However, for simplicity of representation and storage, throughout this article, the lines are defined to be parallel to the basis axes of the coordinate system. Specifically, the sweep

Algorithm 2: ConstructOneSlice($robot$, $obstacle$, $arena$, N_{line}).

```
1:  $C_{\text{obstacle}}, C_{\text{arena}} \leftarrow$ MinkowskiOperations($robot$, $obstacle$, $arena$);
2:  $C_{\text{free}} \leftarrow$ SweepLineProcess($C_{\text{obstacle}}$, $C_{\text{arena}}$, $N_{\text{line}}$);
3:  $roadmap.vertex$.Append (GenerateVertex($C_{\text{free}}$));
4:  $roadmap.edge$.Append (ConnectOneslice($C_{\text{free}}$));
Return: $roadmap$
```



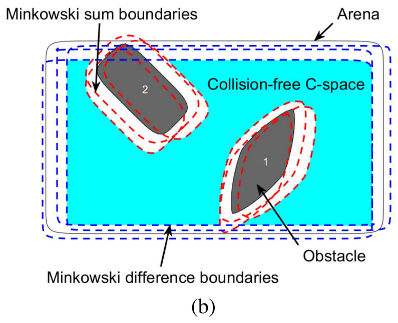


Fig. 5. Characterization of the Minkowski sum between a convex superquadric and a union of ellipsoids. (a) C-obstacle as the Minkowski sum boundaries of individual ellipsoidal bodies and their union. (b) Collision-free C-space as an intersection of free space for individual robot parts.

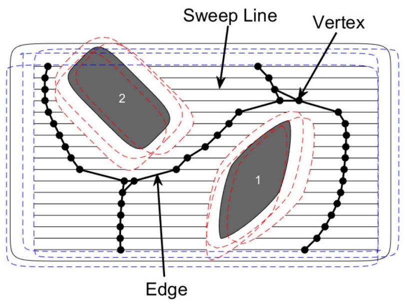


Fig. 6. Sweep line process for detecting free space and construct subgraph in one C-slice.

lines are parallel to x-axis and z-axis for the 2-D and 3-D cases, respectively. Note that, in the 3-D case, one could think the process as first sweeping planes through the 3-D translational space, then sweeping lines within each plane. However, in practice, there is no need to compute each plane completely that includes the silhouettes of C-obstacles. Instead, this article approximates each plane by a bundle of sweep lines, which are then used directly to compute free segments via line–obstacle intersections.

To generate collision-free configurations, segments on each sweep line within all C-obstacles and C-arenas are computed, denoted as L_{O} and L_{A}, respectively. Then, the collision-free segments L_{free} can be computed as [9], [54]

$$L_{\mathrm{free}} = \bigcap_{i=1}^{M_A \times M} L_{\mathrm{A_i}} - \bigcup_{j=1}^{M_O \times M} L_{\mathrm{O_j}} \tag{11}$$

where M_A and M_O are numbers of arenas and obstacles, respectively. All L_{free} are stored in C_{free} (Line 2 of Algorithm 2). Then, collision-free vertices are generated as the middle point of each L_{free} (Line 3 of Algorithm 2). Afterward, more vertices can be generated as an enhancement step. An example is given and applied throughout this article. Denote $L_{j,k}$ as the $k\mathrm{th}$ free segment of the $j\mathrm{th}$ sweep line, with $V_{j,k}$ being its corresponding middle point. First, L_{j+1,k_2} is projected onto L_{j,k_1}. If the projection overlaps with L_{j,k_1} but V_{j,k_1} is not within the overlapping segment, a new vertex within the overlapping segment that is nearest to V_{j,k_1} is added to the vertex list. The resulting new vertex is closer to V_{j+1,k_2} than V_{j,k_1} does, which gives higher chance to make the further connection success, especially in narrow regions.

Once a list of collision-free vertices is generated, the next step is to connect them (Line 4 of Algorithm 2). In this article, only two vertices in adjacent sweep lines attempt to be connected with a straight line segment. Assume that a candidate connection is attempted between V_{j,k_1} and V_{j+1,k_2}. The connection validity is checked by computing the intersections between the line segment $\overline{V_{j,k_1} V_{j+1,k_2}}$ and all meshed C-obstacles. If the segment is outside all C-obstacles, the whole edge is guaranteed to be collision free. Fig. 6 shows the decomposed C-space in one slice of a planar case. The horizontal raster lines indicate the collision-free line segments. This method provides a continuous way of validating edges within each C-slice, in the sense that the whole edge is checked without interpolation.

Algorithm 3: ConnectAdjacentSlice(i, $robot$, R, N_{point}).

```
1  $R_{\mathrm{near}} \leftarrow$ GetAdjacentSlice($i$, $R$);
2  TFE $\leftarrow$ ComputeTightlyFittedEllipsoids($robot$, $R_i$,
       $R_{\mathrm{near}}$, $N_{\mathrm{point}}$) ;
3  $C_{\mathrm{obstacle}}, C_{\mathrm{arena}} \leftarrow$ MinkowskiOperations(TFE,
       $obstacle$, $arena$);
4  foreach vertex $V_1$ in current C-slice do
5      $\{V_{1,\mathrm{near}}\} \leftarrow$ NeighborVerticesInAdjacentSlice(
           $roadmap.vertex$ ) ;
6      foreach $V_2 \in \{V_{1,\mathrm{near}}\}$ do
7          $\{V_{\mathrm{step}}\} \leftarrow$ PathInterp($V_1$, $V_2$, $N_{\mathrm{point}}$) ;
8          if IsPathValid($\{\mathbf{t}_{\mathrm{step}}\}$, $C_{\mathrm{obstacle}}$, $C_{\mathrm{arena}}$) then
9              $roadmap.edge$.Append($\{V_1, V_2\}$) ;
10         end
11     end
12 end
   Return: $roadmap$
```

D. Vertex Connections Between Adjacent C-Slices

Since each C-slice only represents one orientation of the robot, rotational motions are required when connecting different C-slices. A novel "bridge C-slice" method is proposed (i.e., in Line 7 of Algorithm 1) to guarantee that the vertices at different C-slices can be safely connected without performing explicit collision detection. Algorithm 3 outlines this new local planner. The general idea is to construct a new C-slice based on an enlarged ellipsoidal void that encloses the robot at two configurations and compute a translational sweep volume that bounds the whole transition.

1) General Ideas of the "Bridge C-Slice" Local Planner: Each C-slice only attempts to connect with one adjacent C-slice, which is searched at the beginning in Line 1 of Algorithm 3. And the metric that evaluates adjacency is based on the distance of the rotational components. In the 3-D case, for instance, the Euclidean distance between the quaternion parameterization of the two bodies is used. The core steps in Algorithm 3 are Line 2, which constructs an enlarged TFE for each robot part, and Line 8, which validates the whole edge connecting the two vertices.

Suppose that the robot is moving from vertex $V_1 = (R_1, \mathbf{t}_1)$ to $V_2 = (R_2, \mathbf{t}_2)$, where R_i and $\mathbf{t}_i$ $(i = 1, 2)$ represents the rotation and translation part of vertex V_i, respectively. The idea here is to enclose the motions of each ellipsoidal part of the robot, i.e., E_k, between the two configurations by a tightly fitted sweep volume, which is guaranteed to be collision free. The intermediate configurations between V_1 and V_2 can be computed using interpolation technique. To construct the sweep volume, a tightly fitted concentric ellipsoid (TFE) for each E_k at all orientations from the interpolated motions is computed, which will be detailed in Section IV-D2. The computed TFE is the void that guards the safe motions of the actual ellipsoidal part. Then, the computed TFE translates from $\mathbf{t}_1$ to $\mathbf{t}_2$ following the interpolated path (i.e., $\{\mathbf{t}_{\mathrm{step}}\}$) of E_k's center. The resulting sweep volume bounds the whole transition of E_k between the two configurations. To ensure that each computed TFE stays inside the collision-free space, one could query the inside–outside status of all the intermediate translation parts $\{\mathbf{t}_{\mathrm{step}}\}$ with all

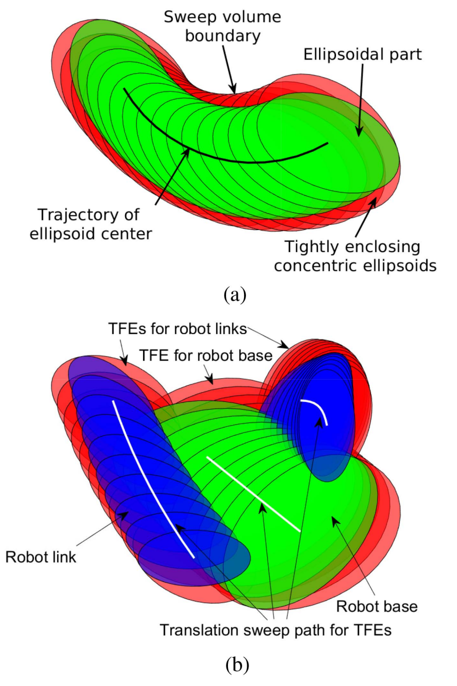


Fig. 7. 2-D example illustrating the sweep volume idea based on the sliding of TFEs. (a) Sweep volume for individual elliptical part. (b) Sweep volume for the whole multibody robot.

C-obstacles and C-arena. Then, if all the positions from $\{\mathbf{t}_{\text{step}}\}$ are valid, the sweep volume is guaranteed to be safe. Therefore, the whole transition for the ellipsoidal part E_k is collision free.

Fig. 7(a) shows the procedure of constructing the sweep volume for an individual body part. Fig. 7(b) illustrates the union of sweep volumes that encloses the whole multibody robot in the planar case. The robot base follows a 2-D straight line with rotations, and the TFEs of different body parts follow different paths (as show in white curves). In this process, the TFE for each body part translates with respect to its own center individually. This differs from the operations within one C-slice, which requires an offset to the C-obstacle and C-arena boundaries in order to make the robot as a whole rigid body. The reason is that, as shown in Fig. 7(b), the motion of each robot part is no longer a pure translation. Therefore, the reference points of Minkowski operations for different body parts have different trajectories to follow. The transition for the whole robot is guaranteed safe if all the individual reference points are within their own free space.

2) Computational Procedure for "TFEs": Line 2 of Algorithm 3 generates the TFE for each individual part of the robot. The detailed computational procedure is shown in Algorithm 4. First, an interpolation of the orientations is computed (Line 1). The number of intermediate orientations is predefined by users as the parameter N_{point}. Then, the TFE set, represented as a set of ellipsoids, is initiated as the robot at the ith orientation (Line 2). For each interpolated step, the orientations of all the robot parts are updated (Line 4). Finally, the updated TFE for each robot part is generated by computing the MVCE (introduced in Section III-D) of the current TFE and each ellipsoidal part at the new orientation (Line 6). This procedure requires N_{point}

Algorithm 4: ComputeTightlyFittedEllipsoids($robot$, R_i, R_j, N_{point}).

```
1 $\{R_{\text{step}}\} \leftarrow$ RotationInterpolation($R_i$, $R_j$, $N_{\text{point}}$) ;
2 TFE $\leftarrow$ $robot$.ForwardKinematics($R_i$) ;
3 **foreach** $R_{\text{step}}$ **do**
4     $robot$.ForwardKinematics($R_{\text{step}}$) ;
5     **foreach** $robot$ part $E_k$ **do**
6         TFE[$k$] $\leftarrow$ MVCE(TFE[$k$], $E_k$) ;
7     **end**
8 **end**
  **Return:** TFE (a set of TFEs for different robot parts)
```

iterations so that all the interpolated orientations between two C-slices can be fully encapsulated.

3) Vertex Connections Based on Bridge C-Slice Calculations: A "bridge C-slice" is constructed via closed-form Minkowski operations between the computed TFE and obstacles/arena (Line 3 of Algorithm 3). Then, the algorithm attempts to connect all the existing vertices to their nearest neighbors within the adjacent C-slice. The nearest neighbors of a vertex are defined as located within the same sweep line (Line 5 of Algorithm 3).

For each candidate connection, the robot is transformed according to the interpolated configurations between two vertices (Line 7 of Algorithm 3). Note that the rotation part of each interpolated motion needs to match those when computing TFEs (i.e., Line 1 of Algorithm 4). This is not hard to achieve for a typical interpolation of rigid-body motions, even when a simultaneous rotation and translation is considered. For example, this article uses interpolations in SE(3) of the form $g_{\text{step}} = g_1 \exp[\tau \log(g_1^{-1} g_2)]$, where $\tau \in [0, 1]$ parameterizes the transition, $g_1, g_2 \in \text{SE}(3)$ are the two end points of the interpolation, $\exp[\cdot]$ and $\log(\cdot)$ are matrix exponential and logarithm, respectively. The rotation part of each step is the same with interpolations on $\text{SO}(3)$ since the group operation for the rotation part is not affected by the translation part.

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 (Line 8 of Algorithm 3). The inside–outside status of this point with all the 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.

The sweep volume gives a conservative encapsulation of the robot transitions between two vertices. However, if the orientation samplings are incremental and uniform, there will not be a large rotational difference between adjacent C-slices. Thus, the extra free space inside the sweep volume will be small.

E. Refinement of the Existing Roadmap

Line 11 of Algorithm 1 tries to refine the existing roadmap by increasing the density of sweep lines at each existing C-slice. This process will be triggered when the termination conditions

```
Algorithm 5: RefineExistRoadMap($robot$, $obstacle$, $arena$, $N_{\text{line}}$, $R$)
1  $N_{\text{line}} = N_{\text{line}} * 2$;
2  foreach $R_i \in R$ do
3      $robot$.ForwardKinematics($R_i$);
4      $roadmap \leftarrow$ ConstructOneSlice($robot$, $obstacle$,
         $arena$, $N_{\text{line}}$);
5      $roadmap \leftarrow$ ConnectExistSlice($i$, $R$);
6      $path \leftarrow$ GraphSearch($roadmap$, $endpts$);
7      if TerminationCondition then
8          break;
9      end
10 end
   Return: $roadmap$, $path$
```

are not satisfied after building and searching the whole roadmap. Detailed procedures are presented in Algorithm 5. First, N_{line} is doubled (Line 1). Then, at each C-slice, the same procedure with Algorithm 2 that constructs one C-slice with more sweep lines is performed (Line 4). Note that the C-obstacles are stored during the initial construction of C-slices; then, they can be directly retrieved without recomputing. Afterward, the new denser set of vertices attempts to connect with the old vertices within the same C-slice (Line 5). This process first locates the vertices that have the same rotation part in the existing roadmap. Then, each new vertex attempts to connect with nearby existing vertex using the same procedure in Line 4 of Algorithm 2. Once connections are done, the graph search is performed again (Line 6).

V. Hybrid Probabilistic Variation of HRM Planner for Articulated Robots With Ellipsoidal Components

The original HRM planner in Section IV only designs for the case when robot parts are rigidly connected to each other. This limits its ability to extend to higher dimensional C-space, i.e., SE(d) $\times (S^1)^n$. To avoid the exponential computational complexity in concatenating all the 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." Algorithm 6 shows the general workflow of the proposed hybrid 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, i.e., the orientation of the base part and all the joint angles.

The robot with fixed rotational components is called a "shape" [15], and a single C-slice is computed for each robot shape. Since for each shape, the internal joint angles are fixed, computations within the same C-slice in Prob-HRM stay the same with those in HRM, i.e., Line 5 of Algorithm 6 is the same as the corresponding subroutine in Algorithm 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-slices (Line 6) with that in HRM is that the connection attempts are made only for the new C-slice in the current loop. In HRM, as a comparison, the adjacent C-slices are connected in the end after all the C-slices are generated. Also, the graph search

```
Algorithm 6: 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;
               $endpts$: start and goal configurations;
   Parameter: $N_{\text{line}}$: number of sweep lines;
               $N_{\text{point}}$: number of points for interpolation
   Outputs   : $roadmap$: a graph structure;
               $path$: an ordered list of configurations
1  $N_{\text{slice}} = 0$ ;
2  while Not TerminationCondition do
3      $R_{\text{current}} \leftarrow$ RandomSampleRobotShape();
4      $robot$.ForwardKinematics($R_{\text{current}}$);
5      $roadmap \leftarrow$ ConstructOneSlice($robot$, $obstacle$,
         $arena$, $N_{\text{line}}$);
6      $roadmap \leftarrow$ ConnectAdjacentSlice($robot$,
         $R_{\text{current}}$, $N_{\text{point}}$);
7  end
8  $path \leftarrow$ GraphSearch($roadmap$, $endpts$);
9  $R$.Append( $R_{\text{current}}$ );
10 $N_{\text{slice}} = N_{\text{slice}} + 1$ ;
11 if Refine current $roadmap$ then
12     $roadmap, path \leftarrow$ RefineExistRoadMap($robot$,
         $obstacle$, $arena$, $N_{\text{line}}$, $R$);
13 end
```

process is conducted each time after the new C-slice is connected to the graph (as in Line 8). 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 [52], followed by the 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. When N_{slice} reaches a certain number but the termination conditions are still not satisfied, the C-slice exploration is paused and the refinement of the current roadmap is triggered. In practice, this refinement process is triggered when each 60 new C-slices are generated. The refinement procedure in Line 12 is the same with Algorithm 5. Once all the existing C-slices are refined, but the algorithm is still not terminated, new C-slices exploration will be resumed. Note that N_{slice} in HRM is no longer a predefined parameter of Prob-HRM since the orientation of the robot base and joint angles is updated online.

VI. Benchmarks on Path Planning for Ellipsoidal Robots in Superquadric Environments

This section compares the performance of the proposed HRM and Prob-HRM planners with some well-known sampling-based motion planners. The proposed planners are written in C++. The baseline sampling-based planners to be compared are sourced from the "OMPL" [19]. All the benchmarks are conducted on Ubuntu 16.04 using an Intel Core i7 CPU at 3.40 GHz $\times$ 8.

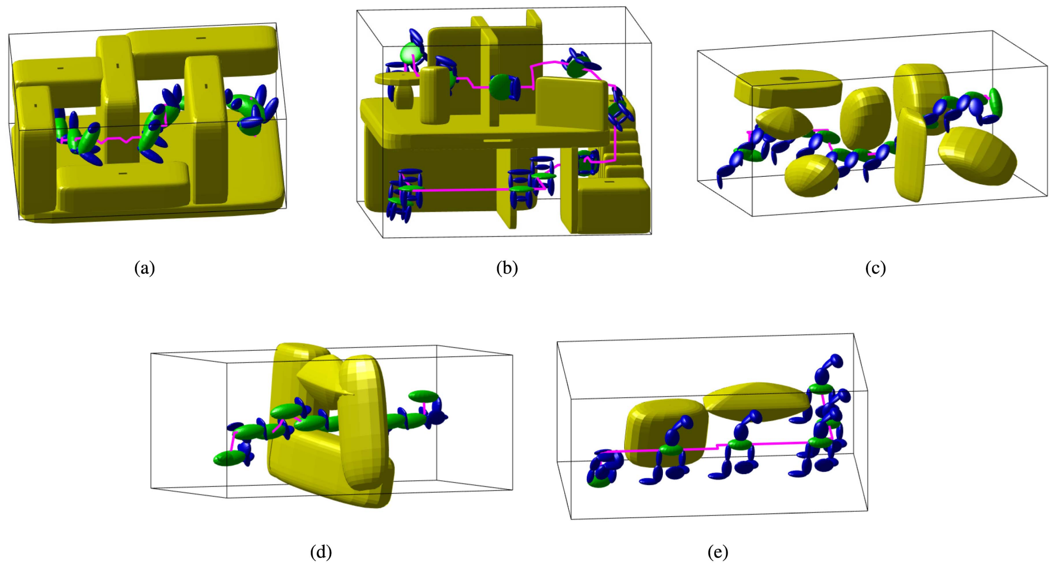


Fig. 8. Demonstration of path planning solutions using our proposed HRM-based planners for different types of robots in different environments. Problems with rigid-body and articulated robots are planned using HRM and Prob-HRM, respectively. Obstacles are 3-D superquadrics and the robots are constructed by unions of 3-D ellipsoids. The magenta curve represents the solved path of the robot base center that is projected from C-space to Euclidean space. (a) 3-D maze map, rabbit-shape robot. (b) 3-D home map, chair object. (c) 3-D cluttered map, snake-like robot. (d) 3-D narrow window map, snake-like robot. (e) 3-D sparse map, tree-like robot.

A. *Planning Environment and Robot Type Settings*

Fig. 8 shows the planning environments and the solved paths for different robots using our proposed HRM or Prob-HRM planners. Both the rigid-body and articulated robots are considered.

The rigid-body robots include the following:

1) tilted rabbit [see Fig. 8(a)], with three body parts being rigidly and serially connected but not coplanar;
2) rigid object with 13 parts, resembling a common chair [see Fig. 8(b)].

The articulated robots include the following:

1) snake-like robot [see Fig. 8(c) and (d)], which is serially configured with one movable base and three links (totally nine degrees of freedom);
2) tree-like robot [see Fig. 8(e)], which is a tree structure with one movable base in the middle and three branches of RRR-typed serial linkages (totally 15 degrees of freedom).

The planning environments being considered include the following:

1) spatial maze map [see Fig. 8(a)] with more narrow corridors;
2) home map [see Fig. 8(b)] that is constructed as a two-floor house with walls, corridors, stairs, and tables;
3) cluttered map [see Fig. 8(c)] with obstacles in arbitrary poses;
4) narrow window map [see Fig. 8(d)] that includes one wall with a small window available for the robot to move through;
5) sparse map [see Fig. 8(e)] with only two obstacles.

B. *Parameter Settings for Planners*

The compared baseline sampling-based planners from OMPL are PRM [2], Lazy PRM [5], RRT [3], RRT Connect [4], and EST [55]. Moreover, different sampling methods for PRM are also considered, including uniform random sampling (Uniform), obstacle-based sampling (OB) [56], Gaussian sampling (Gaussian) [57], bridge test (Bridge) [6], and maximized clearance sampling (MC). We conduct 50 planning trials per planner per map. A time limit of 300 s is set for one planning trial for all the planners. A planning trial is considered failure if the time exceeds this limit.

TABLE I
PARAMETERS FOR HRM-BASED PLANNERS IN SCENARIOS FROM FIG. 8

Map	Robot	N_{slice}	N_{line} $(N_x \times N_y)$
Maze	Rabbit	60	55 (11×5)
Home	Chair	60	400 (20×20)
Cluttered	Snake	–	72 (12×6)
Narrow	Snake	–	18 (6×3)
Sparse	Tree	–	10 (5×2)

1) Parameters for Our Proposed HRM-Based Planners: Table I shows the parameters of the HRM-based planners for each scenarios in Fig. 8. N_{slice} is only defined for HRM planner, as explained in Section IV-B. For scenes including articulated robot (i.e., snake and tree), N_{slice} is not a predefined parameter. The initial value of N_{line} is a parameter defined by user or computed according to the planning scenario. In the following benchmark studies, the latter case is used. Based on the sizes of the obstacles and robot parts, the initial number of lines along a certain direction (i.e., N_{dir}) is computed by

$$N_{\text{dir}} = \left\lfloor \frac{\mathbf{a}_{\text{dir}}(A) - \max_i \mathbf{a}(E_i)}{\min_j \mathbf{a}(O_j)} \right\rfloor \tag{12}$$

where $\mathbf{a}_{\text{dir}}(A)$, $\mathbf{a}(E_i)$, and $\mathbf{a}(O_j)$ denote the semiaxis length of the arena A along direction dir, an ellipsoidal robot part E_i, and an obstacle O_j, respectively. In the 3-D case, N_{line} is a multiplication of the numbers of lines along x- and y-axis directions, i.e., $N_{\text{line}}(N_x \times N_y)$.

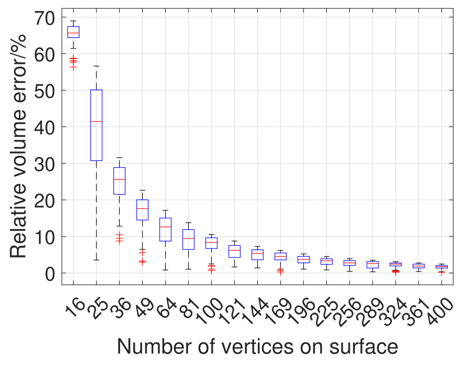


Fig. 9. Relative volume for discretized superquadrics.

2) Parameters for Sampling-Based Planners: For sampling-based planners, 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)" [58], as an external plug-in for collision detection between robot parts and obstacles. In particular, a special and efficient collision object from FCL is applied for ellipsoidal parts of the robot. The library uses 12 extreme vertices to outer bound the exact ellipsoidal surface, resulting in a discretized polyhedral model. For superquadrics, their surfaces are discretized as triangular meshes based on the parametric expressions. Then, the bodies can be seen as convex polyhedra. The collision objects are generated *a priori*, and the collision queries are made online by only changing the poses 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 evaluation metric is based on the relative volume difference between the ground truth and fitted geometries, i.e.,

$$\kappa_{\text{volume}} = \frac{|\text{Vol}_{\text{fitted}} - \text{Vol}_{\text{true}}|}{\text{Vol}_{\text{true}}} \times 100\% \tag{13}$$

where Vol_{true} and $\text{Vol}_{\text{fitted}}$ denote the volume of the ground truth and fitted geometries, respectively. Here, the ground truth is considered as the superquadric and the fitted object is the convex polyhedron. The volume of a superquadric body can be computed as $\text{Vol}_{\text{SQ}} = 2abc\epsilon_1\epsilon_2\beta(\frac{\epsilon_1}{2}+1,\epsilon_1)\beta\left(\frac{\epsilon_2}{2},\frac{\epsilon_2}{2}\right)$, where $\beta(x,y) = 2\int_0^{\pi/2}\sin^{2x-1}\phi\cos^{2y-1}\phi\, d\phi$ is the beta function. κ_{volume} are computed for different numbers of vertices on the superquadric surface. For each discretization, 100 random superquadric shapes are generated. Fig. 9 shows the statistical plot of the discretization quality for different vertex resolutions. After around 100 vertices, the error starts to be plateaued and below 10%. Therefore, we choose 100 as the number of vertices for the superquadric surface. To make the comparison relatively fair, the same number of 100 vertices is chosen to discretize the closed-form Minkowski sums boundaries in each C-slice for our HRM-based planners.

TABLE II
RESULTS OF ABLATION STUDY FOR "BRIDGE C-SLICE"

Map	Robot	HRM version	Total time (s)	N_{edge}
Sparse	Rabbit	Original	0.2779	1883
		Ablated	0.4530	1959
Cluttered	Rabbit	Original	2.238	10421
		Ablated	5.203	11206
Maze	Rabbit	Original	0.8925	2494
		Ablated	1.123	2840
Home	Chair	Original	87.31	72063
		Ablated	153.3	72785

C. Results and Analysis

An ablation study for the "bridge C-slice" subroutine is first conducted, followed by benchmark studies among the proposed HRM-based and the sampling-based planners. The benchmark results include the total time and the success rate to solve different planning problems.

1) Ablation Study for "Bridge C-Slice" Subroutine: In this study, the HRM planner is treated as the baseline because of its deterministic property. The ablated version replaces the bridge C-slice with direct interpolation between two vertices in different C-slices and collision detection at each intermediate step using FCL. The number of steps is chosen as N_{point}, the same with that in the bridge C-slice process. The average planning time and the number of edges in the graph (i.e., N_{edge}) are shown in Table II.

The original HRM with "bridge C-slice" connects less valid edges than the ablated version, which uses direct interpolation and explicit collision detection. This is mainly due to the fact that the computation of TFE for each robot part is conservative. However, the efficient computations of Minkowski sums for TFEs and point inclusion queries in the bridge C-slice help speed up the planner. Especially, in the more complex environments like cluttered and home maps, the proposed HRM runs around two times faster than the ablated version.

2) Benchmark Results for SE(3) and Higher Dimensional Planning Problems: The comparisons of total running time and success rate in $\mathrm{SE}(3)$ rigid-body planning problems are shown in Fig. 10. Figs. 11 and 12 show the computational time and success rate results for articulated robots in $\mathrm{SE}(3) \times (S^1)^n$ C-space, respectively. For our proposed HRM-based and the PRM-based planners, the total running time at each trial includes both graph construction and search phases.

From the benchmark results, sampling-based planners are very efficient when the environments are sparse (such as in Figs. 10(a) and 11(a) and (b)). However, they become slower when the space occupied by obstacles increases. Also, the success rate of sampling-based planners decreases as the environment becomes denser. In cases like in Fig. 12(f) and (h), some planners cannot even find any solution within the assigned time limit of 300 s. For graph-based algorithms, even with the help of different types of samplers, they still take longer time to finally find a valid path. The tree-based planners are much more efficient for single queries in sparse and cluttered maps. And even in the maze map, when the dimensions of the problems increase, both the RRT and RRT-connect planners can still search for a valid

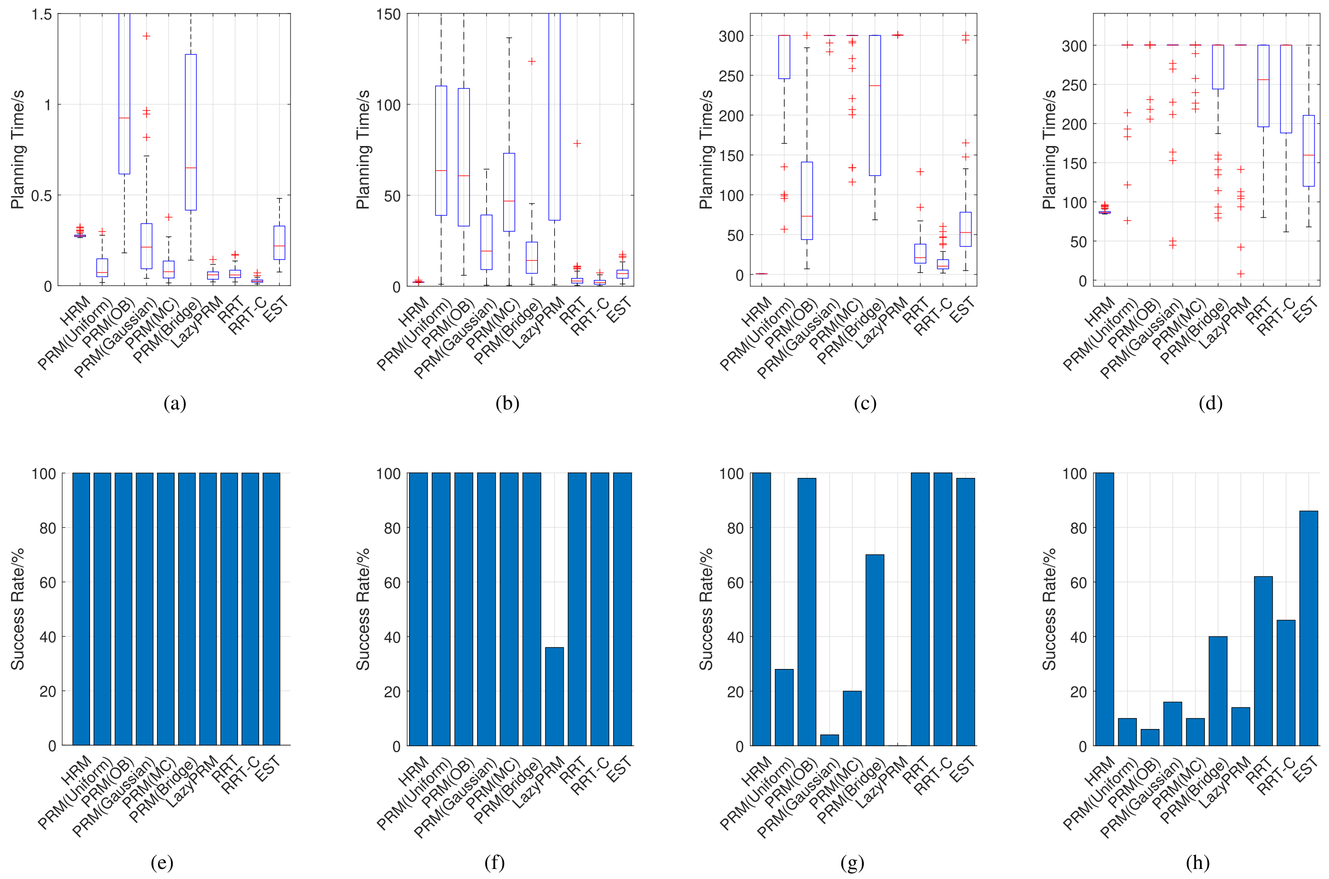


Fig. 10. Running time and success rate comparisons between HRM and sampling-based motion planners. PRM-based planners use different sampling strategies, denoted as "PRM (sampler name)." The planning time is shown as a box plot [(a)–(d)]. The red line inside the each box is the median of the data, while the upper and lower edges of the box show the 25th and 75th percentiles, respectively. The dashed lines extend to the most extreme data points excluding the outliers. And the outliers are plotted as + signs. The success rates are shown as bar plots [(e)–(h)]. (a) Run time: 3-D sparse, rabbit. (b) Run time: 3-D cluttered, rabbit. (c) Run time: 3-D maze, rabbit. (d) Run time: 3-D home, chair. (e) Success rate: 3-D sparse, rabbit. (f) Success rate: 3-D cluttered, rabbit. (g) Success rate: 3-D maze, rabbit. (h) Success rate: 3-D home, chair.

path efficiently. However, in more complex maps like the home and narrow environments, both of their speed and success rate start to drop.

On the other hand, the proposed HRM and Prob-HRM planners are more efficient in complex environments, such as in Figs. 10(c) and (d) and 11(f) and (h). The success rates among multiple planning trials are also higher, as in Figs. 10(g) and (h) and 12(f) and (h). These results show the advantages of the proposed HRM-based planners in solving narrow passage problems. Furthermore, as can be seen from Fig. 10, the performance of HRM is 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 variance in planning time for articulated robots [such as in Fig. 11(c)]. Another feature of our proposed HRM and Prob-HRM is that they are both graph-based planners. They are competitive in solving complex problems with single-query planners [as in Figs. 10(b) and 11(e) and (g)] and outperform all the planners in environments with narrow corridors [as in Figs. 10(d) and 11(f) and (h)]. 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.

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, physical experiments with a NAO humanoid robot [59] 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 planar case, where the robot and all the objects are projected onto the floor. The contour of the robot projection is encapsulated by an ellipse, with predefined semiaxis lengths. The robot is able to walk sideways, and its C-space is $SE(2)$. The arena is a predefined rectangular area, which is bounded by a superellipse with exponent defined as 0.1. The whole experimental pipeline consists of three main modules: perception, planning, and control, as shown in Fig. 13. The Robot Operating System is used to communicate between different modules.

The whole scene is first captured from a fixed RGB-D camera as point cloud data. The point cloud is transformed from the

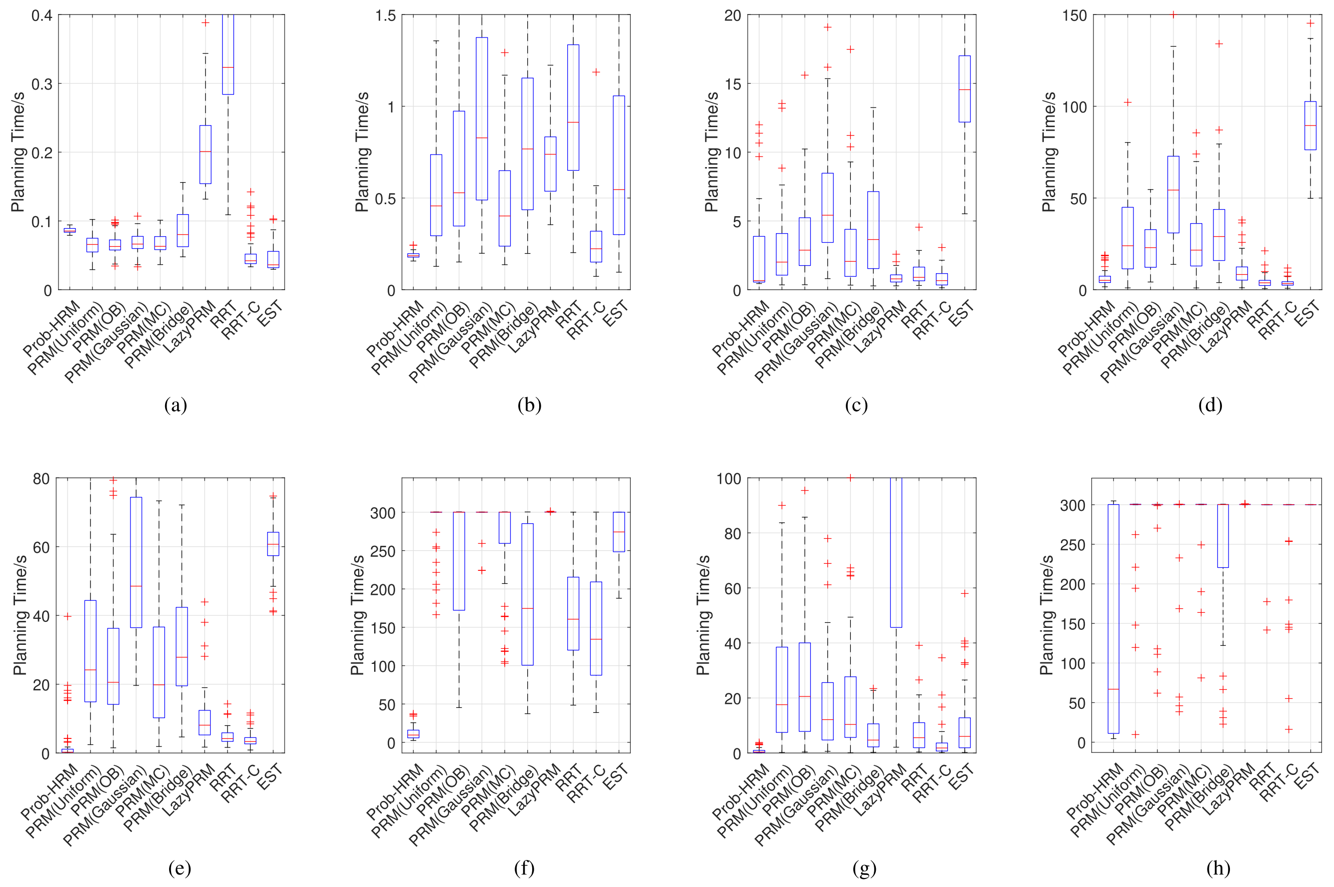


Fig. 11. Running time comparisons between Prob-HRM and sampling-based motion planners. (a) 3-D sparse, snake. (b) 3-D sparse, tree. (c) 3-D cluttered, snake. (d) 3-D cluttered, tree. (e) 3-D maze, snake. (f) 3-D home, snake. (g) 3-D narrow, snake. (h) 3-D narrow, tree.

TABLE III
NAO Walking Path Planning Results Using HRM

Scene	Graph time (ms)	Search time (ms)	Total time (ms)
1	55.71	3.18	58.90
2	17.62	1.07	18.69
3	147.46	4.53	151.99

camera frame into the world frame (indicated by an ArUco marker [60] on the floor) and segmented into disjoint clusters using Point Cloud Library [61]. Each cluster is then projected onto the xy plane and fitted into a superelliptical model using (9). The obtained environmental data are then given as the input 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. Finally, given a list of SE(2) poses, the robot follows the path via a simple proportional controller [62]. 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.

Since the planning scene does not change during the whole trial of the experiment, the perception and planning modules both run offline. The control module runs as an on-board process to keep the robot following the solved path. Table III shows the planning results in different example trials of experiments. Fig. 14 demonstrates the walking sequences of NAO for the three different planning scenarios.

VIII. Discussions

This section discusses the advantages of the proposed HRM-based planners, followed by some potential limitations.

A. *Geometric Approximations of Rigid Objects*

The superquadric is an example model to enclose the environmental features. Alternatively, the convex polyhedron is a well-known type of geometry to represent a complex body, but may require many vertices and faces to describe a rounded region. It is possible to fit a convex polyhedron with superquadric model and vice versa, which introduces approximation errors. The fitting quality is evaluated as the relative volume between the two different models as in (13).

To fit a superquadric model, the vertices of a convex polyhedron is used in (8). The evaluation metrics include not only (13), but also the averaged sum of absolute difference between the point and implicit function, i.e., $\kappa_{\text{implicit}} = \frac{1}{m}\sum_{i=1}^{m} |\Phi(\mathbf{x}_i) - 1|$, where m is the number of vertices of the convex polyhedron.

First, the convex polyhedron is treated as ground truth and generated as the convex hull of a set of 100 random points. Two types of convex polyhedra are studied: centrally symmetric and random shapes. To generate the centrally symmetric

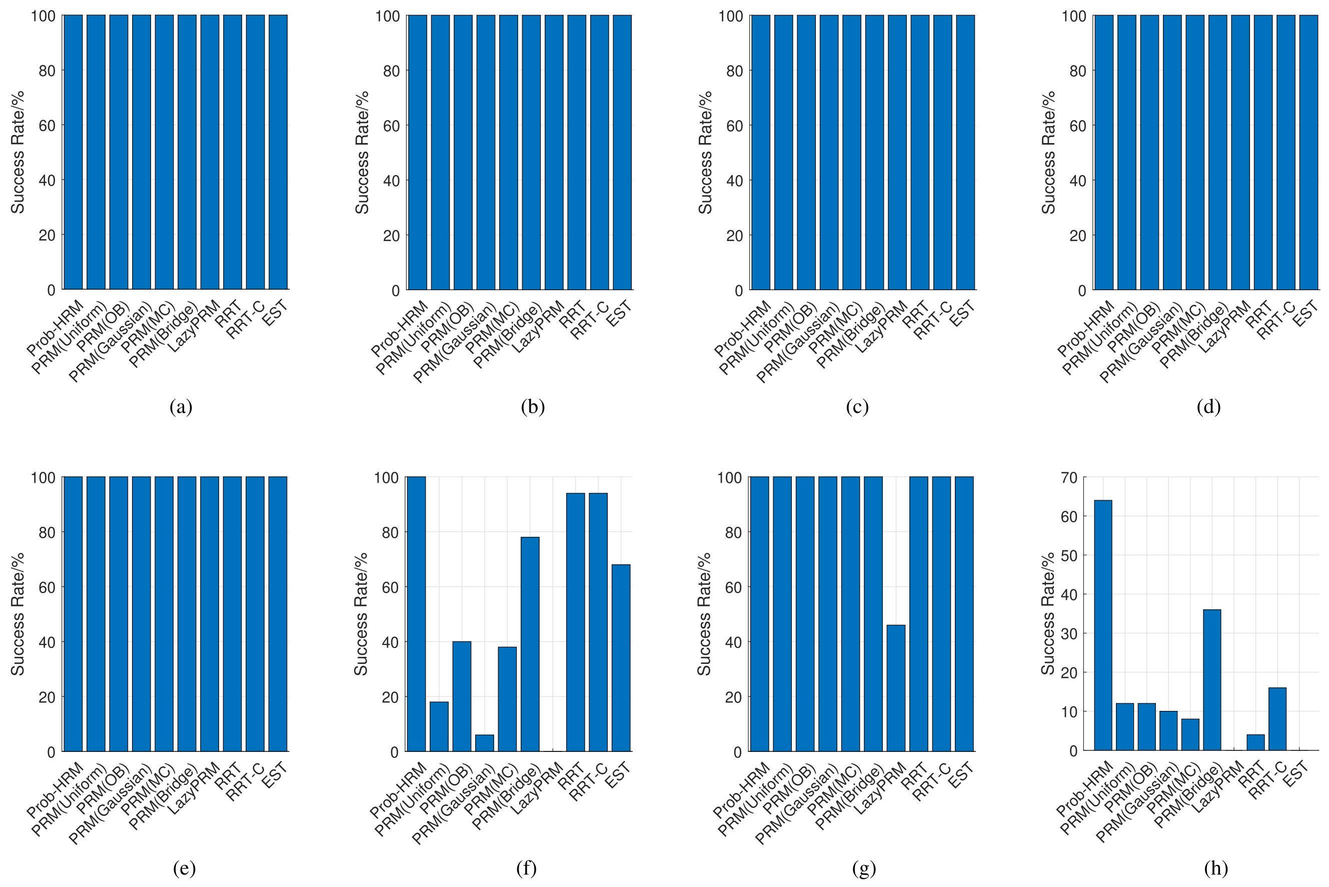


Fig. 12. Success rate comparisons between Prob-HRM and sampled-based motion planners. (a) 3-D sparse, snake. (b) 3-D sparse, tree. (c) 3-D cluttered, snake. (d) 3-D cluttered, tree. (e) 3-D maze, snake. (f) 3-D home, snake. (g) 3-D narrow, snake. (h) 3-D narrow, tree.

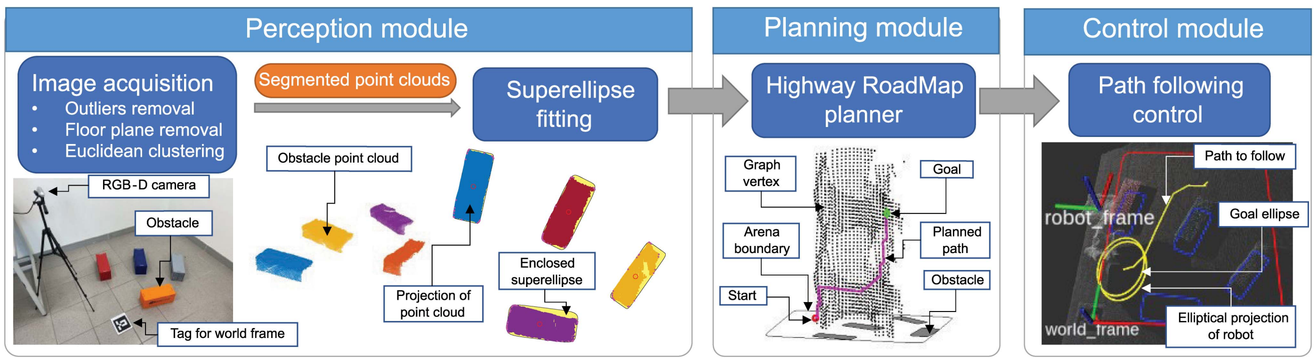


Fig. 13. Pipeline for physical experiments of the walking path planning for NAO humanoid robot.

convex polyhedra, all the random vertices are flipped around the origin, followed by computing the convex hull. Among all the 100 trials, the mean of κ_{volume} and κ_{implicit} are 11.74% and 0.3886 respectively, for the centrally symmetric polyhedra, and 19.88% and 0.6057, respectively, for the random polyhedra. The results show that the superquadric surfaces fit closely to the polyhedral vertices when the object is centrally symmetric. However, when the convex polyhedron is highly nonsymmetric, fitting the central-symmetric superquadric model is conservative and the volume difference might be unavoidably large. On the other hand, a superquadric body can be considered as the ground truth, which this article mainly addresses and uses for benchmarks with sampling-based planners. The fitting process is introduced when selecting parameters for sampling-based planners in Section VI-B2. It can be seen that a good convex polyhedral approximation uses many more sampled points. This is mainly because a better faceted representation of the curved surface of a superquadric requires denser set of sampled points.

B. *Parameter Selection for HRM and Prob-HRM*

Two major parameters that affect the success and performance of the proposed algorithms are the number of C-slices (N_{slice}) and sweep lines (N_{line}).

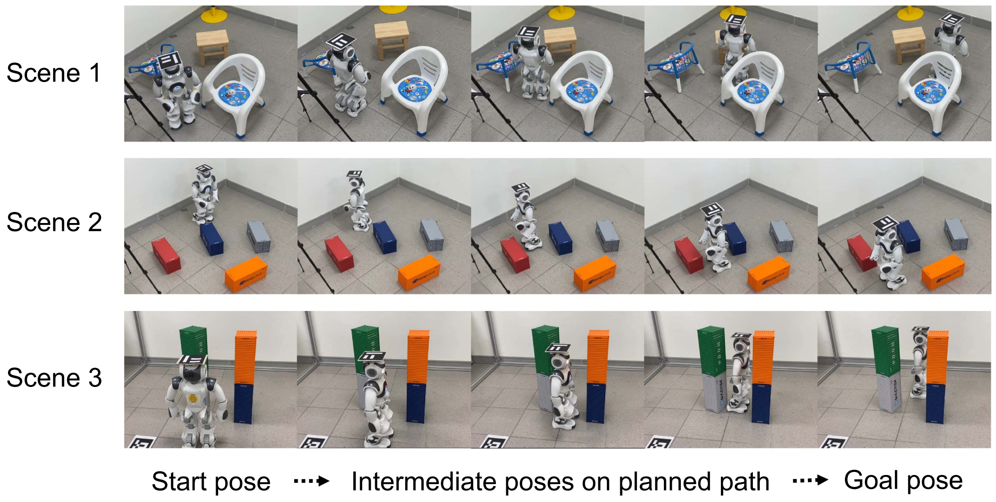


Fig. 14. Walking sequences of NAO that follows the planned $\mathrm{SE}(2)$ paths in physical experiments.

For the HRM planner, a predefined deterministic sampling of the orientation is required, which is discussed in Section IV-B. For the Prob-HRM planner, N_{slice} is incremental during the planning, meaning that the user does not need to provide this parameter beforehand. When there is no path found, this number will keep increasing until the termination condition is satisfied. Therefore, in this case, the roadmap can be kept refining instead of being computed from scratch again. Also, one could store the roadmap after one trial of planning for further reuse and refinements.

For both the HRM and Prob-HRM planners, the selection of N_{line} can be either user defined or computed based on (12). The latter choice is the default of the proposed planners. This choice initially generates a coarse resolution of sweep lines that can efficiently solve an easy problem, but tries to detect most of the C-obstacles. Since there is a refinement step for the existing roadmap, the input N_{line} only defines an initial resolution of the roadmap. When the problem becomes more complex, e.g., including narrow passages, the existing roadmap will be made denser by iteratively doubling the number of sweep lines until one of the termination conditions is satisfied.

C. 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). Therefore, the computational complexity is linear with respect to only one body, not both ones as the traditional polytope-based Minkowski sum [26]. Moreover, the numerical errors introduced in this process only come from the geometric approximations of object since the Minkowski sum computations are exact. The density of sampling vertices on the object surface is determined and used throughout all the experiments in this article. It is shown to be robust in different scenarios in terms of success rate and speed to solve motion planning queries.

The sweep line method in a single C-slice avoids traditional collision detection computation in generating collision-free samples. The vertices computed in each C-slice are automatically guaranteed to be safe. With the enhancement step, more vertices within each free segment can be generated. The added new vertices are closer to the adjacent free segment than the existing middle point, making it possible to circumvent obstacles compared to directly connecting two middle points. This step makes the vertex generation process more robust since more possible valid edges can be connected. Moreover, when connecting an edge between two vertices within one C-slice, 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.

With the roadmap refinement process, the portion of free space represented by the collision-free intervals of each sweep line increases with higher resolutions. Each C-slice can be explored uniformly along the sweeping direction and completely within a certain resolution parameter. The initial resolution parameter set by users might not be enough to find a valid solution. However, with this refinement step, the free space can be explored in an adaptive way, making the proposed algorithms more robust in dealing with resolution errors.

The “bridge C-slice” adds another C-slice to the whole roadmap, which doubles the total number of slices. However, it simplifies the edge validation process. In each bridge C-slice, only the center point of each enlarged ellipsoidal void for robot part is checked with C-obstacles. Computing a path in the bridge C-slice can be viewed as a projection of the $\mathrm{SE}(3)$ (or $\mathrm{SE}(3) \times (S^1)^n$) motion sequence of the robot onto a path for translational motion of the enlarged void in $\mathbb{R}^3$. The validation process still involves interpolations between two $\mathrm{SO}(3)$ (or $\mathrm{SO}(3) \times (S^1)^n$) configurations. However, they are only computed once before connecting two C-slices.

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

D. Limitations

There are also some limitations of the proposed framework. First, the geometry of the robot parts is limited to ellipsoids. The Minkowski sums are exact only when one of the bodies is an ellipsoid. For other geometric representations, such as polyhedra and point cloud, a fitting process is required before running the planner. Also, the meshed surface of the exact Minkowski sum in the sweep-line process introduces another level of approximation errors.

The computation of TFE in the bridge C-slice is conservative in the sense that some free space will be lost when the robot parts are enlarged. This enlargement scarifies the completeness of the planner. However, the efficiency using the bridge C-slice is significant based on the ablation study and benchmark results. Also, when the distance between two C-slices is smaller, the TFE encloses each robot part tighter, resulting in losing less free space.

The current HRM and Prob-HRM are both effective when the robot motions are dominated by translations. However, 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. In this case, useful operations within a single C-slice might be very limited, since no translational connections can be made. 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-slice" method can be applied as the transition validity checker between adjacent C-slices.

IX. Conclusion

This article proposed 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 HRM planner. Collision-free configurations were 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 were connected through a novel "bridge C-slice" method using the sweep volume of enlarged ellipsoidal voids. A new hybrid probabilistic variant, i.e., Prob-HRM, was then proposed to solve higher dimensional problems. It combined 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.

Acknowledgment

The authors would like to thank Dr. Yan Yan, Dr. Yuanfeng Han, and Chan Wee Kiat for useful discussions and helps on the physical experiments. The ideas expressed in this article 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 characterization of Minkowski operations," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2018, pp. 3–18.

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

[3] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, Tech. Rep. TR 98-11, 1998.

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

[5] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, vol. 1, pp. 521–528.

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

[7] K. Shi, J. Denny, and N. M. Amato, "Spark PRM: Using RRTs within PRMs to efficiently explore narrow passages," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 4659–4666.

[8] T. Lai, P. Morere, F. Ramos, and G. Francis, "Bayesian local sampling-based planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1954–1961, Apr. 2020.

[9] Y. Yan, Q. Ma, and G. S. Chirikjian, "Path planning based on closed-form characterization of collision-free configuration-spaces for ellipsoidal bodies obstacles and environments," in *Proc. 1st Int. Workshop Robot Learn. Plan.*, 2016, pp. 13–19.

[10] A. Best, S. Narang, and D. Manocha, "Real-time reciprocal collision avoidance with elliptical agents," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 298–305.

[11] B. Shiffman, S. Lyu, and G. Chirikjian, "Mathematical aspects of molecular replacement. V. Isolating feasible regions in motion spaces," *Acta Crystallographica A: Found. Adv.*, vol. 76, pp. 145–162, 2020.

[12] A. H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Comput. Graph. Appl.*, vol. 1, no. 1, pp. 11–23, Jan. 1981.

[13] J.-C. Latombe, *Robot Motion Planning*, vol. 124. Berlin, Germany: Springer, 2012.

[14] S. Nelaturi and V. Shapiro, "Configuration products and quotients in geometric modeling," *Comput.-Aided Des.*, vol. 43, no. 7, pp. 781–794, 2011.

[15] J.-M. Lien, "Hybrid motion planning using Minkowski sums," in *Proc. Robot.: Sci. Syst. Conf.*, 2008, pp. 97–104.

[16] H.-Y. Yeh, S. Thomas, D. Eppstein, and N. M. Amato, "UOBPRM: A uniformly distributed obstacle-based PRM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 2655–2662.

[17] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1999, vol. 2, pp. 1024–1031.

[18] H.-Y. C. Yeh, J. Denny, A. Lindsey, S. Thomas, and N. M. Amato, "UMAPRM: Uniformly sampling the medial axis," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 5798–5803.

[19] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.

[20] J. Denny and N. M. Amato, "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2013, pp. 297–312.

[21] J. Lee, O. Kwon, L. Zhang, and S.-E. Yoon, “A selective retraction-based RRT planner for various environments,” *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 1002–1011, Aug. 2014.

[22] W. Wang, L. Zuo, and X. Xu, “A learning-based multi-RRT approach for robot path planning in narrow passages,” *J. Intell. Robot. Syst.*, vol. 90, nos. 1/2, pp. 81–100, 2018.

[23] X. Qin and N. T. An, “Smoothing algorithms for computing the projection onto a Minkowski sum of convex sets,” *Comput. Optim. Appl.*, vol. 74, no. 3, pp. 821–850, 2019.

[24] A. Halder, “Smallest ellipsoid containing p-sum of ellipsoids with application to reachability analysis,” *IEEE Trans. Autom. Control*, vol. 66, no. 6, pp. 2512–2525, Jun. 2021.

[25] P. Hachenberger, “Exact Minkowksi sums of polyhedra and exact and efficient decomposition of polyhedra into convex pieces,” *Algorithmica*, vol. 55, no. 2, pp. 329–345, 2009.

[26] E. Fogel, D. Halperin, and C. Weibel, “On the exact maximum complexity of Minkowski sums of polytopes,” *Discrete Comput. Geometry*, vol. 42, no. 4, 2009, Art. no. 654.

[27] G. S. Chirikjian and A. B. Kyatkin, *Harmonic Anal. for Eng. and Appl. Scientists: Updated and Expanded Ed.* New York, NY, USA: Courier Dover, 2016.

[28] J.-M. Lien, “A simple method for computing Minkowski sum boundary in 3D using collision detection,” in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2009, pp. 401–415.

[29] A. Baram, E. Fogel, D. Halperin, M. Hemmer, and S. Morr, “Exact Minkowski sums of polygons with holes,” *Comput. Geometry*, vol. 73, pp. 46–56, 2018.

[30] H. Barki, F. Denis, and F. Dupont, “Contributing vertices-based Minkowski sum computation of convex polyhedra,” *Comput.-Aided Des.*, vol. 41, no. 7, pp. 525–538, 2009.

[31] J.-M. Lien, “Covering Minkowski sum boundary using points with applications,” *Comput. Aided Geometric Des.*, vol. 25, no. 8, pp. 652–666, 2008.

[32] Y. Yan and G. S. Chirikjian, “Closed-form characterization of the Minkowski sum and difference of two ellipsoids,” *Geometriae Dedicata*, vol. 177, no. 1, pp. 103–128, 2015.

[33] A. Halder, “On the parameterized computation of minimum volume outer ellipsoid of Minkowski sum of ellipsoids,” in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 4040–4045.

[34] G. Vezzani, U. Pattacini, and L. Natale, “A grasping approach based on superquadric models,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 1579–1586.

[35] M. P. Polverini, A. Laurenzi, E. M. Hoffman, F. Ruscelli, and N. G. Tsagarakis, “Multi-contact heavy object pushing with a centaur-type humanoid robot: Planning and control for a real demonstrator,” *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 859–866, Apr. 2020.

[36] S. B. Pope, “Algorithms for ellipsoids,” Sibley School Mech. Aerosp. Eng., Cornell Univ., Ithaca, NY, USA, Rep. FDA 08-01, 2008.

[37] A. A. Kurzhanskiy and P. Varaiya, “Ellipsoidal toolbox (ET),” in *Proc. IEEE Conf. Decis. Control*, 2006, pp. 1498–1503.

[38] L. G. Khachiyan, “Rounding of polytopes in the real number model of computation,” *Math. Oper. Res.*, vol. 21, no. 2, pp. 307–320, 1996.

[39] W. Wang, J. Wang, and M.-S. Kim, “An algebraic condition for the separation of two ellipsoids,” *Comput.-Aided Geometric Des.*, vol. 18, no. 6, pp. 531–539, 2001.

[40] X. Jia, Y.-K. Choi, B. Mourrain, and W. Wang, “An algebraic approach to continuous collision detection for ellipsoids,” *Comput.-Aided Geometric Des.*, vol. 28, no. 3, pp. 164–176, 2011.

[41] E. Rimon and S. P. Boyd, “Obstacle collision detection using best ellipsoid fit,” *J. Intell. Robot. Syst.*, vol. 18, no. 2, pp. 105–126, 1997.

[42] S. Iwata, Y. Nakatsukasa, and A. Takeda, “Computing the signed distance between overlapping ellipsoids,” *SIAM J. Optim.*, vol. 25, no. 4, pp. 2359–2384, 2015.

[43] S. Ruan, J. Ding, Q. Ma, and G. S. Chirikjian, “The kinematics of containment for N-dimensional ellipsoids,” *J. Mech. Robot.*, vol. 11, no. 4, 2019, Art. no. 041005.

[44] N. Vaskevicius and A. Birk, “Revisiting superquadric fitting: A numerically stable formulation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 220–233, Jan. 2019.

[45] D. Paschalidou, A. O. Ulusoy, and A. Geiger, “Superquadrics revisited: Learning 3D shape parsing beyond cuboids,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10344–10353.

[46] D. Peng and K. J. Hanley, “Contact detection between convex polyhedra and superquadrics in discrete element codes,” *Powder Technol.*, vol. 356, pp. 11–20, 2019.

[47] S. Ruan, K. L. Poblete, Y. Li, Q. Lin, Q. Ma, and G. S. Chirikjian, “Efficient exact collision detection between ellipsoids and superquadrics via closed-form Minkowski sums,” in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 1765–1771.

[48] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Berlin, Germany: Springer-Verlag, 2008.

[49] F. Solina and R. Bajcsy, “Recovery of parametric models from range images: The case for superquadrics with global deformations,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 2, pp. 131–147, Feb. 1990.

[50] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[51] C. Wülker, S. Ruan, and G. S. Chirikjian, “Quantizing Euclidean motions via double-coset decomposition,” *Research*, vol. 2019, 2019, Art. no. 1608396.

[52] K. Shoemake, “Uniform random rotations,” in *Graphics Gems III (IBM Version)*. Amsterdam, The Netherland: Elsevier, 1992, pp. 124–132.

[53] A. Yershova, S. Jain, S. M. Lavalle, and J. C. Mitchell, “Generating uniform incremental grids on SO(3) using the Hopf fibration,” *Int. J. Robot. Res.*, vol. 29, no. 7, pp. 801–812, 2010.

[54] Y. Yan, “Geometric motion planning methods for robotics and biological crystallography,” Ph.D. dissertation, Dept. Mech. Eng., Johns Hopkins Univ., Baltimore, MD, USA, 2014.

[55] D. Hsu, J.-C. Latombe, and R. Motwani, “Path planning in expansive configuration spaces,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 1997, vol. 3, pp. 2719–2726.

[56] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, “An obstacle-based rapidly-exploring random tree,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 895–900.

[57] V. Boor, M. H. Overmars, and A. F. Van Der Stappen, “The Gaussian sampling strategy for probabilistic roadmap planners,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 1999, vol. 2, pp. 1018–1023.

[58] J. Pan, S. Chitta, and D. Manocha, “FCL: A general purpose library for collision and proximity queries,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 3859–3866.

[59] “NAO the humanoid and programmable robot.” Accessed: Apr. 3, 2021. [Online]. Available: https://www.softbankrobotics.com/emea/en/nao

[60] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognit.*, vol. 47, no. 6, pp. 2280–2292, 2014.

[61] R. B. Rusu and S. Cousins, “3D is here: Point cloud library (PCL),” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 1–4.

[62] Y. Han, R. Li, and G. S. Chirikjian, “Can I lift it? Humanoid robot reasoning about the feasibility of lifting a heavy box with unknown physical properties,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 3877–3883.

Sipu Ruan (Member, IEEE) received the bachelor’s degree in mechanical engineering from the Harbin Institute of Technology, Harbin, China, in 2015, and the master’s degree in robotics and the Ph.D. degree in mechanical engineering from Johns Hopkins University, Baltimore, MD, USA, in 2017 and 2020, respectively.

In January 2021, he became a Research Fellow with the Department of Mechanical Engineering, National University of Singapore, Singapore. His research interests include robot motion planning, computational kinematics, and geometry.

Karen L. Poblete received the B.S. degree in computer and telematics engineering and the M.S. degree in computer science from the Instituto Tecnológico Autónomo de México, Mexico City, Mexico, in 2015 and 2017, respectively, and the M.S. degree in computer science from Johns Hopkins University, Baltimore, MD, USA, in 2020.

She is currently a Software Developer with Epic Systems, Verona, WI, USA. Her main research interests include robot motion planning.

Hongtao Wu (Graduate Student Member, IEE E) received the B.Eng. degree in power engineering from Beihang University, Beijing, China, in 2017, and the M.S.E. degree in robotics from Johns Hopkins University, Baltimore, MD, USA, in 2019.

He is currently with the Laboratory for Computational Sensing and Robotics, Johns Hopkins University. His research interests include robot imagination, object affordances, and robot learning. Specifically, he is interested in developing algorithms to enhance robot interactions with daily objects via robot imagination.

Mr. Wu was the Finalist for the Best Paper Award in Human–Robot Interaction at 2021 IEEE International Conference on Robotics and Automation.

Qianli Ma received the B.S. degree in mechanical engineering from Tianjin University, Tianjin, China, in 2012, and the M.S. degree in robotics and the Ph.D. degree in mechanical engineering from Johns Hopkins University, Baltimore, MD, USA, in 2014 and 2017, respectively.

He is currently working on motion planning and control with Motional, Inc., Pittsburgh, PA, USA, a company that specializes in autonomous driving.

Gregory S. Chirikjian (Fellow, IEEE) received the BSE degree in engineering mechanics and MSE degree in mechanical engineering from Johns Hopkins University, Baltimore, MD, USA, in 1988, and the Ph.D. degree in applied mechanics from the California Institute of Technology, Pasadena, CA, USA, in 1992.

From 1992 to 2021, he was on the Faculty of the Department of Mechanical Engineering, Johns Hopkins University, where he became a Full Professor in 2001 and was a Department Chair from 2004 to 2007. In January 2019, he joined the National University of Singapore, Singapore, where he is currently the Head of the Department of Mechanical Engineering. He is the author of more than 250 journal and conference papers and the primary author of three books, including *Engineering Applications of Noncommutative Harmonic Analysis* (Evanston, IL, USA: Routledge, 2001) and *Stochastic Models, Information Theory, and Lie Groups* (vols. 1/2. New York, NY, USA: Springer, 2009 and 2011). In 2016, an expanded edition of his 2001 book was published as a Dover book under a new title *Harmonic Analysis for Engineers and Applied Scientists*. His research interests include robotics, applications of group theory in a variety of engineering disciplines, applied mathematics, and the mechanics of biological macromolecules.

Dr. Chirikjian was a 1993 National Science Foundation Young Investigator, a 1994 Presidential Faculty Fellow, and a 1996 recipient of the American Society of Mechanical Engineers (ASME) Pi Tau Sigma Gold Medal. In 2008, he became a Fellow of the ASME. From 2014 to 2015, he was a Program Director of the U.S. National Robotics Initiative, which included responsibilities in the Robust Intelligence cluster in the Information and Intelligent Systems Division of Computer and Information Science and Engineering at the National Science Foundation.