Motion Planning Lecture 2

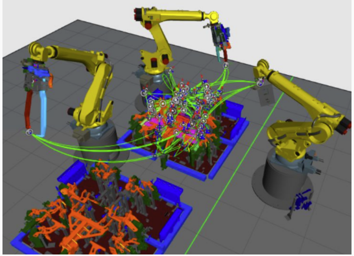
The Structure of Configuration Spaces: Topology, Metrics, Constraints

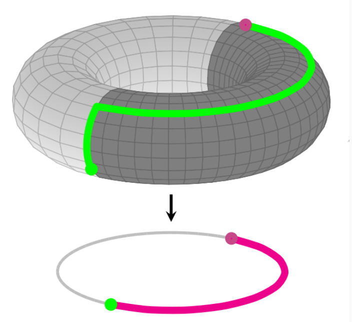
Wolfgang Hönig (TU Berlin) and Andreas Orthey (Realtime Robotics)

May 2, 2023

Introduction Andreas Orthey

realtime ROBOTICS





- Staff Robotics Scientist at Realtime Robotics
- Research on Abstraction Hierarchies in Motion Planning

Recap Last Week

Last week

- Motion Planning: The task of moving robots from A to B
- Fundamental to automation, autonomous driving, health care, games/animation

Terminology

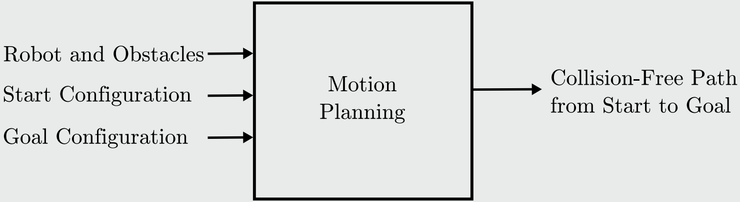
- Configuration space
- Degrees of Freedom
- Configuration map

Today

- Topological Spaces
- Metric Spaces
- Constraints and Collision Checking

Motivation

The Motion Planning Problem



Motivation

Main Idea

Any robot can be modeled as a point in a configuration space
(1979, Lozano-Pérez and Wesley [1])

The Motion Planning Problem

Motivation

Example: Animation of configuration space for a 2-dof manipulator arm

https://aorthey.github.io/configuration-space-visualizer/js-cspace/

Configuration space visualizer

Motivation

Implications

- Configuration space as general purpose modeling tool for any robot.
- One algorithm could solve every problem
- If you want to move robots, you need to understand configuration spaces.

Today's Topics

Our goal for today

Understanding the structure of configuration spaces

Outline

1. Topological spaces: Modelling configuration spaces
2. Metric spaces: Measuring distances in configuration space
3. Constraints: Feasible configurations, collision checking

Topological Spaces

Topology

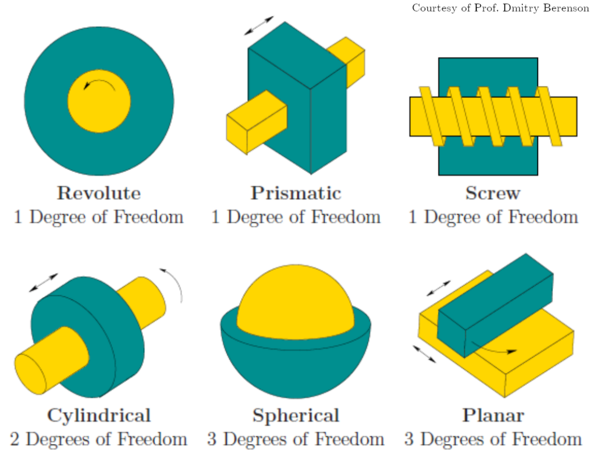
Etymology of Topology

Topos (place, region, space) + Logos (Study) $\longrightarrow$ Study of space

Topology (Mathematics)

Study of properties of geometric objects **invariant** under continuous transformation

Toplogy in Robotics



Motivation: Model configuration spaces of arbitrary robots.

Main idea

Topology as a tool to categorize spaces.

Classification Tool

- What space is associated to a given robot?
- Which robots have topologically equivalent spaces?
- How do two motion planning problems differ?
- What is the computational complexity of a given category of spaces?

Relevant Topics in Topology for Motion Planning

1. Classification of Spaces as Equivalent
2. Combine spaces into Compound spaces
3. Assign spaces to robots

Topological Spaces

Equivalence of Spaces

Topological Spaces

- Thousands of robots might look different from the outside, but they might share a topologically identical configuration space

- Great way to "abstract away" details of the task, and concentrate on the computational challenge

How to establish equivalence?

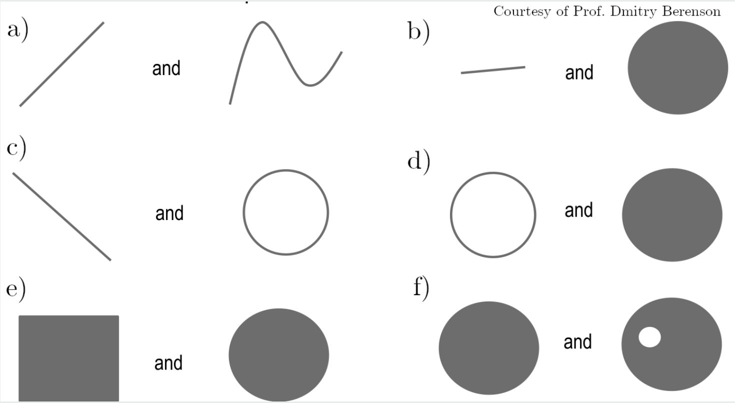
Establishing Equivalence

Two spaces are equivalent, if there exists a homeomorphism between them.

Definition homeomorphism

- A homeomorphism is a mapping between two topological spaces X and Y.
- A homeomorphism is defined as a function $f : X \to Y$ such that
  - f is bijective (one-to-one and onto)
  - f is continuous
  - f^{-1} is continuous
- If a homeomorphism exists, X and Y are said to be equivalent (homeomorphic).
- Intuition: Squeezing, stretching, bending of space

Equivalent spaces under homeomorphism



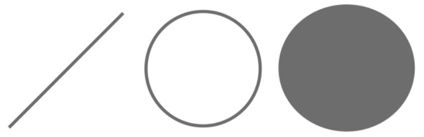
Exercise: Which spaces are homeomorphic?

(You are not allowed to cut holes, tear the space, glue stuff)

Equivalent spaces under homeomorphism

(a) Yes
(b) No, there is no bijection from line to disk (ambiguous)
(c) No, you would need to cut circle (not continuous)
(d) No, same as b)
(e) Yes
(f) No, not continuous

Prototype Spaces



Prototype Spaces

- $\mathbb{R}^1 =]-\infty, +\infty[$ (real number line)
- $S^1 = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1\}$ (circle)
- $D^2 = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$ (disk)
- Naming convention: $\{\text{Symbolic name}\}^{\{\#\text{dimensions}\}}$.

Exercise: Prove that $f(x) = \frac{x}{x^2-1}$ is a homeomorphism from $[-1, +1]$ to $\mathbb{R}^1$.

Topological Spaces

Compound Spaces

Compound Spaces

Atlas Robot by Boston Dynamics

Compound Spaces

Cartesian Product

Given two spaces X, Y, the cartesian product is defined as

$$X \times Y = \{(x, y) \mid x \in X \text{ and } y \in Y\}.$$

Intuition: For every element of a space X, attach a "copy" of space Y.

Compound Spaces

- $\mathbb{R}^1 \times \mathbb{R}^1$ (the plane $\mathbb{R}^2$)
- $\mathbb{R}^1 \times S^1$ (a cylinder)
- $S^1 \times S^1$[!] (the torus T^2, not the sphere S^2)
- $D^2 \times \mathbb{R}^1$ (?)
- $D^2 \times S^1$ (?)

Formalizing Joints

Robotics-Related Spaces

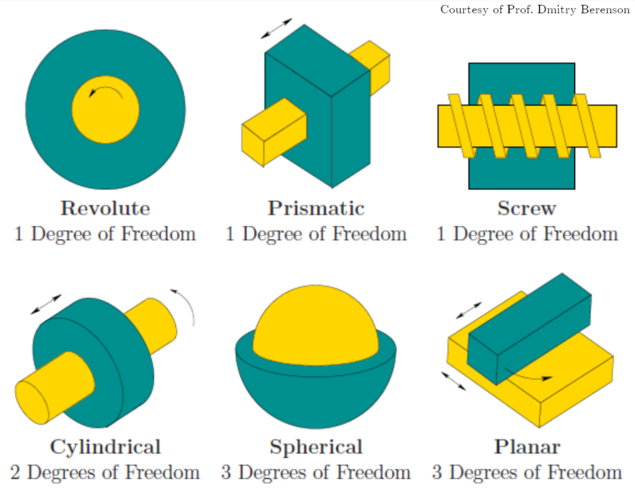
- S^1 (a revolute joint)
- $\mathbb{R}^1$ (a prismatic joint)
- $\mathbb{R}^2$ (a planar disk robot)
- $\mathbb{R}^2 \times S^1$ (a disk robot with a direction)
- Shortcut: $SE(2) = \mathbb{R}^2 \times S^1$ (the special euclidean group)

A group is a set plus a transformation (called the group action), which is closed with respect to the set (applying the action onto the set will result in another member of the set). The euclidean group $E(n)$ are all possible positions of a rigid body in n-dimensional space plus all transformations which keep the shape of the rigid body the same (the group action preserves euclidean distance between any two points). The special euclidean group $SE(n)$ is a subgroup of $E(n)$ consisting of all transformations minus reflections (e.g. mirroring).

Formalizing Joints

3D Robotics-Related Spaces

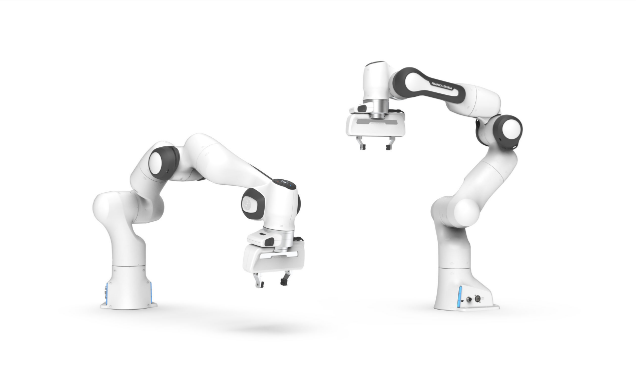
- $SO(2) = S^1$ (all rotations of a rigid body in 2D space)
- $SE(2) = \mathbb{R}^2 \times S^1$ (all rotations and translations of a rigid body in 2D space)
- $SO(3)$ (all rotations of a rigid body in 3D space)
- $SE(3) = \mathbb{R}^3 \times SO(3)$ (all rotations and translations of a rigid body in 3D space)



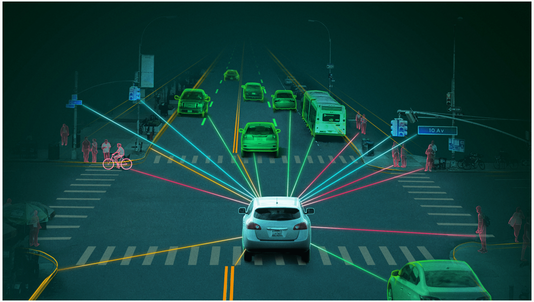
Exercise: Assign the correct topological spaces to each joint.

Topological Spaces

Topological spaces in the wild



$\mathbb{R}^7$ fixed-based manipulator with 7 degrees of freedom (Panda by Franka emica)



$SE(2) = \mathbb{R}^2 \times S^1$ motions of a mobile base robot

$SE(2) \times \mathbb{R}^{31}$ a fixed-base manipulator robot with a mobile base

$SE(3) = \mathbb{R}^3 \times SO(3)$ motions of a rigid body in space
$SO(3)$ free rotation around a point (pitch, roll, yaw)

$SE(3) \times \mathbb{R}^6$ rigid body in space plus manipulator arm

Summary Topological Spaces

What we learned

- Joint type to mathematical space
- Knowledge of Cartesian products of spaces
- Constructing a configuration space from a robot's joints
- Being able to determine equivalence of configuration spaces

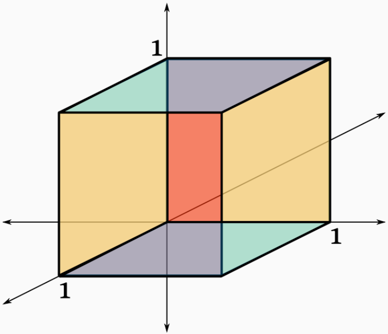
Metric Spaces

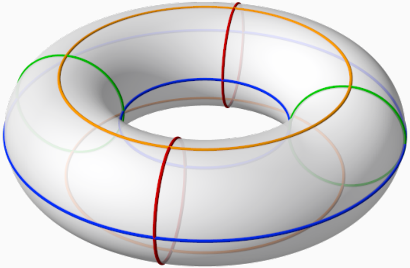
Motivation

Metric Spaces

We want to measure distances in configuration spaces. Here is why:

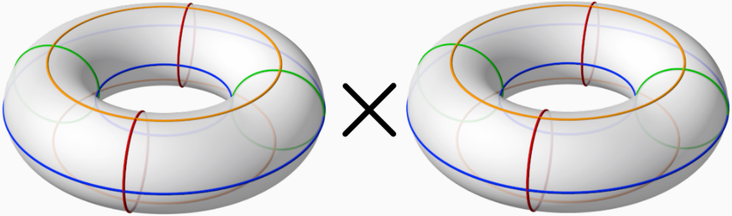
- How far are we away from a goal?
- Informed vs. Uninformed search
- Where should we explore next?





Problems

- Metrics on non-euclidean spaces



Problems

- Metrics on non-euclidean spaces
- Metrics on Cartesian products

Metrics are a choice

A space does not dictate the metric. You choose the metric depending on task requirements, objectives, computational cost.

Choosing Metrics

How do we choose a metric?

- Ideal option: The true distance between points (including constraints)
- Default option: Length of shortest distance paths (geodesics)
- Based on low computational cost
- Based on promotion of better quality paths

Metric Spaces

Metric Spaces: Topological Space + Distance between points (a metric)

Definition Metric Spaces

A metric (or distance) function d in a topological space X is a function $d : X \times X \to \mathbb{R}_{\geq 0}$ such that

1. $d(x, x') = 0$ iff $x = x'$ (identity of indiscernibles)
2. $d(x, x') = d(x', x)$ (symmetry)
3. $d(x, x') \leq d(x, x'') + d(x'', x')$ (triangle inequality)

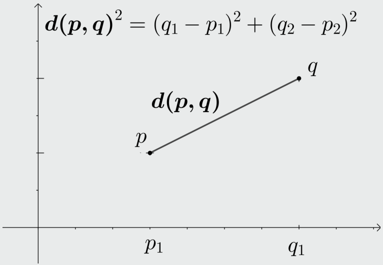
Metric Spaces

Examples of Metrics

Euclidean metric (Straight-Line distance)

Length of line segment between two points

$d(x, x') = \sqrt{\sum_i (x_i - x_i')^2} = \|x - x'\|_2.$



- The most commonly used metric on $\mathbb{R}^n$

Manhattan (Taxicab) metric

Manhattan metric is the sum of absolute values of each dimension: $d(x, x') = \sum_i |x_i - x'_i| = \|x - x'\|_1$.

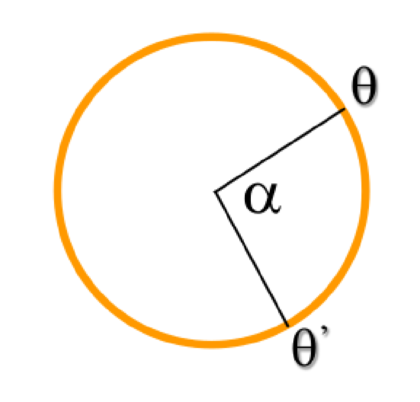
- Your robot has joints which can only be actuated individually
- Set of objects: Only one object can move at a time

Metric Spaces

Metrics on non-euclidean spaces

Circular metric

- Euclidean distance breaks down on spaces like the circle
- Between two distinct points, there are two paths: clockwise or counterclockwise
- Metric: $d(\theta, \theta') = \min\{|\theta - \theta'|, 2\pi - |\theta - \theta'|\}$

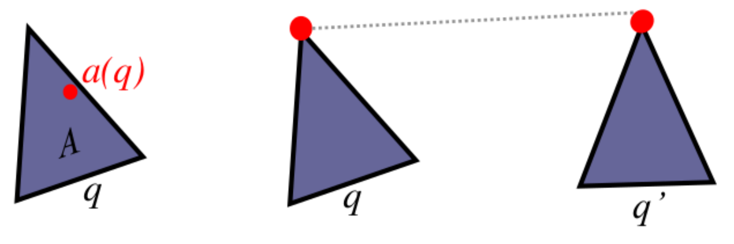


Circular metric

- Distance between two points $\theta = (-2.1, 0)$ and $\theta' = (+2.1, 0)$ on $S^1 \times \mathbb{R}^1$.
- Euclidean metric: $d(\theta, \theta') = \|\theta - \theta'\| = \sqrt{(2.1 - (-2.1))^2} = 4.2$
- Circular metric: $d(\theta, \theta') = \min\{|\theta - \theta'|, 2\pi - |\theta - \theta'|\} = \min\{4.2, 2.08\} = 2.08$

Workspace metric

1. Euclidean distance of the "most-displaced" point: $d(x, x') = \max_{a \in A} \|a(x) - a(x')\|_2$.
2. $a(x)$ is a point on robot A when at configuration x (similar to the configuration map $\mathcal{B}$ from lecture 1).
3. In practice, we often use designated points (like the end-effector)

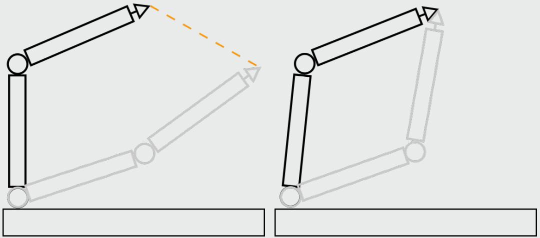


Metric Spaces

Exotic Metrics

Pseudometric

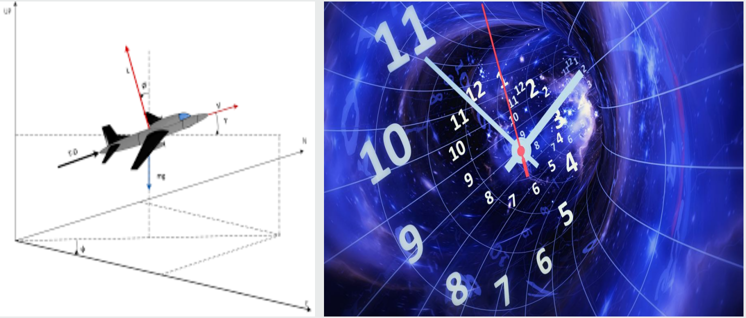
1. ~~$d(x, x') = 0$ iff $x = x'$ (identity of indiscernibles)~~
2. $d(x, x') = d(x', x)$ (symmetry)
3. $d(x, x') \leq d(x, x'') + d(x'', x')$ (triangle inequality)



End-effector distance pseudometric

Quasimetrics

1. $d(x, x') = 0$ iff $x = x'$ (identity of indiscernibles)
2. $\cancel{d(x, x') = d(x', x)}$ ~~(symmetry)~~
3. $d(x, x') \leq d(x, x'') + d(x'', x')$ (triangle inequality)

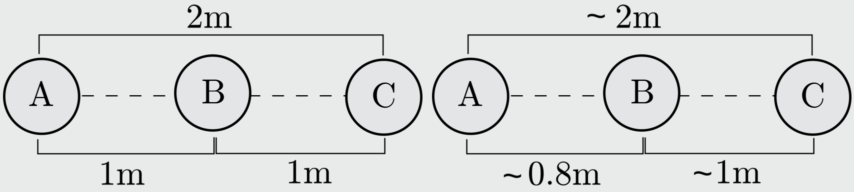


Dynamic-system metric or time-based metric

Semimetric

1. $d(x, x') = 0$ iff $x = x'$ (identity of indiscernibles)
2. $d(x, x') = d(x', x)$ (symmetry)
3. $\cancel{d(x, x') \leq d(x, x'') + d(x'', x')}$ ~~(triangle inequality)~~

Measurement error in position, i.e. $d(A, C)$ larger than $d(A, B) + d(B, C)$.



Measurement-based metric

Metric Spaces

Metric Proof

Prove Manhattan Metric is a metric

Usefulness of Proof

- Motion planners can exploit properties of metrics
- Gives you the tools to decide or adjust custom metrics

Prove Manhattan Metric is a metric

Theorem

Manhattan distance $d(x, x') = |x_1 - x_1'| + |y_1 - y_1'|$ in $\mathbb{R}^2$ is a metric.

Proof (0)

We need to prove the three requirements of a metric. Let x and x' be two elements of $\mathbb{R}^2$.

1. Identity of indiscernibles
2. Symmetry
3. Triangle Inequality

Prove Manhattan Metric is a metric

Theorem

Manhattan distance $d(x, x') = |x_1 - x_1'| + |y_1 - y_1'|$ in $\mathbb{R}^2$ is a metric.

Proof (1)

1. Identity of indiscernibles: $d(x, x') = 0$ iff $x = x'$. Need to prove two directions:
   $\Rightarrow$ Assume $d(x, x') = 0$. Then $|x_1 - x_1'| + |y_1 - y_1'| = 0$. Four cases:
   1.1 $x_1 - x_1' < 0$, $y_1 - y_1' < 0$: $-x_1 + x_1' - y_1 + y_1' = 0$
   1.2 $x_1 - x_1' < 0$, $y_1 - y_1' \geq 0$: $-x_1 + x_1' + y_1 - y_1' = 0$
   1.3 $x_1 - x_1' \geq 0$, $y_1 - y_1' < 0$: $x_1 - x_1' - y_1 + y_1' = 0$
   1.4 $x_1 - x_1' \geq 0$, $y_1 - y_1' \geq 0$: $x_1 - x_1' + y_1 - y_1' = 0$

   By writing out $(1.1) - (1.3)$ and $(1.2) - (1.4)$, we get $x_1 = x_1'$ and $y_1 = y_1'$.
   $\Leftarrow$: Assume $x = x'$. Then $|x_1 - x_1'| = 0$ and $|y_1 - y_1'| = 0$, and therefore $d(x, x') = 0$.

Prove Manhattan Metric is a metric

Theorem

Manhattan distance $d(x, x') = |x_1 - x_1'| + |y_1 - y_1'|$ in $\mathbb{R}^2$ is a metric.

Proof (2)

(2) Symmetry: $d(x, x') = |x_1 - x_1'| + |y_1 - y_1'| \overset{|a|=|-a|}{=} |x_1' - x_1| + |y_1' - y_1| = 0.$

Prove Manhattan Metric is a metric

Theorem

Manhattan distance $d(x, x') = |x_1 - x_1'| + |y_1 - y_1'|$ in $\mathbb{R}^2$ is a metric.

Proof (3)

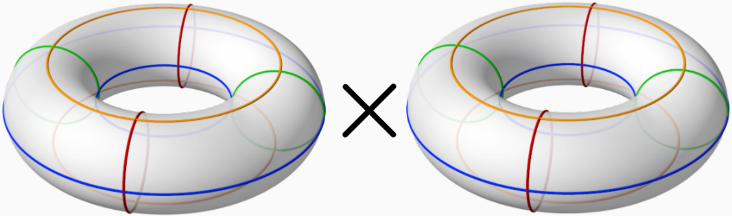
(3) Triangle Inequality:

$$\begin{aligned}
d(x, x') &= |x_1 - x_1'| + |y_1 - y_1'| && (1)\\
&= |x_1 - x_1' + x_1'' - x_1''| + |y_1 - y_1' + y_1'' - y_1''| && (2)\\
&= |(x_1 - x_1'') + (x_1'' - x_1')| + |(y_1 - y_1'') + (y_1'' - y_1')| && (3)\\
&\leq |x_1 - x_1''| + |y_1 - y_1''| + |x_1'' - x_1'| + |y_1'' - y_1'| && (4)\\
&= d(x, x'') + d(x'', x') && (5)
\end{aligned}$$

Using $|a + b| \leq |a| + |b|$.

Metric Spaces

Compound Metrics



Compound metrics

How can we create metrics on Cartesian products?

Compound metrics

Taking the sum is a straightforward way to define a metric.

Theorem

Let (X, d_X) and (Y, d_Y) be metric spaces and let $Z = X \times Y$.
Then (Z, d_Z) is a metric space if $d_Z = d_X + d_Y$.

Compound metrics

Theorem

Let (X, d_X) and (Y, d_Y) be metric spaces and let $Z = X \times Y$.
Then (Z, d_Z) is a metric space if $d_Z = d_X + d_Y$.

Proof (0)

We need to prove the three requirements of a metric.

1. Identity of indiscernibles
2. Symmetry
3. Triangle Inequality

Compound metrics

Theorem

Let (X, d_X) and (Y, d_Y) be metric spaces and let $Z = X \times Y$.
Then (Z, d_Z) is a metric space if $d_Z = d_X + d_Y$.

Proof (1)—Identity of indiscernibles

We want to show that $d_Z(z, z') = 0$ iff $z = z'$ with $z = (x, y)$.

($\Longrightarrow$) Assume $d_Z(z, z') = 0$. Then $d_Z(z, z') = d_X(x, x') + d_Y(y, y') = 0$.

- This implies $d_X(x, x') = d_Y(y, y') = 0$ (since $d_X, d_Y \geq 0$).
- This implies $x = x'$, $y = y'$ and therefore $z = z'$

($\Longleftarrow$) Assume $z = z'$. Then $x = x'$ and $y = y'$.

- It follows that $d_Z(z, z') = d_X(x, x') + d_Y(y, y') = 0$ (by property of d_X, d_Y)

Compound metrics

Theorem

Let (X, d_X) and (Y, d_Y) be metric spaces and let $Z = X \times Y$.
Then (Z, d_Z) is a metric space if $d_Z = d_X + d_Y$.

Proof (2)—Symmetry

$$d_Z(z, z') = d_X(x, x') + d_Y(y, y') = d_X(x', x) + d_Y(y', y) = d_Z(z, z')$$

This is true by the symmetry of d_X, d_Y.

Compound metrics

Theorem

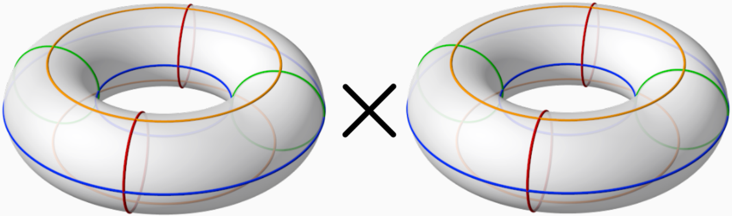
Let (X, d_X) and (Y, d_Y) be metric spaces and let $Z = X \times Y$.
Then (Z, d_Z) is a metric space if $d_Z = d_X + d_Y$.

Proof (3)—Triangle Inequality

We want to show $d_Z(z, z') \leq d_Z(z, z'') + d_Z(z'', z')$.

$$\begin{aligned}
d_Z(z, z') &= d_X(x, x') + d_Y(y, y') \\
&\leq d_X(x, x'') + d_X(x'', x') + d_Y(y, y'') + d_Y(y'', y') \\
&= d_X(x, x'') + d_Y(y, y'') + d_X(x'', x') + d_Y(y'', y') \\
&= d_Z(z, z'') + d_Z(z'', z')
\end{aligned}$$

$\square$



Compound metrics

Taking sum of individual metrics produces a compound metric.

Recap Metric Spaces

1. There is no single best metric for a problem
2. Default choice is the **geodesic-based metric** (length of shortest path)
3. Choice is also affected by compute time and path quality
4. Compound metrics by sum of individual metrics

Constraints

Motivation

Configuration space allows all possible motions.

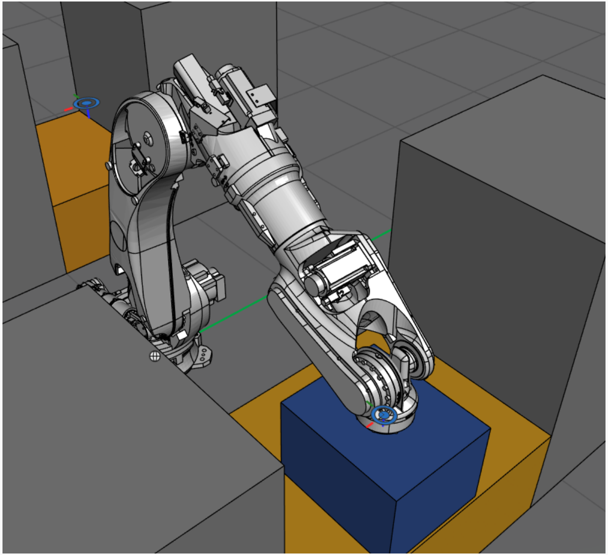
Motivation

Constraints

Depending on the desired task, you need to restrict the motions.

This is accomplished by **constraints on the configuration space**.

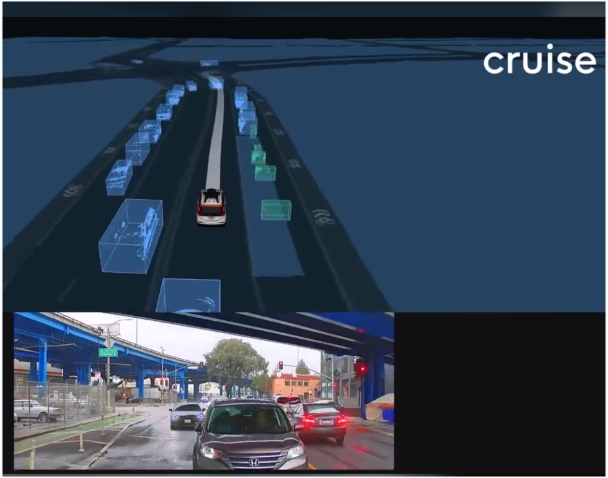
Motivation: Package transport robot



Robot Constraints

- Robot should not collide with external objects
- Robot should not self-collide
- Robot should keep end-effector in certain orientation

Motivation: Autonomous driving



Robot Constraints

- Robot should avoid future car collisions
- Robot should keep lane

Motivation: Walking robot

Robot Constraints

- Robot should not collide with itself
- Robot should not fall down (Static stability)

Motivation

Robot Constraints

How can we formalize those requirements?

- Constraint function $\phi : \mathcal{Q} \to \{\text{True}, \text{False}\}$

Constraint examples

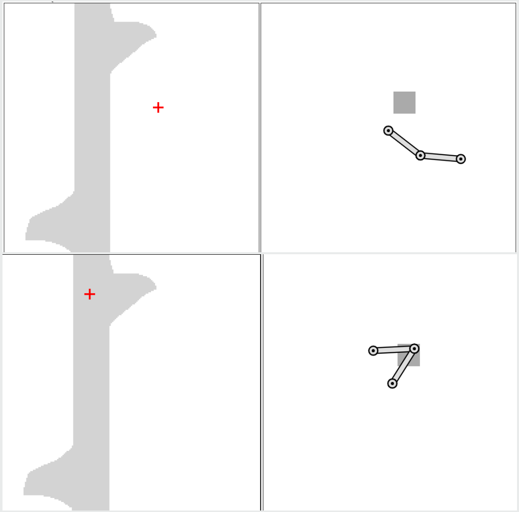
- Collision constraint
- Stability constraint

Constraints

Collision checking

Collision Checking

Function `IsValid(q)`: $\mathcal{Q} \rightarrow \{\text{True}, \text{False}\}$ Is robot at configuration q collision-free?



True

False

Collision Checking: Outline

- Representation
- Computational Complexity
- Broad Phase and Bounding Volumes
- Narrow Phase and Gilbert-Johnson-Keerthi (GJK)
- Flexible collision library (FCL)

Collision Checking: Representation

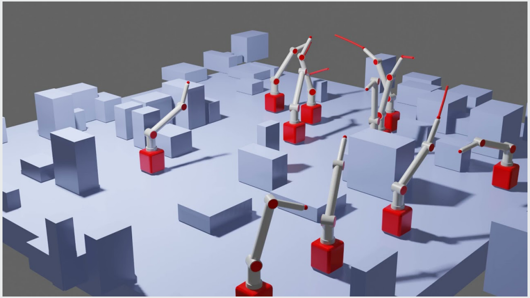
Representation of Obstacles and Robot Links

- Polygon Mesh (Known Obstacles)
    - Objects are represented using sets of triangles
- Voxel-representation (Unknown obstacles)
    - Objects are represented using sets of voxels
    - Red Green Blue Depth (RGBD) Camera
    - Light Detection and Ranging (LIDAR)
    - OctoMap Library

Collision Checking: Computational complexity

Let us assume there are n rigid bodies in our scene.

- Collision checking has worst-case complexity of $\mathcal{O}(n^2)$ (requires $\frac{n(n-1)}{2}$ collision checks)
- If you have a { mesh | voxel } representations, you need to check every pair of { triangles | voxels }.



Constraints

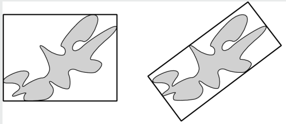
Collision checking: Broad phase vs. Narrow phase

Main idea

Use a broad phase to prune collision pairs. This can lower runtime significantly.

Broad Phase Collision Checking

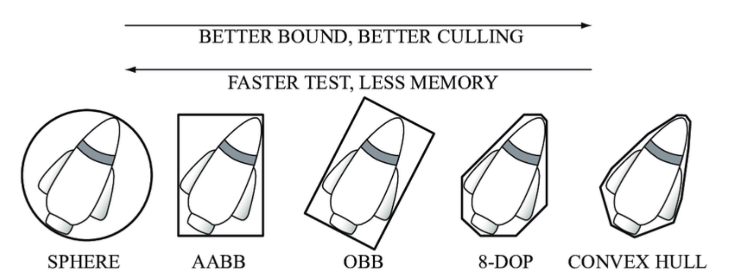
- Represent rigid bodies using bounding volumes.
- Check if bounding volumes intersect.
- This is conservative
  - If they do not intersect, we can prune pair
  - If they intersect, we need to go to a narrow phase



NOTE: This method is actually part of a general pattern, which is quite ubiquitous in planning: we simplify a problem to get a necessary condition (here: overlap of shapes as necessary condition to find intersections), solve this problem, then use the solution to solve the original problem. We come back to this when talking about admissible heuristics.

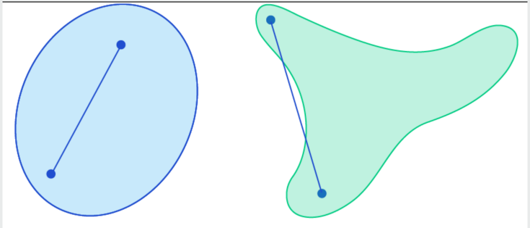
Types of Bounding Volumes

- Bounding sphere
- Axis-aligned bounding box (AABB)
- Oriented bounding box (OBB)
- Discrete oriented polytope (DOP)
- Convex Decomposition



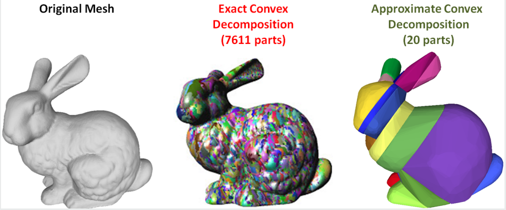
Convex shapes

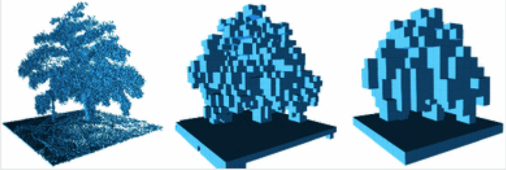
- Definition: A set X is convex, if for any two points x, y in X, there exists a line segment lying in X
- Most shapes are decomposable into convex shapes



Narrow Phase Collision Checking

- Exact collision checking for all pairs which have not been pruned in broad-phase
- Widely used strategy: Convex collision checking between two pairs of objects.
    - Decompose every object into convex shapes
    - Check collision between every two convex shapes
- Collision checking for convex shapes is cheap (see GJK)





Taken from "OctoMap: an efficient probabilistic 3D mapping framework based on octrees"

Constraints

Gilbert-Johnson-Keerthi (GJK)

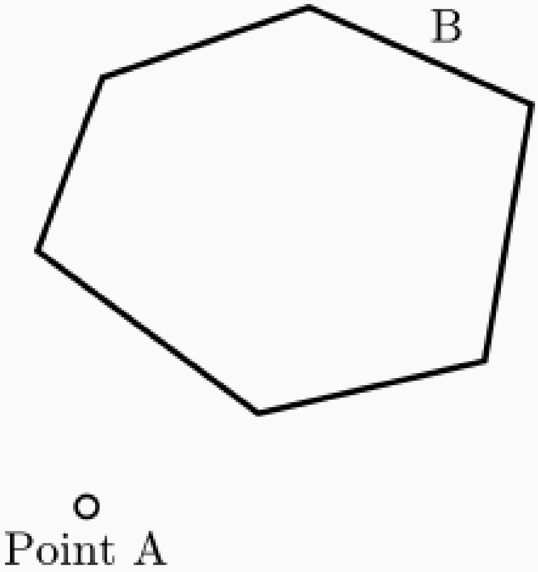
Gilbert-Johnson-Keerthi (GJK)

- Very efficient algorithm for convex collision checking
- Publication "A fast procedure for computing the distance between complex objects in three-dimensional space" (1988)

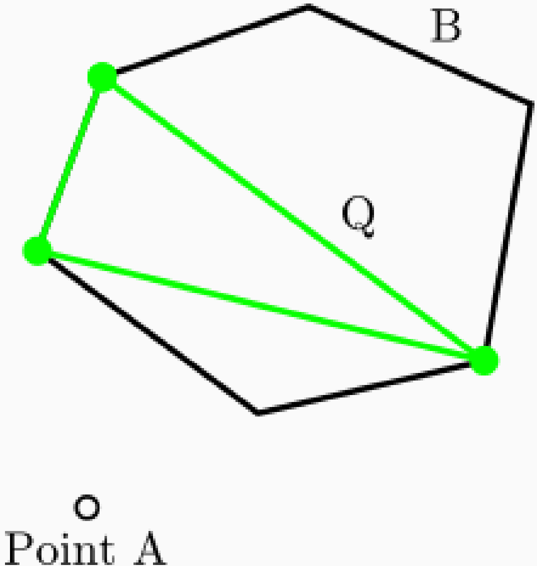
Gilbert-Johnson-Keerthi (GJK)

- Assumption: Represent objects as convex polygons
- Develop algorithm for polygon-to-point collision checking
- Reduce polygon-to-polygon checking to polygon-to-point checking

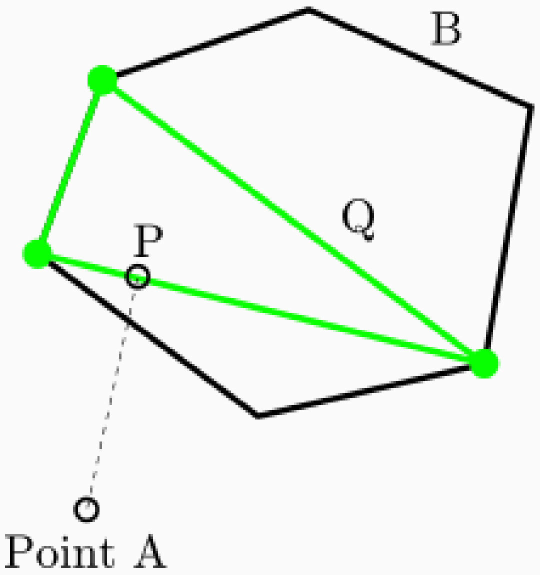
Gilbert-Johnson-Keerthi (1): Point to Polygon



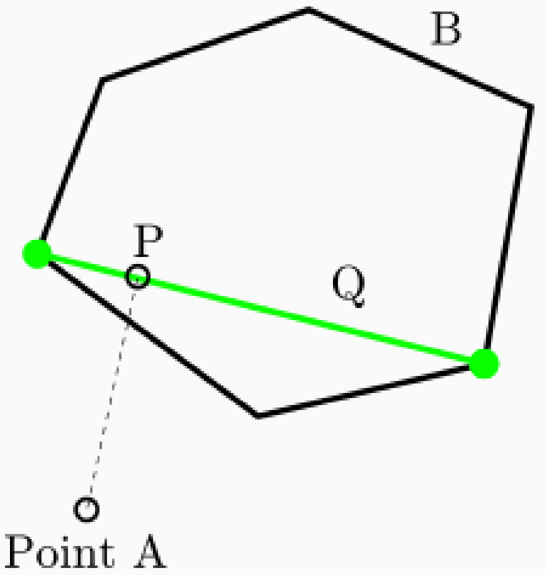
Gilbert-Johnson-Keerthi (2): Initialize simplex set Q with $(d + 1)$ vertices (d is number of dimensions).



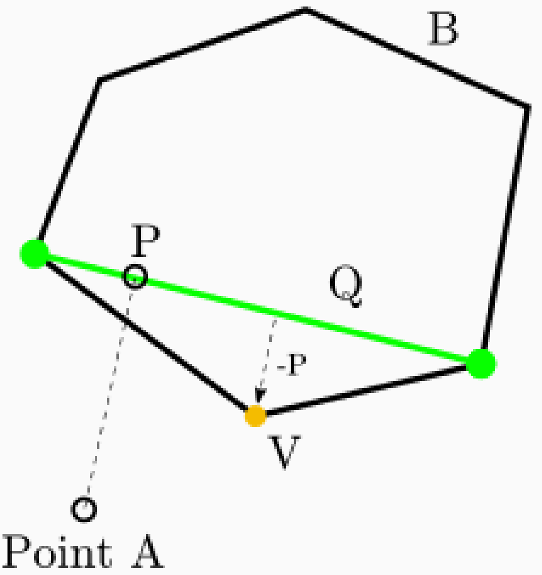
Gilbert-Johnson-Keerthi (3): Compute minimum norm point P on Q.



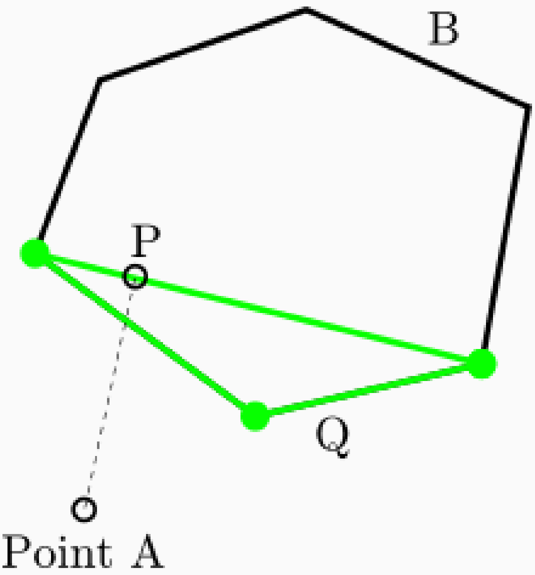
Gilbert-Johnson-Keerthi (4): Reduce Q to minimal set incuding P



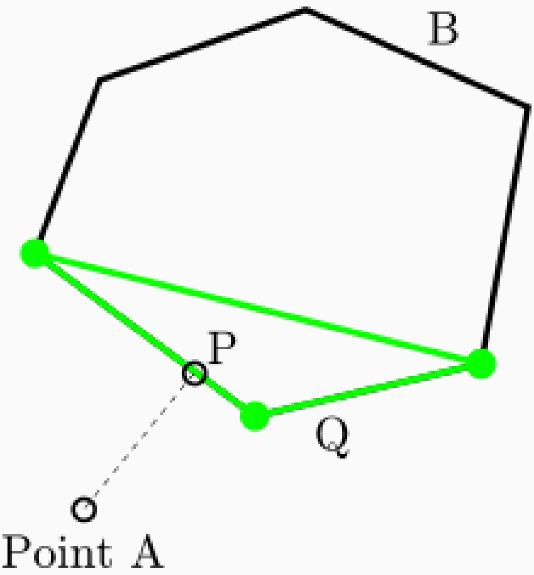
Gilbert-Johnson-Keerthi (5): Find vertex V with largest dot product in $-P$ direction.



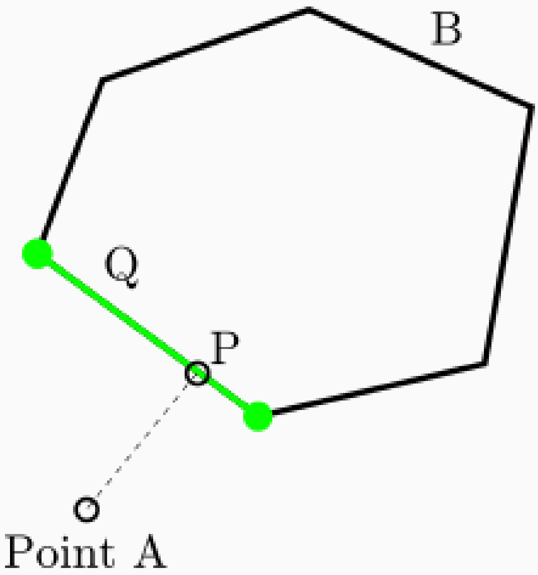
Gilbert-Johnson-Keerthi (6): Create new simplex Q.



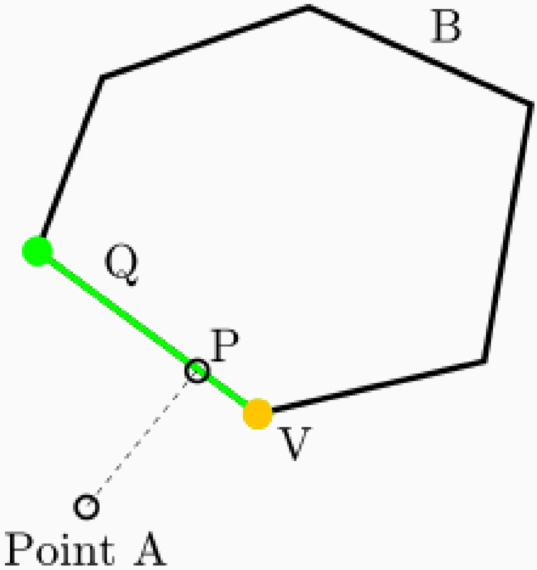
Gilbert-Johnson-Keerthi (7): Compute minimum norm point P to A.



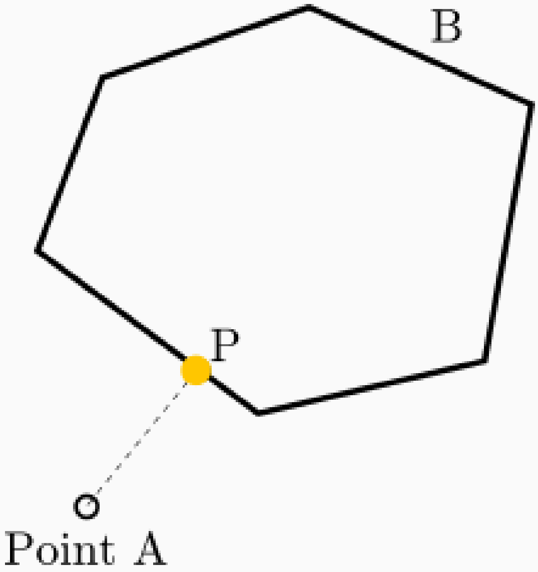
Gilbert-Johnson-Keerthi (8): Reduce Q.



Gilbert-Johnson-Keerthi (9): Find vertex V.

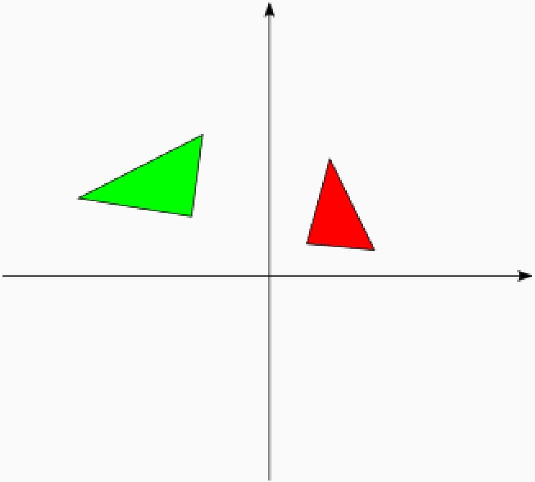


Gilbert-Johnson-Keerthi (10): If V does not improve into direction $-P$, return P.



Gilbert-Johnson-Keerthi: Polygon to Polygon

How to use this to compute polygon-to-polygon collisions?



Gilbert-Johnson-Keerthi: Polygon to Polygon

Fundamental idea

Two polygons collide if their Minkowski difference contains the origin

Two questions:

- What is the Minkowski difference?
- Why does it have to contain the origin?

Gilbert-Johnson-Keerthi: Minkowski difference

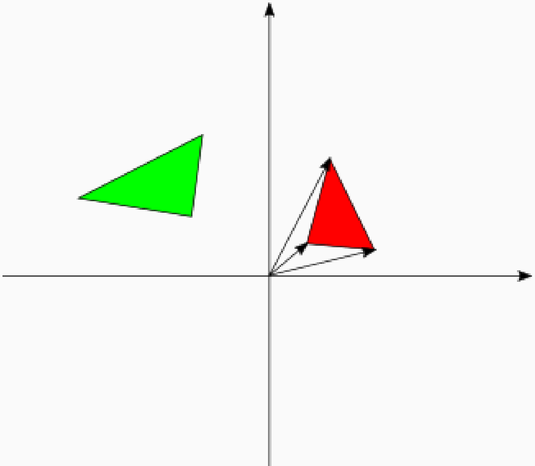
Minkowski Difference

Minkowski Difference: $A \ominus B = \{a - b \mid a \in A, b \in B\}$.

Gilbert-Johnson-Keerthi: Polygon to Polygon

Minkowski Difference

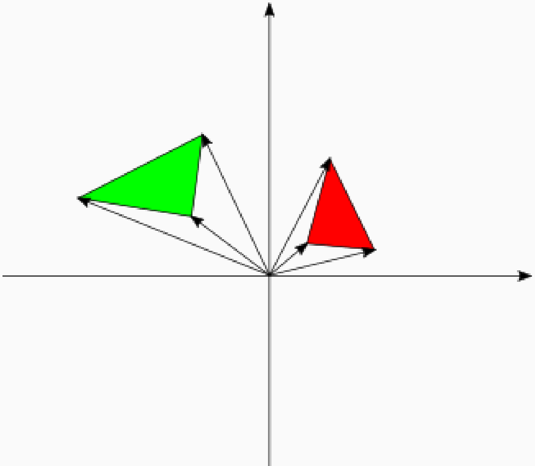
Minkowski Difference: $A \ominus B = \{a - b \mid a \in A, b \in B\}$.



Gilbert-Johnson-Keerthi: Polygon to Polygon

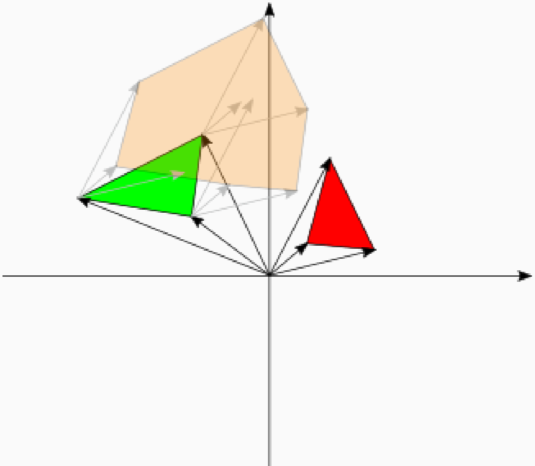
Minkowski Difference

Minkowski Difference: $A \ominus B = \{a - b \mid a \in A, b \in B\}$.



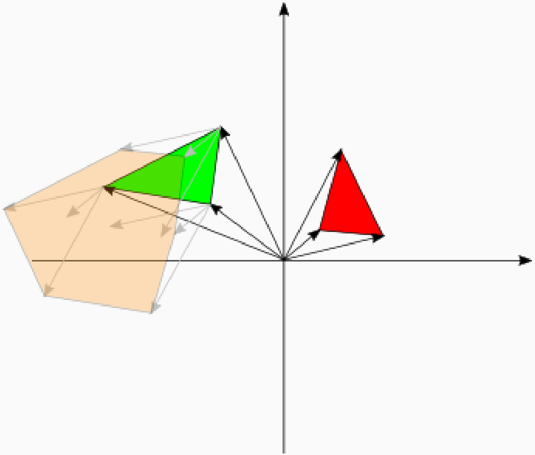
Minkowski Difference

Minkowski Difference: $A \ominus B = \{a - b \mid a \in A, b \in B\}$.



Minkowski Difference

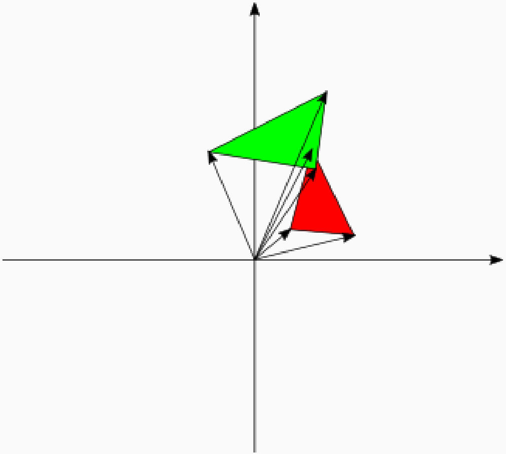
Minkowski Difference: $A \ominus B = \{a - b \mid a \in A, b \in B\}$.



Gilbert-Johnson-Keerthi: Polygon to Polygon

Origin inside Minkowski Difference

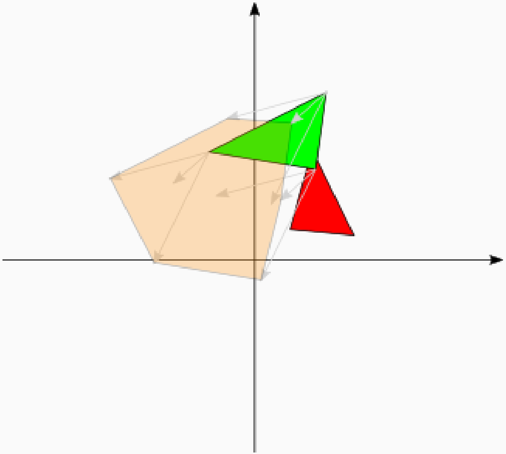
If origin is inside $A \ominus B$ then A and B share a point.



Gilbert-Johnson-Keerthi: Polygon to Polygon

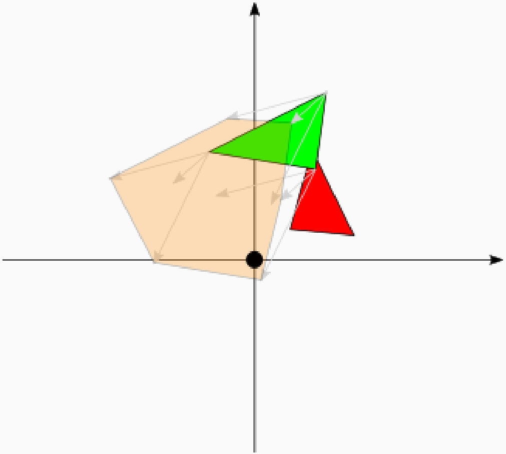
Origin inside Minkowski Difference

If origin is inside $A \ominus B$ then A and B share a point.



Origin inside Minkowski Difference

If origin is inside $A \ominus B$ then A and B share a point.



Reduction

This means: Polygon-to-polygon problem is reduced to Point-to-Polygon

- Whereby the point is the origin
- And the polygon is the Minkowski difference

Open source collision library:

https://github.com/flexible-collision-library/fcl

Flexible Collision Library

```
// Given two objects o1 and o2
CollisionObject* o1;
CollisionObject* o2;
DistanceRequest request;
DistanceResult result;
distance(o1, o2, request, result);
```

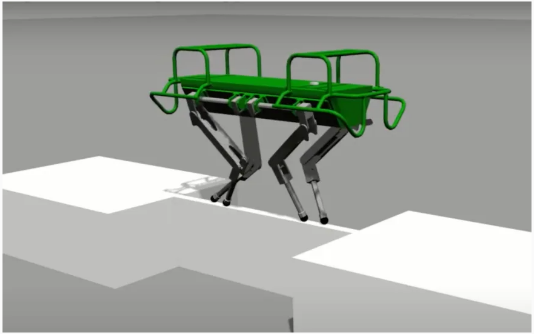
Useful Links

- High-level introduction to collision detection
  https://en.wikipedia.org/wiki/Collision_detection
- Larger list of possible bounding volumes
  https://en.wikipedia.org/wiki/Bounding_volume
- Description of GJK https://slideplayer.com/slide/689954/
- Good video on GJK in 2D https://www.youtube.com/watch?v=ajv46BSqcK4

Constraints

Stability constraint

Static stability



Static stability

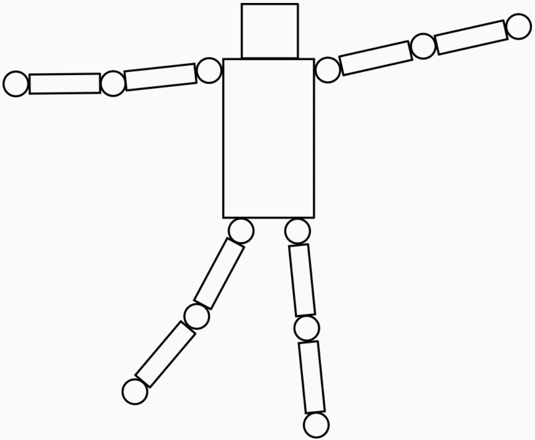
Definition of static equilibrium

- Robot body is at rest (sum of forces acting on robot is zero)

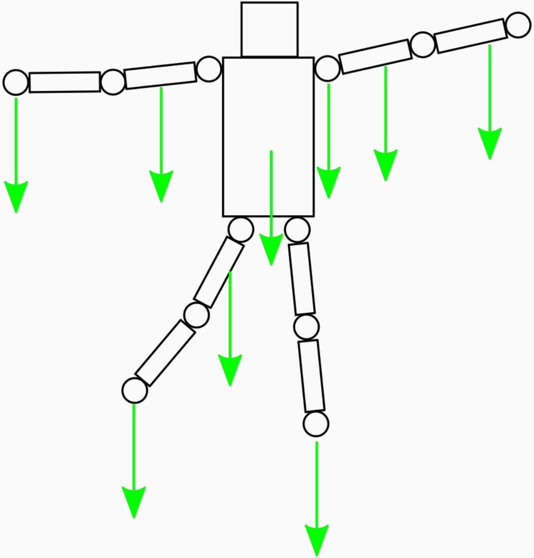
As Constraint Function

Function `IsStaticallyStable(q)`: $\mathcal{Q} \rightarrow \{\text{True}, \text{False}\}$ Is robot at configuration q in static equilibrium?

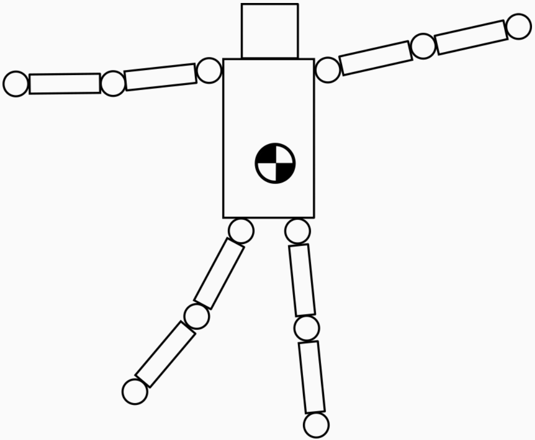
Force-Torque Analysis



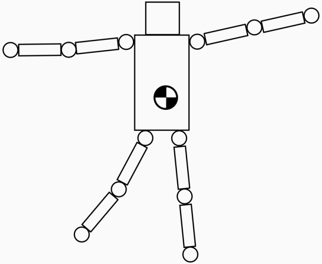
Force-Torque Analysis



Force-Torque Analysis



Force-Torque Analysis

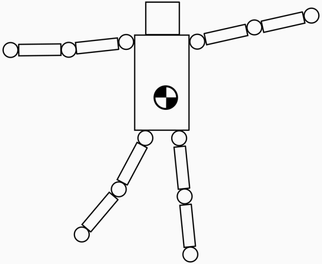


Center Of Mass (CoM)

Mean location of a distribution of mass in space.

$$\text{CoM} = \frac{\sum r_i \cdot m_i}{\sum m_i}$$

Force-Torque Analysis

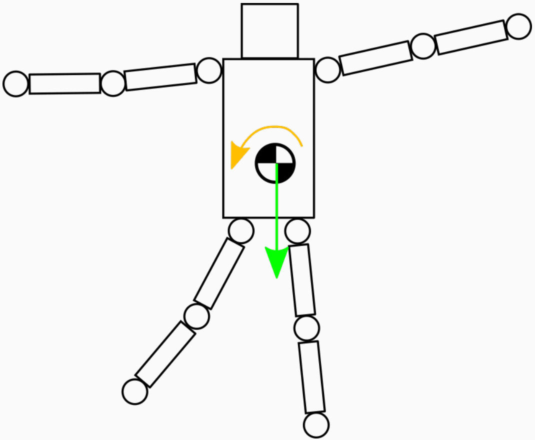


Center Of Mass

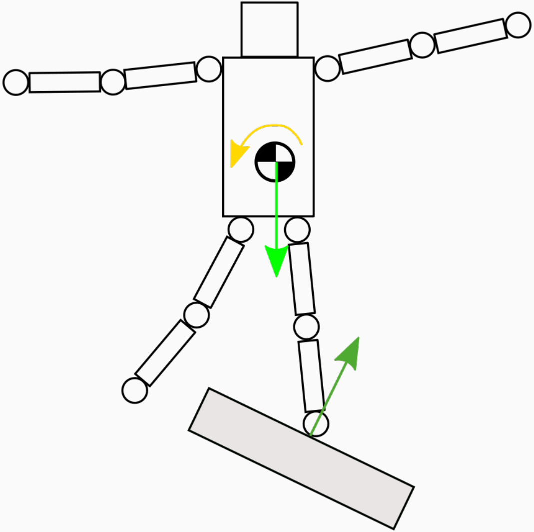
If you support the CoM, the robot does not tip over.

If you add counter forces to all forces centered at CoM, the robot does not tip over.

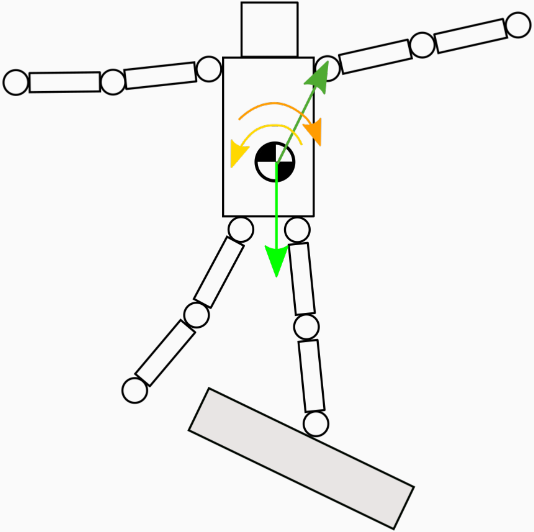
Force-Torque Analysis



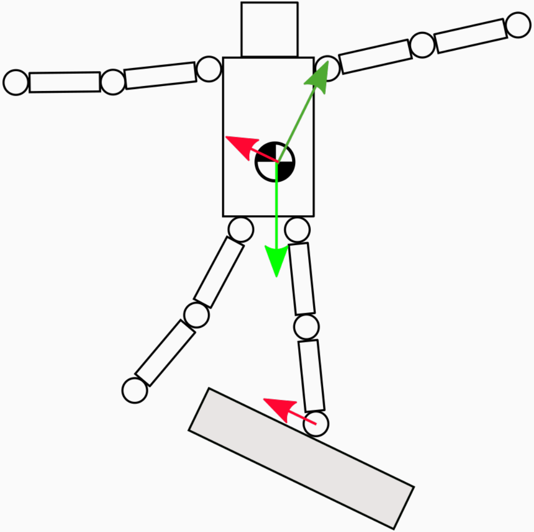
Force-Torque Analysis (Ground Reaction Force)



Force-Torque Analysis



Force-Torque Analysis (Friction)



Static stability

Computing Stability through Force-Torque Analysis

When is a robot stable?

- Assume robot body is at configuration $q \in \mathcal{Q}$
- Approximate robot as a *single* rigid body
- Compute all forces and torques acting on robot body
- Sum them up relative to center of mass
- If the sum is zero, the robot is statically stable

Useful Links

- "Testing Static Equilibrium for Legged Robots" by Timothy Bretl (2008) https://lall.stanford.edu/papers/bretl_eqmcut_ieee_tro_projection_2008_08_01_01/pubdata/entry.pdf

Summary Lecture 2

Recap

- Big idea: Modeling robots as a point in a configuration space
- Topology as tool to model configuration spaces
- Measuring distances $\rightarrow$ Metric spaces
- Applying constraints and having efficient evaluation functions
- GJK algorithm for collision checking of convex objects

Next Time

- Using discretization to find paths in a configuration space
- A*: finding paths optimally
- Admissible heuristics

Additional Links

- https://en.wikipedia.org/wiki/Homeomorphism
- https://en.wikipedia.org/wiki/Euclidean_group
- https://en.wikipedia.org/wiki/Metric_(mathematics)

[1] Tomás Lozano-Pérez and Michael A. Wesley. "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles". In: *Commun. ACM* 22.10 (Oct. 1979), pp. 560–570. ISSN: 0001-0782. DOI: 10.1145/359156.359164.

[2] Steven M. LaValle. *Planning algorithms*. Cambridge University Press, 2006. ISBN: 978-0-521-86205-9. URL: http://planning.cs.uiuc.edu.

[3] Dmitry Berenson. "Motion Planning: Robotics and Beyond". In: (2021). URL: https://web.eecs.umich.edu/~dmitryb/courses/winter2021motionplanning/index.html.

[4] James R Munkres. *Topology*. Vol. 2. Prentice hall Upper Saddle River, 2000.