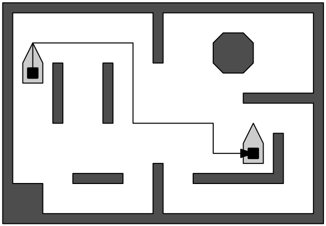
Path Planning and Minkowski Sums (Chapter 13)

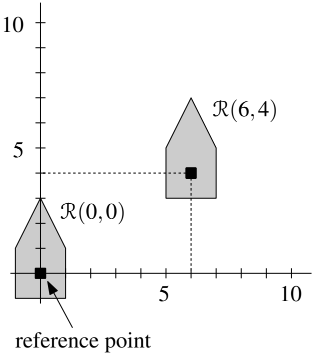
Elisha Sacks

Path Planning



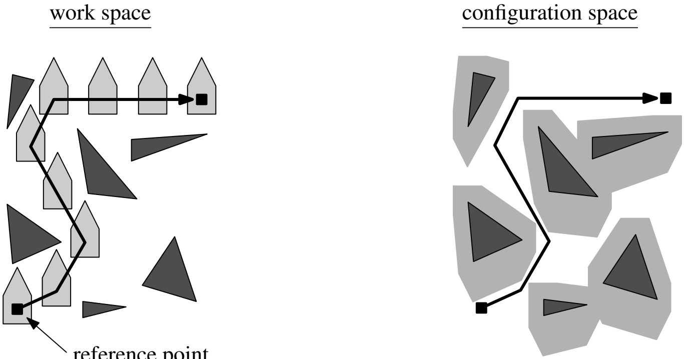
Find a path for a robot between a start configuration and an end configuration. The robot cannot overlap the obstacle.

Polygonal Robot That Translates



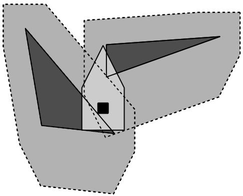
- The configuration of $\mathcal{R}$ is the position of its reference point.
- The textbook uses the notation $\mathcal{R}(a, b)$.
- We will use the notation $(a, b) + \mathcal{R}$.

Configuration Space



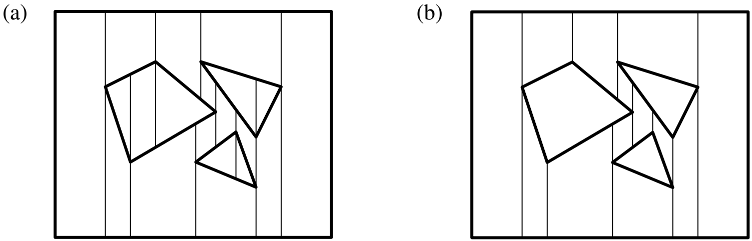
- Configuration space is the set of configurations.
- Free space is where the robot and the obstacle are disjoint.
- Blocked space is where they intersect.
- Contact space is where they touch.
- The blocked space in the example is light and dark grey.

Blocked Space



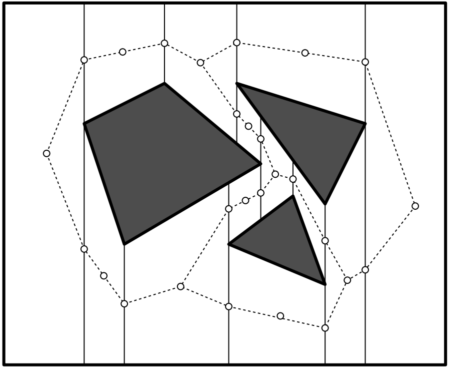
- Each obstacle generates a blocked configuration space region.
- Blocked space is the union of these regions.
- At blocked configurations where multiple regions intersect, the robot intersects the obstacle at multiple points.

Trapezoidal Decomposition



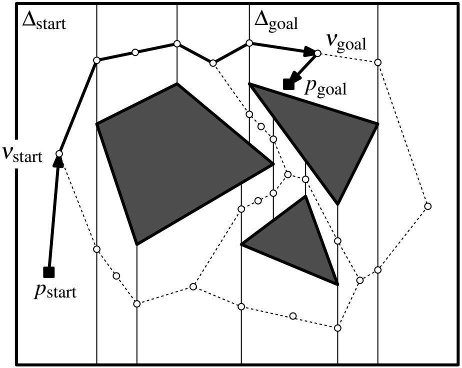
- Trapezoidal decomposition supports path planning.
- Decompose configuration space then drop the blocked faces.

Roadmap



- The roadmap is a planar graph for path planning.
- The nodes are the centers of the faces and the vertical edges.
- Each face node is linked to its edge nodes.

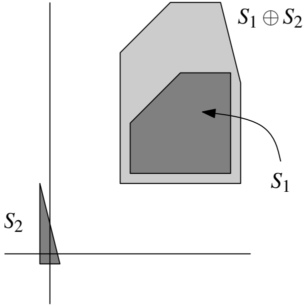
Path Planning Algorithm



1. Create start and goal nodes p_{start} and p_{goal}.
2. Link them to their face nodes v_{start} and v_{goal}.
3. Find a path from v_{start} to v_{goal} by breadth-first search.

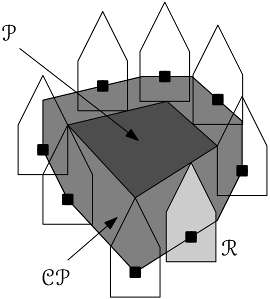
We will discuss shortest paths in Chapter 15.

Minkowski Sum



- The Minkowski sum is a core function of point sets.
  $S_1 \oplus S_2 = \{a + b \mid a \in S_1, b \in S_2\}$
- The blocked space of a translating robot is a Minkowski sum.

Path Planning with Minkowski Sums



Theorem The blocked space $\mathcal{CP}$ of a translating robot $\mathcal{R}$ with respect to an obstacle $\mathcal{P}$ is $\mathcal{P} \oplus (-\mathcal{R})$ where $-\mathcal{R} = \{-r \mid r \in \mathcal{R}\}$.
Proof Let t be a translation. We will show that $t + \mathcal{R}$ intersects $\mathcal{P}$ iff $t \in \mathcal{P} \oplus (-\mathcal{R})$.

Let $q \in (t + \mathcal{R}) \cap \mathcal{P}$. Since $q \in t + \mathcal{R}$, $q - t \in \mathcal{R}$, so $t - q \in -\mathcal{R}$, so $t \in q + (-\mathcal{R})$. Since $q \in \mathcal{P}$, $t \in \mathcal{P} \oplus (-\mathcal{R})$.

Let $t \in \mathcal{P} \oplus (-\mathcal{R})$. There are points $p \in \mathcal{P}$ and $r \in \mathcal{R}$ such that $t = p - r$, so $p = t + r \in t + \mathcal{R}$, so $p \in (t + \mathcal{R}) \cap \mathcal{P}$.

Minkowski Sums of Polygons

Let A and B be polygons in general position.

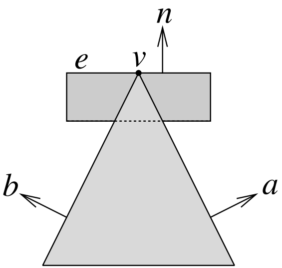
Claim If $a + b$ is a boundary point of $A \oplus B$, a is a vertex and b is on an edge or vice versa.

Proof A point a in the interior of A has a neighborhood $D \subset A$, so $a + b$ is in the interior of $D \oplus b \subset A \oplus B$. Likewise for b in the interior of B. If $a \in e$ and $b \in f$ for edges e and f, e and f are not parallel by general position, so $a + b$ is in the interior of the open set $e \oplus f \subset A \oplus B$.

Claim $A \oplus B$ is a polygon.

Proof For a vertex u and an edge vw, define $u + vw$ as the line segment $[u + v, u + w]$. The boundary of $A \oplus B$ is a subset of the union of these sums over A and B.

Compatible Features

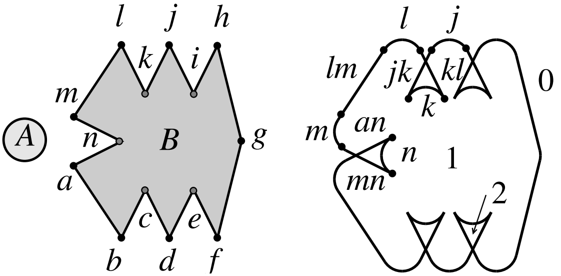


A vertex v of one polygon is compatible with an edge e of the other polygon if the outward normal n of e is between the outward normals a and b of the two edges incident on v.

The Minkowski sum boundary is a subset of the sums of the compatible features.

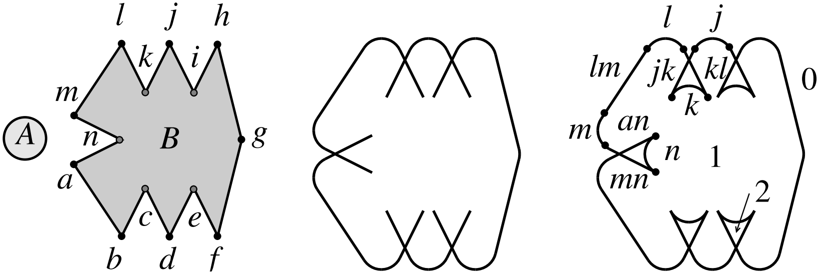
A circular arc is compatible with an edge if their normals at the point of contact are equal.

Kinetic Convolution



- The kinetic convolution is the union of the sums of the compatible pairs of features.
- It defines a subdivision.
- The crossing number of a cell equals the number of intersections of $-A + t$ and B for any t in the cell.
- The Minkowski sum is the union of the cells with positive crossing numbers.

Convex Convolution

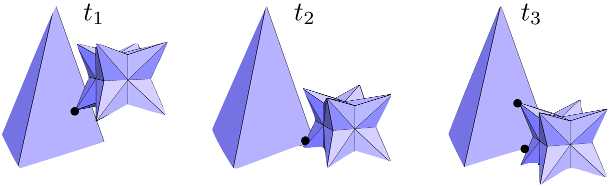


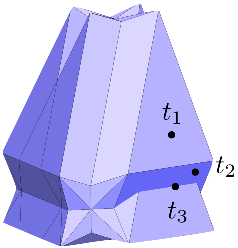
convex convolution

kinetic convolution

- The convex convolution is the subset of the kinetic convolution where the features are convex.
- It defines a subdivision.
- The Minkowski sum is the union of the cells where $-A + t$ intersects B for any t in the cell.

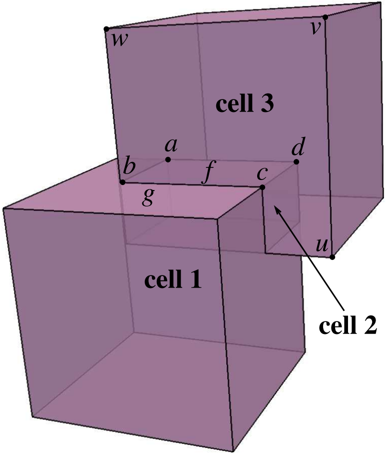
Minkowski Sum of Polyhedra





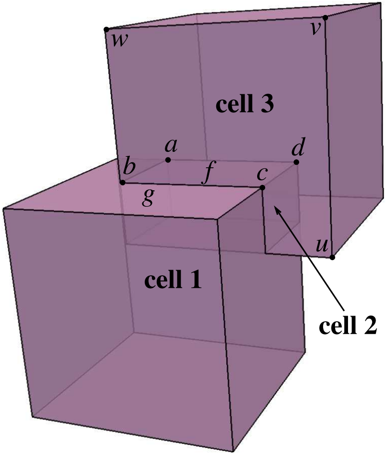
- The Minkowski sum of polyhedra is a polyhedron.
- The facets are subsets of feature sums.
- The kinetic and convex convolutions generalize to 3D.

Boundary Representation



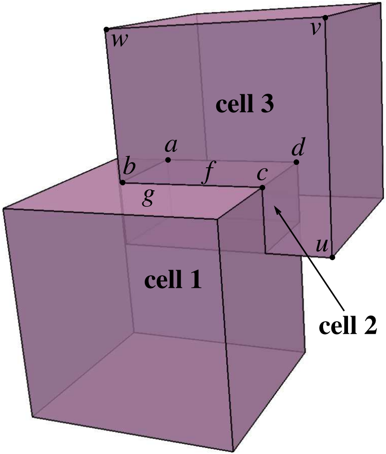
- A shell is a closed surface comprised of facets.
- A cell is an open region bounded by shells.
- Edge loop *abcd* bounds facet *f*.
- Facet *f* bounds cells 2 and 3.

Manifold Surfaces



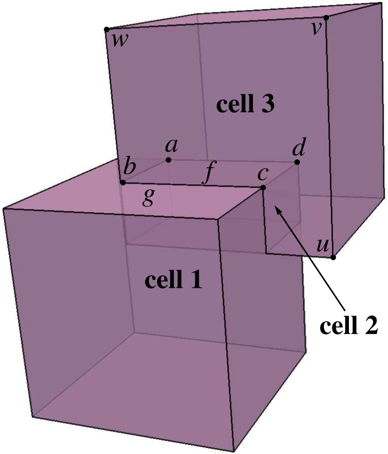
- Manifold surfaces
  - Every edge bounds a single facet.
  - if an edge bounds a facet, so does its twin.
- Are the shells manifold surfaces?

Manifold Surfaces



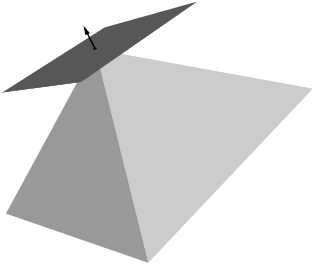
- Manifold surfaces
  - Every edge bounds a single facet.
  - if an edge bounds a facet, so does its twin.
- Are the shells manifold surfaces? Yes.
- Is the entire subdivision a manifold?

Manifold Surfaces

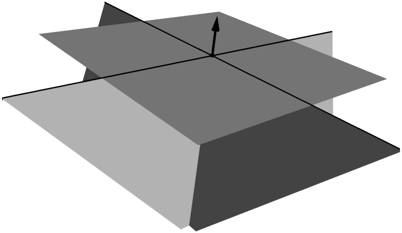


- Manifold surfaces
  - Every edge bounds a single facet.
  - if an edge bounds a facet, so does its twin.
- Are the shells manifold surfaces? Yes.
- Is the entire subdivision a manifold? No.

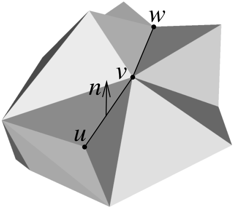
Compatible Features



vertex/facet

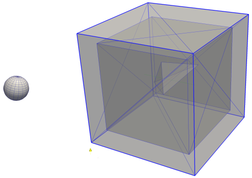


edge/edge

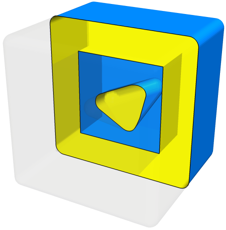


convexity

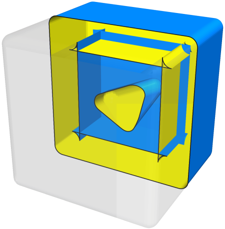
Kinetic and Convex Convolutions



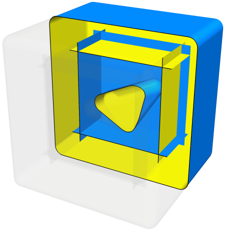
sphere and hollow box



Minkowski sum



kinetic convolution



convex convolution

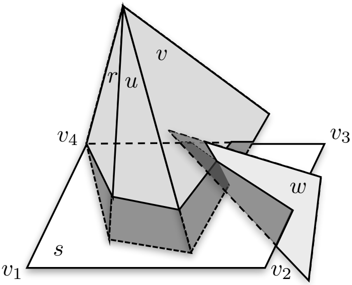
Minkowski Sum Algorithm

Input: polyhedra A and B with triangular facets.

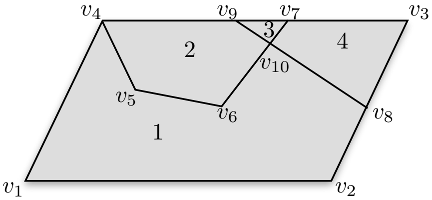
1. Construct convex convolution.
2. Intersect facets.
3. Subdivide facets.
4. Triangulate subdivision faces.
5. Group triangles into surfaces.
6. Classify surfaces as outer or inner.
7. Compute surface nesting and form cells.

Output: triangulated boundary of $A \oplus B$.

Facet Subdivision

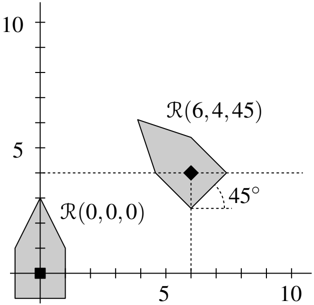


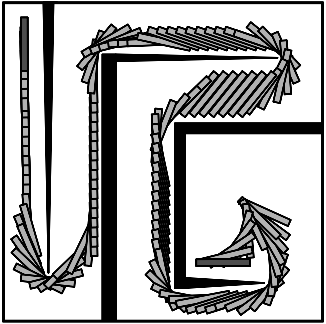
intersecting facets



subdivision of s

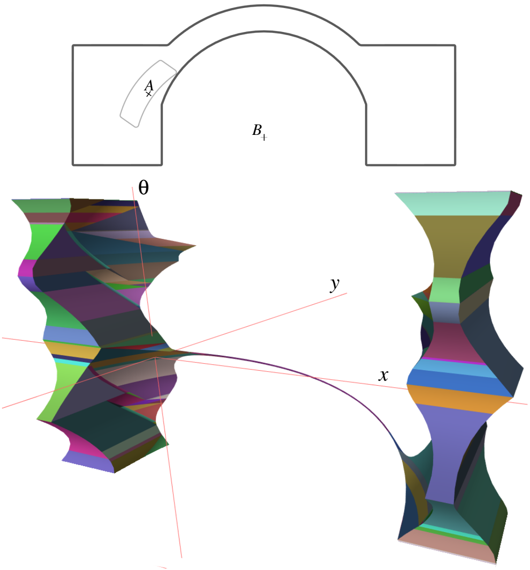
3D Path Planning



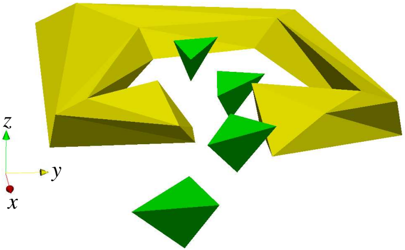


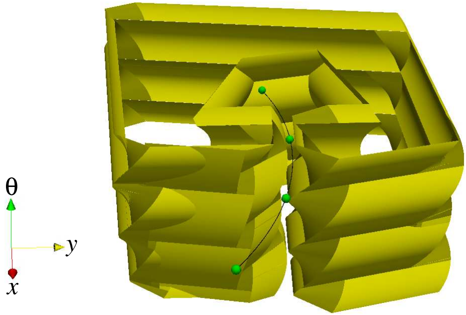
- The robot configuration is (x, y, θ).
- Free space construction is much harder: 3D and nonlinear.
- We have done it robustly with ACP.
- Likewise for a polyhedral robot that moves in a plane.

3D free space for curved planar parts

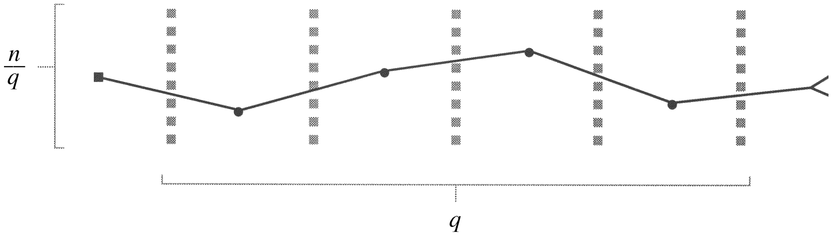


3D free space for polyhedron with planar motion



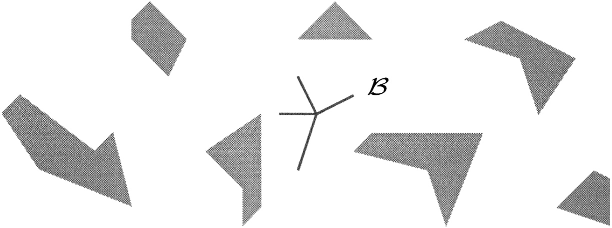


General Path Planning



- Robot has constant complexity and q degrees of freedom.
- Obstacles are disjoint and have constant complexity.
- Robot can touch q objects simultaneously.
- These configurations are free space vertices.
- Free space complexity is $O(n^q)$ for n obstacles.
- Construction and planning times are somewhat higher.
- There are no practical algorithms for $q > 3$.

Low Density Obstacles



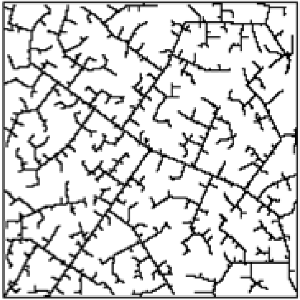
Van der Stappen et al, Motion Planning in Environments with Low Obstacle Density, *Discrete and Computational Geometry,* 20(4):561–587, 1998.

Theorem 2.4 The free space complexity is $O(n)$ for low density obstacles.

Theorem 4.5 The complexity of low density motion planning is $O(n \log n)$ for a robot of size $b\rho$ with b a constant and ρ the minimum obstacle size.

RRT Path Planning

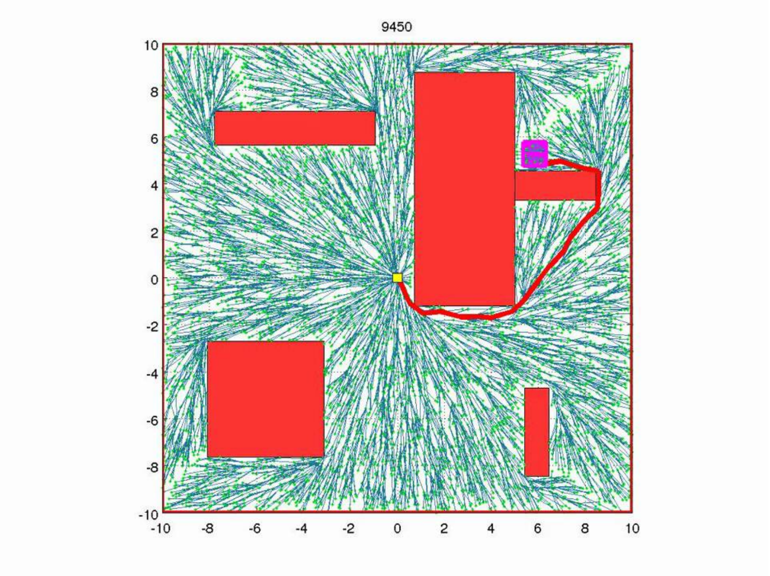
45 iterations



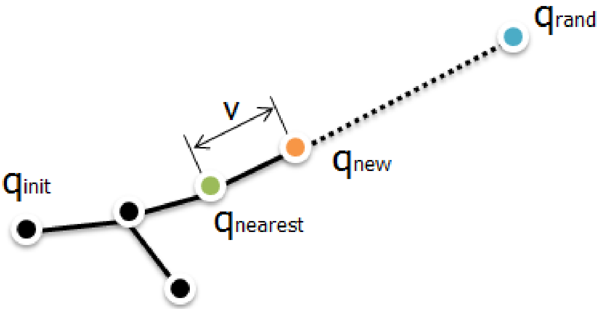
390 iterations

- Build a rapidly exploring random tree (RRT) from a start configuration.
- The vertices and the edges are in free space.
- Find a path from s to g by alternately expanding their trees and checking if the segment between their closest vertices is in free space.

RRT with Obstacles



RRT Algorithm



Input: Initial configuration q_{init}, number of vertices k, distance v.

1. Create a graph G with root q_{init}.
2. Repeat k times
    3. Set q_{rand} to a random configuration.
    4. Set q_{nearest} to the vertex of G closest to q_{rand}.
    5. Set q_{new} at distance v from q_{nearest} on $q_{\text{nearest}}q_{\text{rand}}$.
    6. If q_{new} is free, add it to G and connect it to q_{nearest}.

Evaluation of RRT Planning

- RRT is far simpler than free space construction.
- One algorithm applies to all types of problems.
- RRT performs well when the robot has ample clearance.
- RRT performs poorly on long narrow corridors.
- There are *many* extensions that try to fix this problem.