Path Planning for Elliptical Agents in SE(2) via Generalized Closed-form Minkowski Operations

Sipu Ruan, Karen P. Rodriguez, Qianli Ma, Gregory S. Chirikjian

July 20, 2018

1 Contributions

Key contributions are listed as follows:

1. Derived the closed-form Minkowski operations between ellipsoid and any parametric surfaces. Explicitly computed the expression for 2D ellipse-superellipse case.

2. Modified the general Highway RoadMap algorithm:
(1) Add vertex at each sweep line to ensure the edges are within a convex cell;
(2) Enclose the curved C-obstacles boundaries by straight line segments, so that all the valid cells are convex;
(3) Applied the concept of the Kinematics of Containment to connect vertexes among different C-layers. Provided a procedure for layer connections: add a middle vertex at the intersection between two local C-spaces, and connect the middle vertex with the original ones.

3. Implemented in C++ and compared with sampled-based algorithms in OMPL.

2 Mathematical Preliminaries

This section provides the mathematical preliminaries for the path planning algorithm. Closed-form Minkowski sum and difference between ellipsoid and any surface with implicit and parametric expressions is derived at first. As a concrete environmental representations in this paper, the surface in the form of a superellipse is studied. And the closed-form Minkowski sum and difference between an ellipse and a superellipse is derived explicitly.

2.1 Minkowski Sum and Difference between Two Convex Objects

The Minkowski sum of two convex point sets (or bodies) each centered at the origin, P_1 and P_2 in $\mathbb{R}^n$, is defined as

$$P_1 \oplus P_2 \doteq \{p_1 + p_2 \mid p_1 \in P_1, p_2 \in P_2\}. \tag{1}$$

The Minkowski difference between P_1 and P_2 is defined as [1]

$$P_1 \ominus P_2 \doteq \bigcap_{p_2 \in P_2} (P_1 + p_2), \tag{2}$$

Alternatively, the Minkowski difference of two convex bodies can be defined relative to the Minkowski sum as the body $P_1' = P_1 \ominus P_2$ for which $P_1 = P_1' \oplus P_2$.

Minkowski operations are used in a wide range of applications such as robot motion planning [2], CAD/CAM, assembly planning [3] and computer-aided design [4]. For example, consider an obstacle P_1 and a robot P_2 that moves by translation. By choosing a reference point attached to P_2, then $P_1 \oplus P_2$ is the locus of positions of the reference point where $P_1 \cap P_2 \neq \emptyset$. In the study of motion planning this sum is called a C-space obstacle. Alternatively, if P_1 is an "arena" (i.e., a bounded workspace) in which the robot P_2 is moving, then $P_1 \ominus P_2$ is the locus of positions of the reference point where $P_1 \cap P_2 = P_2$, and represents the robot's collision-free C-space with respect to the arena. While defining the Minkowski operations mathematically is easy, computing useful representations of Minkowski sums or differences can be difficult and computationally expensive, especially when the exact boundary of the geometry of these entries need to be represented explicitly. Many algorithms exist for numerically computing the boundary of the Minkowski sum of polygonal/polyhedral sets in two/three dimensions [4–9]. These algorithms mainly either compute the convolution of geometric boundaries [5], or use polygon/polyhedra decompositions [4,6–8]. Minkowski sums of curved regions/surfaces have also been studied (e.g. [10–13]).

2.2 Generalized Closed-form Minkowski Operations between an Ellipsoid and Any Surface Embedded in N-dimensional Euclidean Space

In previous work, the closed-form Minkowski operations between two ellipsoids are proposed. It is further observed that such operations can be extended when one ellipsoid is substituted by any surface with implicit and explicit expressions. The procedure is introduced as follows.

Assume S_1 is a surface embedded in N-dimensional Euclidean space, it can be expressed implicitly as a constraint function

$$\Phi(\mathbf{x}) = 1, \tag{3}$$

and parametrically as

$$\mathbf{x} = \mathbf{f}(\phi), \tag{4}$$

where both $\mathbf{x}_1$ and $\mathbf{f}$ are vector-valued functions with parameters ϕ. And let E_2 be an arbitrary ellipsoid in $\mathbb{R}^n$, with semi-axis lengths given by $\mathbf{a}_2 = [a_1, a_2, ..., a_n]^\top$. Then, the implicit and explicit equations are of the form

$$\mathbf{x}^\top A_2^{-2}\mathbf{x} = 1 \text{ and } \quad \mathbf{x} = A_2\mathbf{u}(\psi), \tag{5}$$

where $A_2 = R_2\Lambda(\mathbf{a}_2)R_2^\top$ is the shape matrix of E_2 with $R_2 \in \mathrm{SO(n)}$ denoting the orientation of the ellipsoid, and $\Lambda(\cdot)$ being a diagonal matrix. Here $\mathbf{u}(\psi)$ is the standard parameterization of the hyper-sphere with parameters $\psi = [\psi_1, \psi_2, ..., \psi_{n-1}]^\top$.

The idea of the closed-form Minkowski operations is based on defining an affine transformation to "shrink" the ellipsoid E_2 into a sphere E_2' of radius r and constructing an offset surface for the transformed surface S_1'. For the sake of simplicity, the following derivations only focus on the computations of Minkowski sum.

The affine transformations that shrink the ellipsoid into a sphere with radius $r = \min\{a_1, a_2, ..., a_n\}$ on the surface S_1 can be expressed as

$$\mathbf{x}' = R_2\Lambda(r/\mathbf{a}_2)R_2^\top\mathbf{x} \doteq T\mathbf{x}, \tag{6}$$

where $T = R_2\Lambda(r/\mathbf{a}_2)R_2^\top$ denotes the "shrinking" affine transformation, and is symmetric and positive definite due to the fact that $\Lambda(r/\mathbf{a}_2)$ is diagonal and positive definite.

The implicit expression for the boundary of the "shrunk" S_1, denoted as E_1' is

$$\Phi(T^{-1}\mathbf{x}') = 1. \tag{7}$$

Then the Minkowski sum between S_1' and E_2', which now is a sphere, can be obtained by computing the boundary of the offset surface with offset radius r as

$$\mathbf{x}_{ofs} = \mathbf{x}' + r\mathbf{n}', \tag{8}$$

where $\mathbf{n}' = \frac{\nabla\Phi(T^{-1}\mathbf{x}')}{\|\nabla\Phi(T^{-1}\mathbf{x}')\|}$ is the outward normal of the surface and $\nabla\Phi(T^{-1}\mathbf{x}') = T^{-\top}\nabla\Phi(\mathbf{x})$ with $T^{-\top} = (T^{-1})^\top = (T^\top)^{-1} = T^{-1}$.

The Minkowski sum between the original surface S_1 and ellipsoid E_2 can be given by "stretching" the transformed space back, using inverse affine transformation, as

$$\mathbf{x}_{eb} = T^{-1}\mathbf{x}_{ofs} = T^{-1}(\mathbf{x}' + r\mathbf{n}') = T^{-1}(T\mathbf{x} + r\frac{T^{-\top}\nabla\Phi(\mathbf{x})}{\|T^{-\top}\nabla\Phi(\mathbf{x})\|}) = \mathbf{x} + r\frac{T^{-2}\nabla\Phi(\mathbf{x})}{\|T^{-1}\nabla\Phi(\mathbf{x})\|} \tag{9}$$

The Minkowski difference $S_1 \ominus E_2$ therefore can be obtained by switching the plus sign in Eq. (9) to minus. However, for the Minkowski difference, not all points on the offset surface are valid. In this case, a "curvature constraint" should be satisfied: after the "shrinking" operation, the curvature of every point on the transformed surface S_1' should be smaller than the curvature of the transformed ellipsoid E_2'.

2.3 Explicit Expressions of the Closed-form Minkowski Operations between an Ellipse and a Superellipse

We now give a concrete example for the closed-form Minkowski operations when the surface is a superquadric. This formulates a mathematical representations for the implementations in this paper. The calculations in both 2-dimensional (ellipse-superellipse) and 3-dimensional (ellipsoid-superquadric) cases are provided.

2.3.1 2D Case: Ellipse-Superellipse

The implicit equation for a superellipse SQ_1 in a 2-dimensional Euclidean space is defined as:

$$\Phi(\mathbf{x}) = \left(\frac{x_1}{a_1}\right)^{\frac{2}{\epsilon}} + \left(\frac{y_1}{b_1}\right)^{\frac{2}{\epsilon}} = 1 \tag{10}$$

And the corresponding explicit equation can then be written as:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} a_1\cos^\epsilon\theta \\ b_1\sin^\epsilon\theta \end{pmatrix}, \quad -\pi \le \theta \le \pi \tag{11}$$

The exponentiation with ϵ is a signed power function such that $\cos^{\epsilon}\theta = \text{sign}(\cos\theta)|\cos\theta|^{\epsilon}$. The shape described by the above function can change with ϵ, and we only consider the case of $0 < \epsilon < 2$ so that the corresponding shape will always be convex.

With the above knowledge of superellipse, we define a superelliptic surface as:

$$\Phi(x_1, y_1) = \left(\frac{x_1}{a_1}\right)^{\frac{2}{\epsilon}} + \left(\frac{y_1}{b_1}\right)^{\frac{2}{\epsilon}} = 1 \tag{12}$$

Then the normal vector of the superellipse can be obtained by calculating the gradient of $\Phi(x_1, y_1)$:

$$\nabla\Phi(x_1, y_1) = \left[\frac{2}{a_1\epsilon}\left(\frac{x_1}{a_1}\right)^{\frac{2}{\epsilon}-1}, \frac{2}{b_1\epsilon}\left(\frac{y_1}{b_1}\right)^{\frac{2}{\epsilon}-1}\right]^{\top}. \tag{13}$$

Substituting Eq. (11) into Eq. (13) and normalizing the obtained parametric gradient gives a unit normal vector of the superellipse:

$$\nabla\Phi(x_1(\theta), y_1(\theta)) = \frac{2}{\epsilon}\begin{pmatrix} \cos^{2-\epsilon}\theta/a_1 \\ \sin^{2-\epsilon}\theta/b_1 \end{pmatrix}, \tag{14}$$

where $0 < \epsilon < 2$.

Let the ellipse be defined by the parameters $\mathbf{a}_2 = [a_2, b_2]^{\top}$ and its orientations be characterized as $R_2 = R(\theta) \in \text{SO}(3)$, if we further define $r = \min\{a_2, b_2\}$, then the "shrinking" transformation can be computed as $T = R_2\Lambda(r/\mathbf{a}_2)R_2^{\top}$.

Now we have all the information to calculate the closed-form Minkowski sum and difference between an ellipse and a superellipse, i.e. $SQ_1 \oplus E_2$ and $SQ_1 \ominus E_2$ respectively. Figs. 1 and 2 illustrate the computational process of the closed-form Minkowski sum and difference between a superellipse and an ellipse.

2.3.2 3D Case: Ellipsoid-Superquadrics

We now briefly show that the same closed-form Minkowski operations can be extended to the 3D case. The implicit equation for superquadrics is:

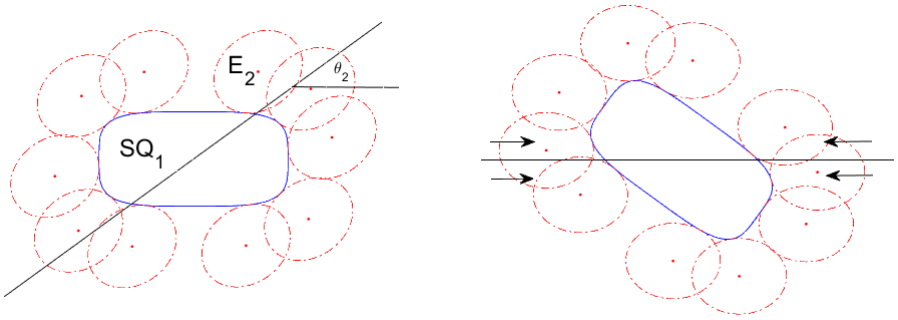
$$\Phi(\mathbf{x}) = \left(\left(\frac{x}{a}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{b}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{c}\right)^{\frac{2}{\epsilon_1}} = 1. \tag{15}$$

The corresponding explicit equation is:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a\cos^{\epsilon_1}\eta\cos^{\epsilon_2}\omega \\ b\cos^{\epsilon_1}\eta\sin^{\epsilon_2}\omega \\ c\sin^{\epsilon_1}\eta \end{pmatrix}, \quad \begin{matrix} -\pi/2 \le \eta \le \pi/2 \\ -\pi \le \omega < \pi \end{matrix} \tag{16}$$

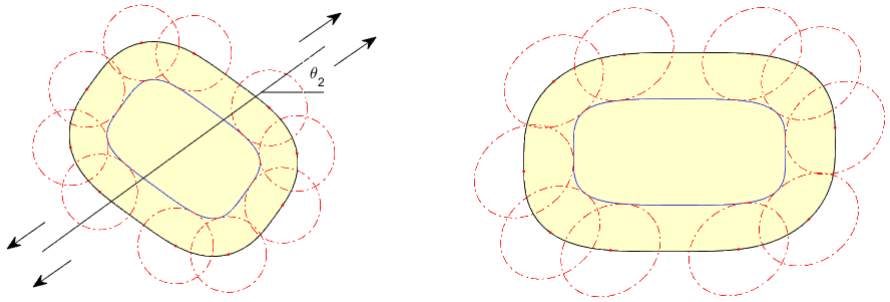
The normal vector a point $\mathbf{x}(\eta, \omega)$ on the superellipsoid surface is defined as:

$$\begin{aligned} &\mathbb{N}_1(\eta, \omega) = \mathbf{r}_{\eta}(\eta, \omega) \times \mathbf{r}_{\omega}(\eta, \omega) \\ =&\begin{pmatrix} -bc\epsilon_1\epsilon_2\sin^{\epsilon_1-1}\eta\cos^{\epsilon_1+1}\eta\cos\omega\sin^{\epsilon_2-1}\omega \\ -ac\epsilon_1\epsilon_2\sin^{\epsilon_1-1}\eta\cos^{\epsilon_1+1}\eta\sin\omega\cos^{\epsilon_2-1}\omega \\ -ab\epsilon_1\epsilon_2\sin\eta\cos^{2\epsilon_1-1}\eta\sin^{\epsilon_2-1}\omega\cos^{\epsilon_2-1}\omega \end{pmatrix} \\ =&f(\eta, \omega)\mathbb{N}_2(\eta, \omega), \end{aligned} \tag{17}$$



(a) Both bodies are rotated clockwise with an angle θ_2.

(b) Both of the bodies are shrunk until ∂E_2 becomes a circle.



(c) The Minkowski sum in the shrunk space is an offset surface (the shaded area).

(d) Transform back to the original space. The shaded region represents $SQ_1 \oplus E_2$.

Figure 1: Algorithm for obtaining the characterizations of the Minkowski sum between a superellipse body SQ_1 and an ellipse body E_2.

where

$$\mathbf{r}_\eta(\eta,\omega) = \begin{pmatrix} -a\epsilon_1 \sin\eta \cos^{\epsilon_1-1}\eta \cos^{\epsilon_2}\omega \\ -b\epsilon_1 \sin\eta \cos^{\epsilon_1-1}\eta \sin^{\epsilon_2}\omega \\ c\epsilon_1 \sin^{\epsilon_1-1}\eta \cos\eta \end{pmatrix} \tag{18}$$

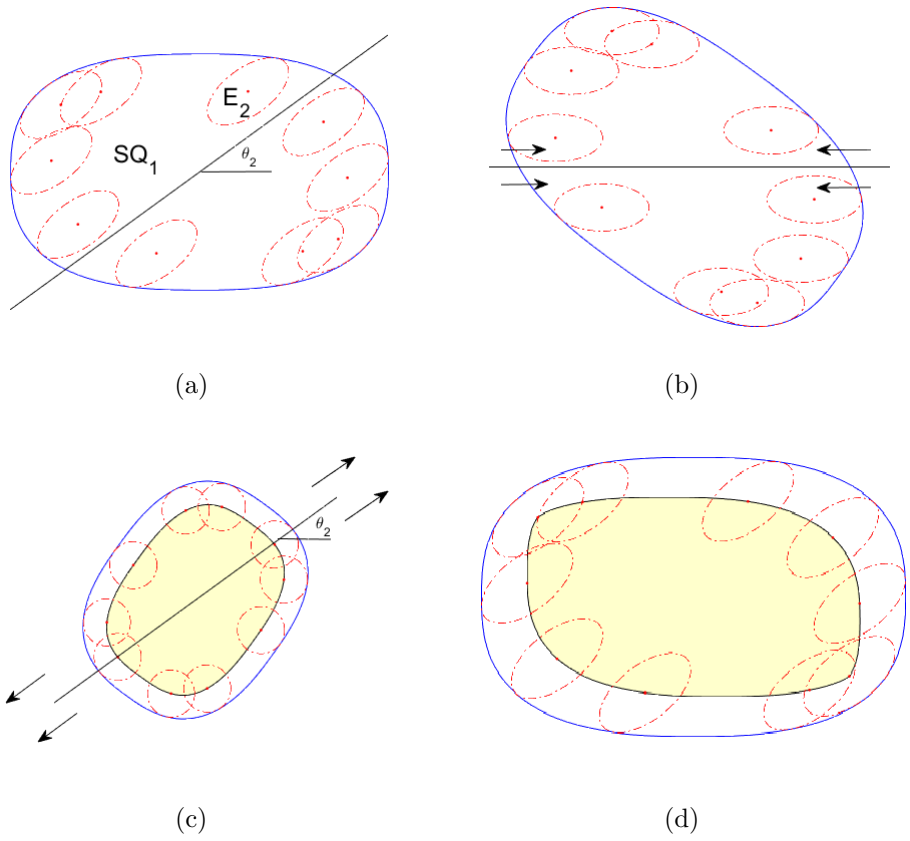
and

$$\mathbf{r}_\omega(\eta,\omega) = \begin{pmatrix} -a\epsilon_2 \cos^{\epsilon_1}\eta \sin\omega \cos^{\epsilon_2-1}\omega \\ b\epsilon_2 \sin\eta \cos^{\epsilon_1}\eta \cos\omega \sin^{\epsilon_2-1}\omega \\ 0 \end{pmatrix} \tag{19}$$

$\mathbf{r}_\eta(\eta,\omega)$ and $\mathbf{r}_\omega(\eta,\omega)$ are tangent vectors along η and ω respectively.

After extracting and dropping a common factor of $\mathbf{n}(\eta,\theta)$, an equivalent form of normal vector can be obtained as:

$$\nabla\Phi(x,y,z) = \mathbb{N}_2(\eta,\omega) = \begin{pmatrix} \cos^{2-\epsilon_1}\eta \cos^{2-\epsilon_2}\omega/a \\ \cos^{2-\epsilon_1}\eta \sin^{2-\epsilon_2}\omega/b \\ \sin^{2-\epsilon_2}\eta/c \end{pmatrix}. \tag{20}$$



(a) (b)

(c) (d)

Figure 2: Algorithm for the closed-form Minkowski difference of a superelliptical body SQ_1 and an elliptical body E_2. This presumes that the radius of curvature of the rotated and shrunk ellipse (E_2), which becomes a circle, is smaller than that of the bounding rotated and shrunk superellipse in anywhere. The algorithm follows the same steps as Figure 1. The shaded region represents $SQ_1 \ominus E_2$

For the case where $\epsilon_1 = \epsilon_2 = \epsilon$, Eq .(15) will become

$$\left(\frac{x}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\epsilon}} + \left(\frac{z}{c}\right)^{\frac{2}{\epsilon}} = 1, \tag{21}$$

the explicit equation is then:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a\cos^{\epsilon}\eta\cos^{\epsilon}\omega \\ b\cos^{\epsilon}\eta\sin^{\epsilon}\omega \\ c\sin^{\epsilon}\eta \end{pmatrix}, \quad \begin{matrix} -\pi/2 \le \eta \le \pi/2 \\ -\pi \le \omega < \pi \end{matrix} \tag{22}$$

If we further define a transformation $T = R_2\Lambda(r/\mathbf{a}_2)R_2^{\top}$, the offset surface of the Minkowski sum and difference follows the same fashion as in the 2D case.

3 The Highway RoadMap Path Planning Algorithm for Ellipsoidal Agents: Reviews and Extensions

With the knowledge of exact closed-form contact boundary of an ellipsoid and a superquadrics in Sec.2.3, path planning problems can be solved effectively. In this section, the "Highway RoadMap" algorithm [**?**] which takes advantage of the knowledge of collision-free C-space a priori is briefly reviewed and extended.

3.1 Overview of the Highway RoadMap Algorithm

The Highway RoadMap system is built based on the idea of cell decompositions at each orientation of the robot. At first, the orientations are discretized, which can be either uniformly or randomly sampled from the configuration space. Then, at each fixed orientation, a subset of the C-space that only contains translational motions is built, denoted as a "C-layer". The closed-form Minkowski sum and difference are computed between the robot and obstacles and arenas, respectively. Once the Minkowski operations are applied, the configuration space with C-obstacles is constructed at each C-layer, then the free space can be characterized and decomposed into disjoint collision-free cells. A subset of the roadmap can be constructed by detecting the middle point at each boundary of the adjacent cells as a vertex and connecting edges between two vertexes. The whole roadmap system can then be built by connecting vertexes among different C-layers. This procedure is based on the previously published work in [**?**] using two ellipsoids. In this paper, we extend the descriptions of the environment by using superellipses, modify some of the steps when building a roadmap at each C-layer, and propose a new method to connect vertexes among different C-layers. The overall algorithm is illustrated as

Algorithm 1: Highway RoadMap Algorithm

Input: *Robot*; *Obstacle*; *Arena*; *EndPoint*
Output: Graph; Path

```
Rot = Discretized Orientations;
foreach Rot do
    C_Obs, C_Arena ← Minkowski Operations between Robot and Obstacle, Arena;
    Cell_CF ← Cell Decomposition by Sweep-Line Process;
    Graph ← Vertex Generations and Edge Connections Within the C-Layer;
end
Graph ← Connect Closest Vertexes among Adjacent C-Layers;
Path ← Graph Search Process;
```

3.2 A Sweep-Line Process for the Cell Decomposition within One C-Layer

Within one C-layer, the motion of a robot is restricted to translation only, so the Minkowski operations are applied here to expand the boundary of obstacles and shrink the boundary of

arenas so that the robot can be represented as a point in the configuration space. This idea goes back to the traditional path planning algorithm, with the exception that the boundaries after the Minkowski operations are curved surfaces instead of polyhedra. So in this case, a process similar to the "sweep line" method to decompose the collision-free space as disjoint cells was proposed.

3.2.1 Overview of the Sweep-Line Process

Suppose that the robot is constructed by a finite union of N ellipsoids $E_1, E_2, \ldots, E_n$, and the rigid transformations between the first ellipsoid E_1 and other ellipsoids $E_2, E_3, \ldots, E_n$ are defined as $g_2, g_3, \ldots, g_n$ [1] respectively. Let the collision-free space (obtained by Minkowski operation) for each ellipsoid E_i be C_i $(i = 1, \ldots, n)$, then the collision-free space for the whole robot can be characterized as a union of them as viewed in the body frame of first ellipsoid:

$$C = C_1 \cap (g_2 \circ C_2) \cap (g_3 \circ C_3) \cap \cdots \cap (g_n \circ C_n) \tag{23}$$

where $g_i \circ C_j \doteq R_i C_j + p_i$ $(i, j = 1, \ldots, n)$

In order to detect those regions, a set of sweep lines parallel to x-axis are generated. Each sweep line intersects with all the curves, with the intersecting points saved as pairs or intervals. Denote the line segments within the obstacles as P_{O_i}, and those within the arenas as P_{A_i}. Then the collision-free line segment P_{CF} for each sweep line is represented as

$$P_{CF} = \bigcap_{i=1}^{n_A \times n} P_{A_i} - \bigcup_{j=1}^{n_O \times n} P_{O_i} \tag{24}$$

where n_A and n_O are numbers of superquadrics that represent arenas and obstacles respectively, and n is the number of ellipsoids that represent different parts of the robot. Fig. **??** demonstrates the constructing process of collision-free space for an elliptical robot translating in superquadric environment. Fig. 3 shows the decomposed C-space at one layer with collision-free cells highlighted by shades.

3.2.2 C-Obstacle Boundary Enclosing via an Offset Polyhedron

Collision-free line segments P_{CF}s decompose the space into difference cells, and each cell consists of two adjacent collision-free line segments and the boundary segments of the arena or obstacle. However, since the C-obstacles boundaries are curved surfaces, the cells connected with obstacles are partially concave. This makes the paths within those cells not always collision-free. As a modifications of the previous algorithm, we propose to enlarge the C-obstacles by enclosing the boundaries with polyhedra. The vertexes for each polyhedron are generated based on the intersecting points between the sweep lines and the boundary of the C-obstacle. As an example in 2D, suppose the intersecting points between two adjacent sweep lines with the C-obstacle are $P_1 = [x_1, y_1]^\top$ and $P_2 = [x_2, y_2]^\top$ respectively (without loss of generality, assume $y_1 < y_2$). Then between the two points, we can search for a point $P = [x, y]^\top$ $(y \in [y_1, y_2])$ on the boundary of the C-obstacle that is farthest to the line

[1] Each rigid body transformation g_i consists of a rotation R_i and a translation p_i, and is written as a homogeneous representation matrix.

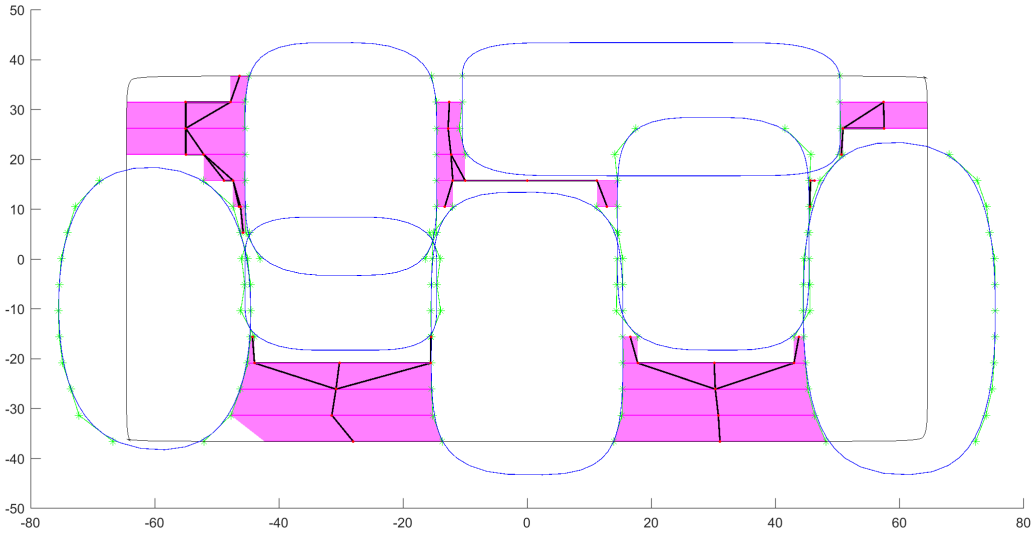


Figure 3: The cell decomposition at one C-layer.

segment $\overline{P_1P_2}$. The search is based on the distance from a point to a line segment which is determined by two end points as

$$d = \frac{\|(y_2 - y_1)x - (x_2 - x_1)y + x_2y_1 - y_2x_1\|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}. \tag{25}$$

Then the line segment $\overline{P_1P_2}$ is moved outwards from the C-obstacle, as $\overline{P_1'P_2'}$, by an offset distance

$$d_{off} = \frac{d}{\sin[\text{atan2}(y_2 - y_1, x_2 - x_1)]}. \tag{26}$$

The offset line segments are constructed based on two adjacent sweep lines, so for one sweep line, there will be two different points for the two offset line segments. Therefore, the point which is further to the C-obstacle is stored so that each offset line segment is either tangent to or separated with the C-obstacle.

Once the collision-free cells are constructed, the vertexes of the subgraph can be obtained as the mid-point of all the collision-free line segments. Then connecting vertexes on the same cell gives collision-free paths at each C-layer. Fig. 4 illustrates the idea of the Minkowski sum boundary enlargement in the 2D case.

3.3 A Local Planner for Vertex Connections among Different C-Layers

At each C-layer the robot can only translate, so it remains a question of how to connect the subgraphs among different C-layers and how the robot can transform between different layers by rotations. One way to do so is through the interpolation of the motions in SE(2) so that between each discrete vertexes, the changes of the configurations are almost infinitesimal

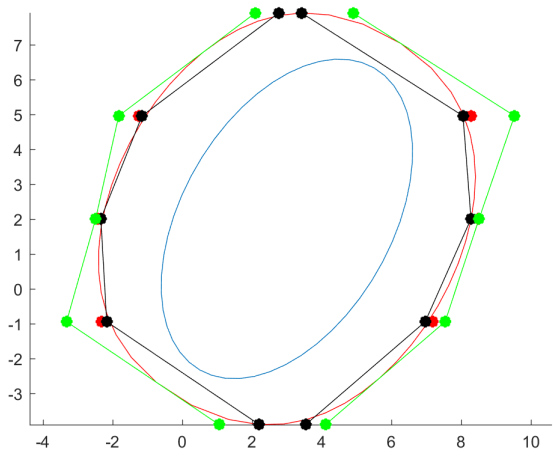


Figure 4: An illustration of the enlargement for the Minkowski sum boundary. The inner blue curve is the actual obstacle, and the outer red curve is the Minkowski sum boundary. The black dots and straight line segments are the original cell boundaries, while the green dots and line segments are the enlarged cell boundaries.

and connecting the adjacent configurations can be seen as "safe". However, it is always good to find a continuous collision-free space that can enclose all the steps along the edge between two vertexes, so we propose a local planner to check whether two close vertexes at the adjacent layers can be connected and build a valid path between them.

The basic idea is to encapsulate the actual robot with a slightly larger ellipsoid, i.e. inflate the actual robot by a factor of $\epsilon = 0.1$. Then the robot is allowed to move inside by a small amount of motions without collisions with the larger ellipsoid. Such motions can be described locally in the C-space, denoted by a "Local C-space". The local C-space becomes collision-free if the Minkowski operations are conducted using the larger ellipsoid. And the descriptions of the local C-space can be done before building the RoadMap a priori, since when the shape of the robot is fixed, the local C-space at each vertex is the same except for an offset between each other. Once the local C-space of the two vertexes intersect, a new vertex can be generated within the intersecting area and connected with the two vertexes respectively. The following subsections introduce in details about the characterizations of the local C-space and the procedure to connect two vertexes by a collision-free path.

3.3.1 The Characterizations of the Local C-space

The local C-space is characterized based on enclosing a slightly larger ellipsoid to the actual robot, which is a smaller ellipsoid in this case. So now we review the allowable motions of the smaller ellipsoid inside a larger ellipsoid, both of which can be described in N-dimensional. The related study traces back to the concept of the "Kinematics of Containment", which deals with any convex bodies that moved inside another. Recent study of a special case, which both bodies are N-dimensional ellipsoids, is exactly what can be applied here.

Given two N-dimensional ellipsoids E_a and E_b, and $E_a \subseteq E_b$, we let $\mathbf{a} = [a_1, a_2, ..., a_n]^T, \mathbf{b} = [b_1, b_2, ..., b_n]^T \in \mathbb{R}^n$ denote the semi-axes of E_a and E_b respectively. By substituting the ex-

plicit expression of the moving ellipsoid E_a into the implicit expression of the fixed ellipsoid E_b that is aligned with the world frame, the algebraic condition for E_a to move inside E_b without collision can be written as

$$(R_a\Lambda(\mathbf{a})\mathbf{u}+\mathbf{t}_a)^\top\Lambda^{-2}(\mathbf{b})(R_a\Lambda(\mathbf{a})\mathbf{u}+\mathbf{t}_a)\leq 1. \tag{27}$$

For this highly nonlinear expression, a small angle approximation can make it much simpler, where some better properties, such as convexity, can be proved. If the rotation of E_a is restricted, the rotation part calculated by exponential map can be approximated to the first order as

$$R_a=\exp(\widehat{\omega}_a)=\mathbb{I}+\frac{\sin\|\omega_a\|}{\|\omega_a\|}\widehat{\omega}_a+\frac{1-\cos\|\omega_a\|}{\|\omega_a\|^2}\widehat{\omega}_a^2 \tag{28a}$$

$$\approx\mathbb{I}+\widehat{\omega}_a. \tag{28b}$$

Substituting Eq. 28b into Eq. 27 and grouping parameters ($\mathbf{u}$) and variables (ω and $\mathbf{t}$) gives the first-order approximation of the left-hand side of the algebraic condition of containment as

$$C_{\mathbf{u}}(\xi)=\xi^\top H(\mathbf{u})\xi+\mathbf{h}^\top(\mathbf{u})\xi+c(\mathbf{u}), \tag{29}$$

where $H(\mathbf{u})\in\mathbb{R}^{n(n+1)/2\times n(n+1)/2}$, $\mathbf{h}(\mathbf{u})\in\mathbb{R}^{n(n+1)/2}$ and $c(\mathbf{u})\in\mathbb{R}$. The first order algebraic condition of containment is then defined as $C_{\mathbf{u}}(\xi)\leq 1$.

It can be further proved that Eq. (29) is a family of convex functions, with parameters $\mathbf{u}$ and unknown variables ξ. As a result, enclosing several valid configurations, which ensure the smaller ellipsoid is fully contained inside the larger one, by a convex hull gives a convex polyhedron in the C-space of the smaller ellipsoid. This convex polyhedron is a collision-free subspace and any path inside is guaranteed to be collision-free. The choice of those valid configurations that define the polyhedron can be random, or specifically computed as the extreme points on the axes of the configuration space. And the local C-space of each vertex in the roadmap can be characterized by such convex polyhedron. For the details of the proof and the construction of the convex polyhedron, please refer to [].

3.3.2 Vertex Connections Based on the Convex Polyhedron Local C-space

As stated at the beginning of this section, once the local C-space is defined, connecting two vertexes, V_1 and V_2, by a collision-free path $Path_{12}$ can be achieved by finding the intersection of the two local C-space. Concretely speaking, if the local C-space is characterized as a convex polyhedron, we first search a new vertex V_3 that is inside both polyhedron C-space of V_1 and V_2 by querying in both polyhedron. Then, connecting V_1, V_3 and V_2, V_3 by line segments gives $Path_{13}$ and $Path_{23}$ respectively. These two path segments are guaranteed to be collision-free since both are fully inside the convex polyhedron of V_1 or V_2. Finally, the desired collision-free path is a combination the two segments, i.e. $Path_{12}=Path_{13}\cup Path_{23}$. The idea is illustrated in Fig.

Note that, for the storage in terms of a graph structure, each new vertex is stored at the end of the "Vertex" list, and the new edges between the new vertex and the existing vertexes are added as a pair in the "Edge" list. Fig.5 demonstrates the proposed connection scheme for vertexes in difference C-layers based on the ideas of a convex lower bound of the allowable motions of one ellipsoid being fully contained in another.

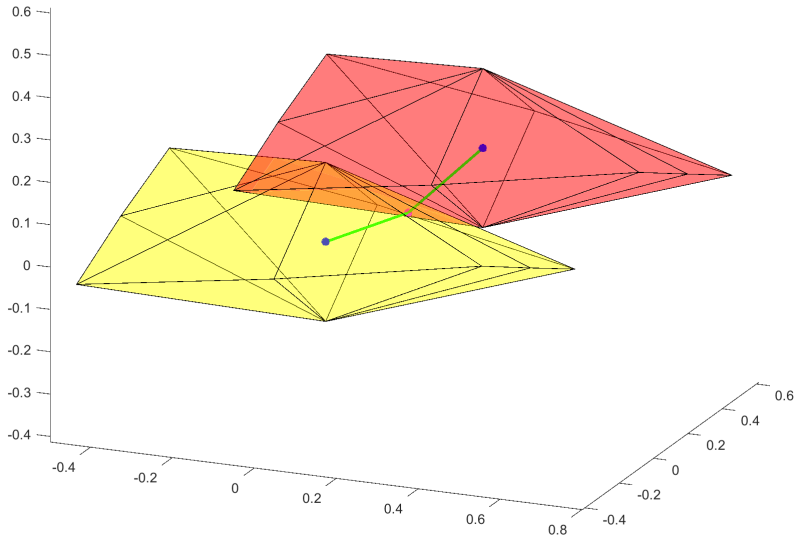


Figure 5: A collision-free connection scheme for vertexes in difference C-layers. The blue dots are the two vertexes, with two convex polyhedra being their local C-space. The green line segments connects the new middle vertex at the intersection area and the two vertexes respectively.

4 Experiments on Path Planning for an Elliptical Agent in SE(2)

As an initial demonstration of the extended Highway RoadMap algorithm, we consider a rigid body planning problem in SE(2), where the agent is an ellipse and the arena and obstacles are defined as superellipses. We deal with the situations of a sparse map, where there are only a few obstacles and the free space occupies more area, and a narrow map, where much space is occupied by the obstacles and only some narrow passages are free. In order to show the performance, we compare our proposed algorithm with sampled-based algorithms in the well-known and commonly used Open Motion Planning Library (OMPL). And the comparisons include parameters of the vertex number, running time for solving the problem and the success rate after multiple trials of the experiment. The algorithm is implemented in C++ and all the comparisons are running in an Intel Core i7 CPU with 3.60GHz $\times$ 8.

4.1 Planning Map Settings

To make a concrete meaning of the "sparse" and "narrow" map, we define them according to the relative volume of the free space related to the volume of arena. In particular, the

area (since we are in 2D case) of a superellipse is defined as

$$A_{SQ} = 2a_1 b_1 \epsilon B(\frac{\epsilon}{2}, \frac{\epsilon+2}{2}), \tag{30}$$

where $B(x, y)$ is the beta function and can be related to the gamma function as

$$B(x, y) = 2\int_0^{\pi/2} \sin^{2x-1}\phi \cos^{2y-1}\phi d\phi = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}, \tag{31}$$

and a_1, b_1, ϵ are parameters in the explicit expression of the superellipse. As a special case of the superellipse, when $\epsilon = 1$, it becomes to a ellipse, and the area can be computed as $A_{EL} = \pi a_1 b_1$. For the sake of simplicity, the obstacles are placed without overlapping each other and intersecting the arena, so the relative area can be computed as $\rho = 1 - \sum A_{obs} / \sum A_{arena}$.

4.2 Benchmark Settings with OMPL

To ensure a fair comparison with the standard implementations in OMPL, we control the parameters of the algorithms and choose relatively efficient collision checking methods. The performance of the algorithms are benchmarked by evaluating the same metric that combines running time and success rate. And the comparisons are conducted on different shapes of the robot and environment.

4.2.1 Collision Checking Methods for the Sample-Based Algorithms

For sample-based methods, the majority of time is consumed in collision checking, so choosing a relatively fast collision checker is the priority for planning. Here according to the shapes of the obstacles, we select different collision checking methods as follows.

We first deal with the special example of elliptical environment, where all the arena and obstacles are ellipses. People have done a lot of work on collision checking for ellipsoidal bodies, the relatively more efficient of which being the "Algebraic Conditions of Separation". The idea is to count the number of the sign changes of the characteristic polynomial of two ellipsoids, and we select this method for collision checking in the elliptical environment.

Then for the general case, where the environment is modeled by superellipses, the collision checkers are applied through the standard and widely-used "Flexible Collision Library" (FCL). In this case, the superellipses are treated as polygons with discrete points generated along the boundary.

4.2.2 Comparison Metric

The performance of each algorithm is characterized as the planning time and success rate through multiple trials of experiments. For the roadmap planners, the planning time is further divided into map building time and path searching time. We also combine the time and success rate together and define a "nominal planning time" as

$$T_n = T_p / R, \tag{32}$$

where T_p is the actual planning time and R is the success rate.

4.3 Experimental Results

Two shapes of the obstacles are created: ellipses and superellipses. Different sparsities are compared via relative area of the free space. And the comparisons are made between the extended Highway RoadMap and PRM from OMPL. Each map consists of 50 trials. The average roadmap construction time, path search time and success rate are recorded, then the nominal planning time are computed.

Table 1: Comparisons for Highway RoadMap and PRM in different 2D maps

Map	Rel. Area	Planner	N_V	N_p	Planning Time	S.R.	Nominal Time
Ellipse Sparse	83.5%	Highway	289	13	0.0624s	100%	0.0624s
		PRM			1.1394s	82%	1.3895s
Ellipse Narrow	63.3%	Highway	1800	33	0.1943s	100%	0.1943s
		PRM			1.4202s	60%	2.3671s
Ellipse Cluttered	71.25%	Highway	2050	47	0.2420s	100%	0.2420s
		PRM			1.3893s	58%	2.3953s
SQ Sparse		Highway					
		PRM					
SQ Narrow		Highway					
		PRM					
SQ Cluttered		Highway					
		PRM					

Table notes:
Rel. Area: Relative area of the free space;
N_V: Number of valid vertexes on the roadmap;
N_p: Number of vertexes on the path;
S.R: Success rate.

5 Conclusions

References

[1] S. N. Chiu, D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic geometry and its applications*. John Wiley & Sons, 2013.

[2] J.-C. Latombe, "Robot motion planning," 1991.

[3] D. Halperin, J.-C. Latombe, and R. H. Wilson, "A general framework for assembly planning: The motion space approach," *Algorithmica*, vol. 26, no. 3-4, pp. 577–601, 2000.

[4] E. E. Hartquist, J. Menon, K. Suresh, H. B. Voelcker, and J. Zagajac, "A computing strategy for applications involving offsets, sweeps, and minkowski operations," *Computer-Aided Design*, vol. 31, no. 3, pp. 175–183, 1999.

[5] L. Guibas, L. Ramshaw, and J. Stolfi, "A kinetic framework for computational geometry," in *Foundations of Computer Science, 1983., 24th Annual Symposium on.* IEEE, 1983, pp. 100–111.

[6] E. Fogel and D. Halperin, "Exact and efficient construction of minkowski sums of convex polyhedra with applications," *Computer-Aided Design*, vol. 39, no. 11, pp. 929–940, 2007.

[7] 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.

[8] P. K. Agarwal, E. Flato, and D. Halperin, "Polygon decomposition for efficient construction of minkowski sums," *Computational Geometry*, vol. 21, no. 1, pp. 39–61, 2002.

[9] E. Behar and J.-M. Lien, "Dynamic minkowski sum of convex shapes," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 2011, pp. 3463–3468.

[10] C. L. Bajaj and M.-S. Kim, "Generation of configuration space obstacles: The case of moving algebraic curves," *Algorithmica*, vol. 4, no. 1-4, pp. 157–172, 1989.

[11] J.-R. Sack and J. Urrutia, *Handbook of computational geometry.* Elsevier, 1999.

[12] A. Kaul and R. T. Farouki, "Computing minkowski sums of plane curves," *International Journal of Computational Geometry & Applications*, vol. 5, no. 04, pp. 413–432, 1995.

[13] I.-K. Lee, M.-S. Kim, and G. Elber, "Polynomial/rational approximation of minkowski sum boundary curves," *Graphical Models and Image Processing*, vol. 60, no. 2, pp. 136–165, 1998.