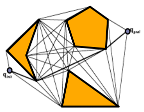
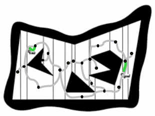
Last lecture

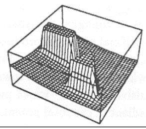
- Path planning for a moving point
  - Visibility graph



  - Cell decomposition



  - Potential field



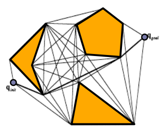
Fundamental question of motion planning

- Given the geometry of
  - Robot
  - Obstacles
- Find a path between initial and goal robot positions
- Key: capture the **connectivity** of the space

Completeness

- A **complete motion planner** always returns a solution when one exists and indicates that no such solution exists otherwise.
  - Is the visibility graph algorithm complete? Yes.
  - How about the exact cell decomposition algorithm and the potential field algorithm?

Geometric Preliminaries



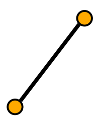
Primitive objects in the plane

- 0-dim

```
class Point {
  double x;
  double y;
}
```

- 1-dim

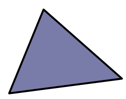
```
class Segment {
  Point p;
  Point q;
}
```



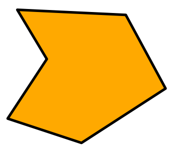
Primitive objects in the plane

- 2-dim

```
class Triangle {
  Point p;
  Point q;
  Point r;
}
```

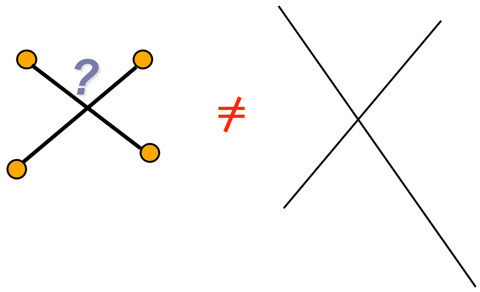


```
class Polygon {
  list<Point> vertices;
  int numVertices;
}
```



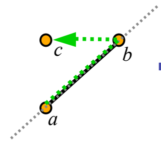
Intersecting primitive objects in the plane

- Case 1: line segment & line segment



Intersect line segments

- Counter-clockwise test



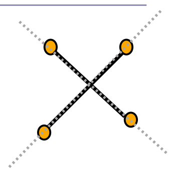
$$\rightarrow \overrightarrow{ab} \times \overrightarrow{bc} \rightarrow \begin{vmatrix} i & j & k \\ x_b - x_a & x_c - x_b & 0 \\ y_b - y_a & y_c - y_b & 0 \end{vmatrix}$$

```
isCcw(Point a, Point b, Point c )
{
   u = b - a;
   v = c - b;
   return  (u.x*v.y - v.x*u.y) > 0;
}
```

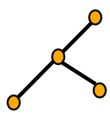

Intersect line segments with ccw

- Pseudo-code

```
isCoincident(Segment s, Segment t)
{
  return xor(isCcw(s.p, s.q, t.p),
             isCcw(s.p, s.q, t.q)) &&
         xor(isCcw(t.p, t.q, s.p),
             isCcw(t.p, t.q, s.q))
}
```

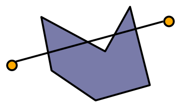


- Special cases

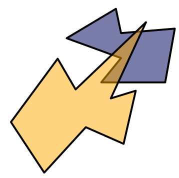


Intersecting primitive objects in the plane

- Case 2 line segment l & polygon P
  - Intersect l with every edge of P



- Case 3 polygon P & polygon Q
  - Intersect every edge of P with every edge of Q
  - More efficient algorithms exist if P or Q are convex.

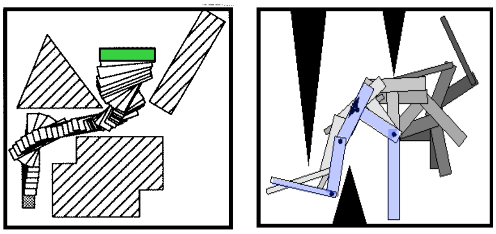


Additional information

- Computational geometry
- Efficient geometry libraries CGAL, LEDA, *etc.*

Configuration Space

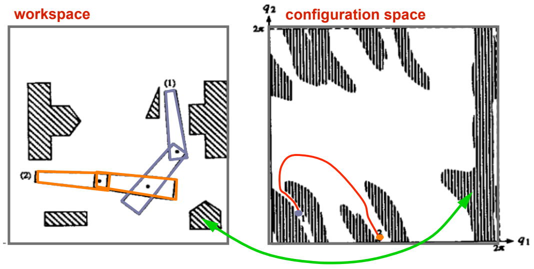
How to specify motion?



Rough idea

- Convert rigid robots, articulated robots, *etc.* into points
- Apply algorithms for moving points

Mapping from the workspace to the configuration space

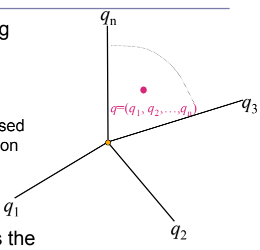


Configuration space

- **Definitions and examples**
- Obstacles
- Paths
- Metrics

Configuration space

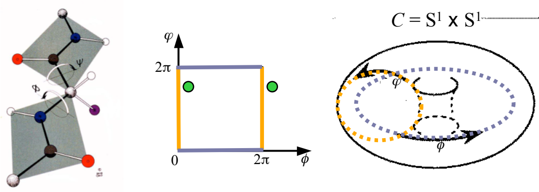
- The **configuration** of a moving object is a specification of the position of **every** point on the object.
  - Usually a configuration is expressed as a vector of position & orientation parameters: $q = (q_1, q_2, \ldots, q_n)$.



- The **configuration space** C is the set of all possible configurations.
  - A configuration is a point in C.

Topology of the configuration pace

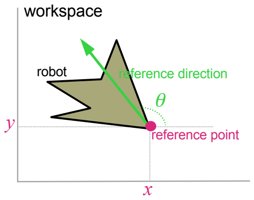
- The topology of C is usually **not** that of a Cartesian space R^n.



Dimension of configuration space

- The **dimension of a configuration space** is the **minimum** number of parameters needed to specify the configuration of the object completely.
- It is also called the **number of degrees of freedom** (dofs) of a moving object.

Example: rigid robot in 2-D workspace



- 3-parameter specification: $q = (x, y, \theta)$ with $\theta \in [0, 2\pi)$.
  - 3-D configuration space

Example: rigid robot in 2-D workspace

- 4-parameter specification: $q = (x, y, u, v)$ with $u^2+v^2 = 1$. Note $u = \cos\theta$ and $v = \sin\theta$.
- dim of configuration space = 3
  - Does the dimension of the configuration space (number of dofs) depend on the parametrization?
- Topology: a 3-D cylinder $C = R^2 \times S^1$

  - Does the topology depend on the parametrization?

Example: rigid robot in 3-D workspace

- q = (position, orientation) = $(x, y, z, ???)$
- Parametrization of orientations by matrix: $q = (r_{11}, r_{12}, \ldots, r_{32}, r_{33})$ where $r_{11}, r_{12}, \ldots, r_{33}$ are the elements of rotation matrix

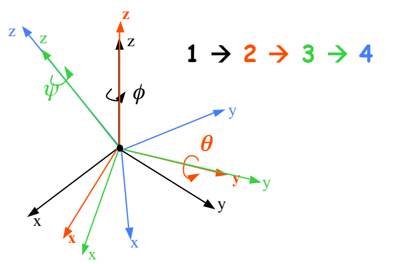
$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

with

  - $r_{1i}^2 + r_{2i}^2 + r_{3i}^2 = 1$ for all i,
  - $r_{1i}r_{1j} + r_{2i}r_{2j} + r_{3i}r_{3j} = 0$ for all $i \neq j$,
  - $\det(R) = +1$

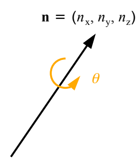
Example: rigid robot in 3-D workspace

- Parametrization of orientations by Euler angles: (ϕ, θ, ψ)



Example: rigid robot in 3-D workspace

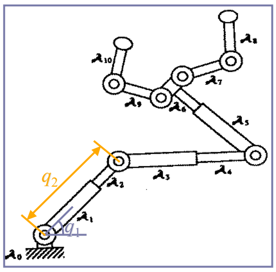
- Parametrization of orientations by **unit quaternion**: $u = (u_1, u_2, u_3, u_4)$ with $u_1^2 + u_2^2 + u_3^2 + u_4^2 = 1$.
  - Note $(u_1, u_2, u_3, u_4) = (\cos\theta/2, n_x\sin\theta/2, n_y\sin\theta/2, n_z\sin\theta/2)$ with $n_x^2 + n_y^2 + n_z^2 = 1$.
  - Compare with representation of orientation in 2-D: $(u_1, u_2) = (\cos\theta, \sin\theta)$



Example: rigid robot in 3-D workspace

- Advantage of unit quaternion representation
  - Compact
  - No singularity
  - Naturally reflect the topology of the space of orientations
- Number of dofs = 6
- Topology: $R^3 \times SO(3)$

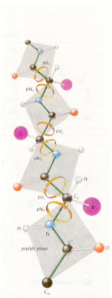
Example: articulated robot



An articulated object is a set of rigid bodies connected at the joints.

- $q = (q_1, q_2, \ldots, q_n)$
- Number of dofs $= n$
- What is the topology?

Example: protein backbone



- What are the possible representations?
- What is the number of dofs?
- What is the topology?

Configuration space

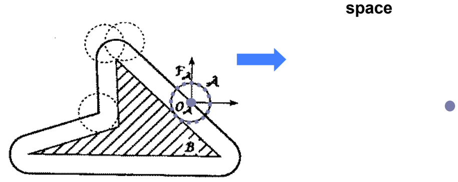
- Definitions and examples
- **Obstacles**
- Paths
- Metrics

Obstacles in the configuration space

- A configuration q is collision-free, or **free**, if a moving object placed at q does not intersect any obstacles in the workspace.
- The **free space** F is the set of free configurations.
- A configuration space obstacle (**C-obstacle**) is the set of configurations where the moving object collides with workspace obstacles or with itself.

Disc in 2-D workspace

workspace **configuration space**

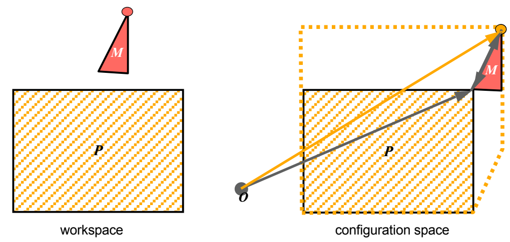


Problem

- **Input:**
  - Convex polygonal moving object translating in 2-D workspace
  - Convex polygonal obstacles
- **Output**: configuration space obstacles represented as polygons

Observation

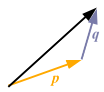
- If P is an obstacle in the workspace and M is a moving object. Then the C-space obstacle corresponding to P is $P \ominus M$.



Minkowski sum

- The **Minkowski sum** of two sets P and Q, denoted by $P \oplus Q$, is defined as

$$P \oplus Q = \{ p+q \mid p \in P, q \in Q \}$$



- Similarly, the **Minkowski difference** is defined as

$$P \ominus Q = \{ p-q \mid p \in P, q \in Q \}$$

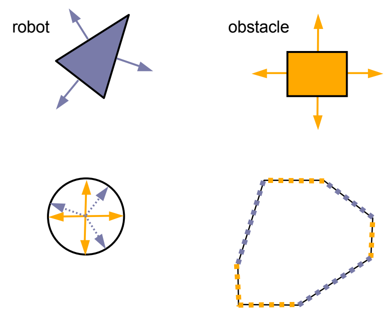
Minkowski sum of convex polygons

- The Minkowski sum of two convex polygons P and Q of m and n vertices respectively is a convex polygon $P \oplus Q$ of $m + n$ vertices.
  - The vertices of $P \oplus Q$ are the "sums" of vertices of P and Q.

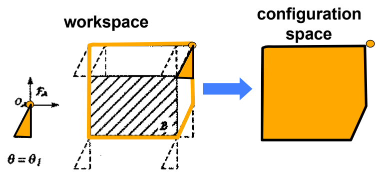
Computational efficiency

- Running time $O(n+m)$
- Space $O(n+m)$
- Non-convex obstacles
  - Decompose into convex polygons (*e.g.*, triangles or trapezoids), compute the Minkowski sums, and take the union
  - Complexity of Minkowksi sum $O(n^2m^2)$
- 3-D workspace

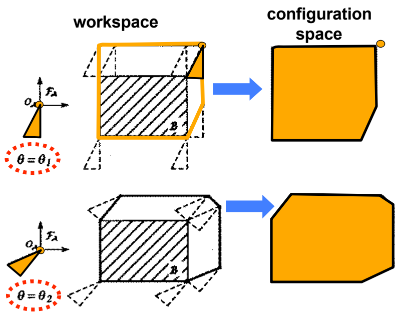
Computing C-obstacles



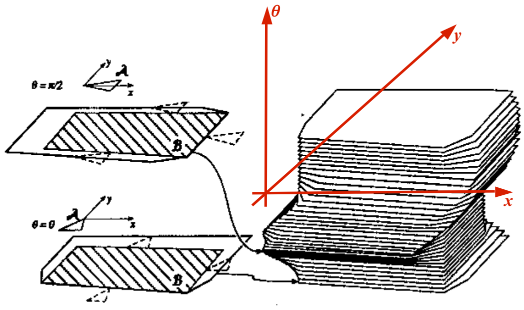
Polygonal robot translating in 2-D workspace



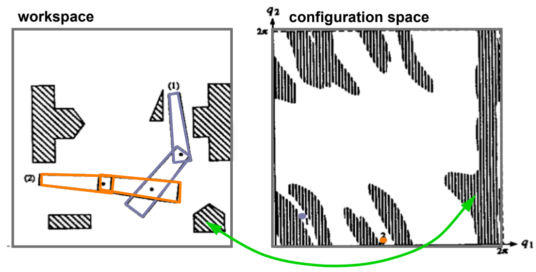
Polygonal robot translating & rotating in 2-D workspace



Polygonal robot translating & rotating in 2-D workspace



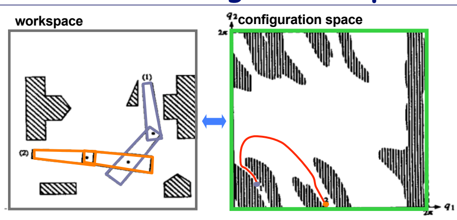
Articulated robot in 2-D workspace



Configuration space

- Definitions and examples
- Obstacles
- **Paths**
- Metrics

Paths in the configuration space



- A **path** in C is a continuous curve connecting two configurations q and q':

$$\tau : s \in [0,1] \rightarrow \tau(s) \in C$$

such that $\tau(0) = q$ and $\tau(1) = q'$.

Constraints on paths

- A **trajectory** is a path parameterized by time:

$$\tau : t \in [0,T] \rightarrow \tau(t) \in C$$

- Constraints
  - Finite length
  - Bounded curvature
  - Smoothness
  - Minimum length
  - Minimum time
  - Minimum energy
  - ...

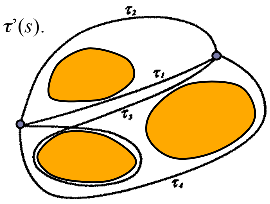
Homotopic paths

- Two paths τ and τ' with the same endpoints are **homotopic** if one can be continuously deformed into the other:

$$h : [0,1] \times [0,1] \rightarrow F$$

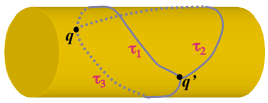
with $h(s,0) = \tau(s)$ and $h(s,1) = \tau'(s)$.

- A homotopic class of paths contains all paths that are homotopic to one another.



Example

- τ_1 and τ_2 are homotopic
- τ_1 and τ_3 are not homotopic
- Infinity number of of homotopy classes exists.



R^1 x S^1

Connectedness of C-Space

- C is **connected** if every two configurations can be connected by a path.
- C is **simply-connected** if any two paths connecting the same endpoints are homotopic.
  Examples: R^2 or R^3
- Otherwise C is multiply-connected.
  Examples: S^1 and $SO(3)$ are multiply- connected:
  - In S^1, infinite number of homotopy classes
  - In $SO(3)$, only two homotopy classes

Configuration space

- Definitions and examples
- Obstacles
- Paths
- **Metrics**

Metric in configuration space

- A **metric** or **distance** function d in a configuration space C is a function

$$d : (q,q') \in C^2 \rightarrow d(q,q') \geq 0$$

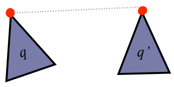
such that
  - $d(q, q') = 0$ if and only if $q = q'$,
  - $d(q, q') = d(q', q)$,
  - $d(q,q') \leq d(q,q'') + d(q'',q')$.

Example

- Robot A and a point a on A
- $a(q)$: position of a in the workspace when A is at configuration q
- A distance d in C is defined by

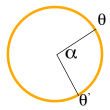
  $d(q, q') = \max_{a \in A} \| a(q) - a(q') \|$

  where $\|a - b\|$ denotes the Euclidean distance between points a and b in the workspace.



Examples in R^2 x S^1

- Consider R^2 x S^1
  - $q = (x, y, \theta)$, $q' = (x', y', \theta')$ with θ, $\theta' \in [0,2\pi)$
  - $\alpha = \min \{ |\theta - \theta'| , 2\pi - |\theta - \theta'| \}$

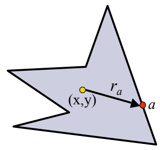


- $d(q,q') = \max_{a \in A} \| a(q) - a(q') \|$

$$= \max_{a \in A} \sqrt{(x-x')^2 + (y-y')^2 + \alpha r_a}$$

$$= \sqrt{(x-x')^2 + (y-y')^2 + \alpha \max_{a \in A} r_a}$$

$$= \sqrt{(x-x')^2 + (y-y')^2 + \alpha r_{\max}}$$



Summary on configuration space

- Parametrization
- Dimension (dofs)
- Topology
- Metric