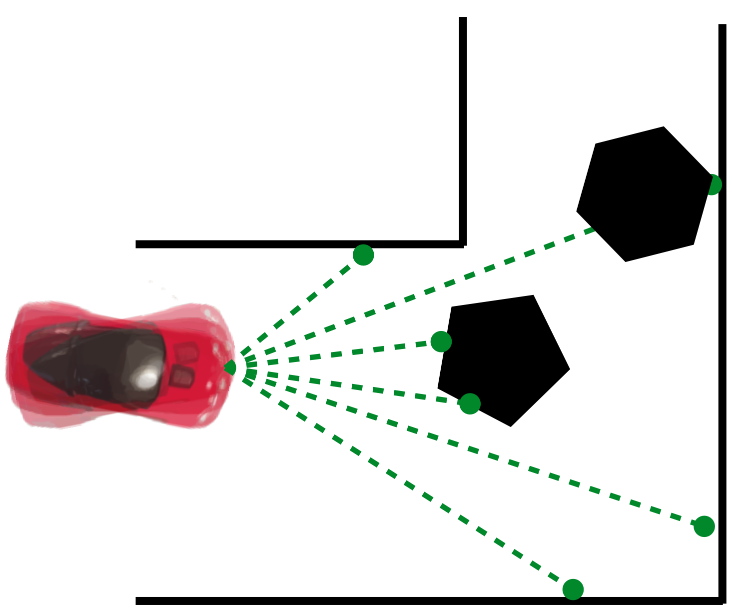
Content adapted from LaValle

Introduction to Motion Planning

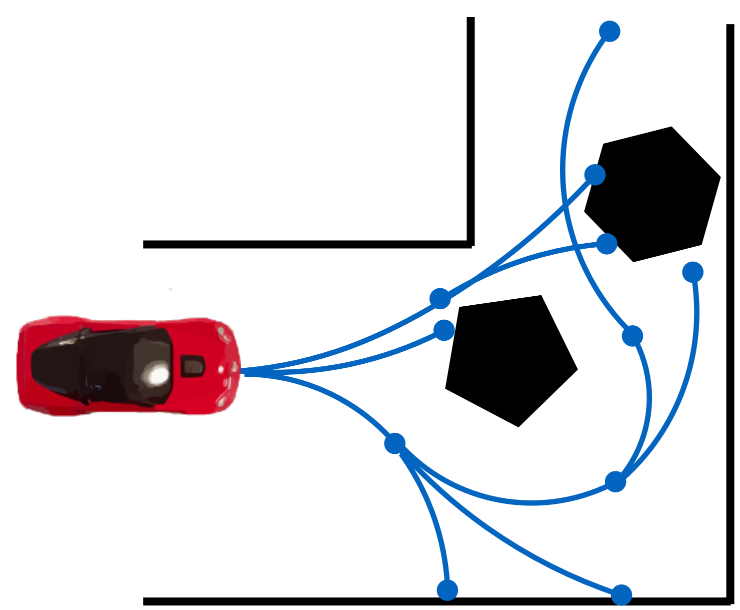
Sanjiban Choudhury

TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle

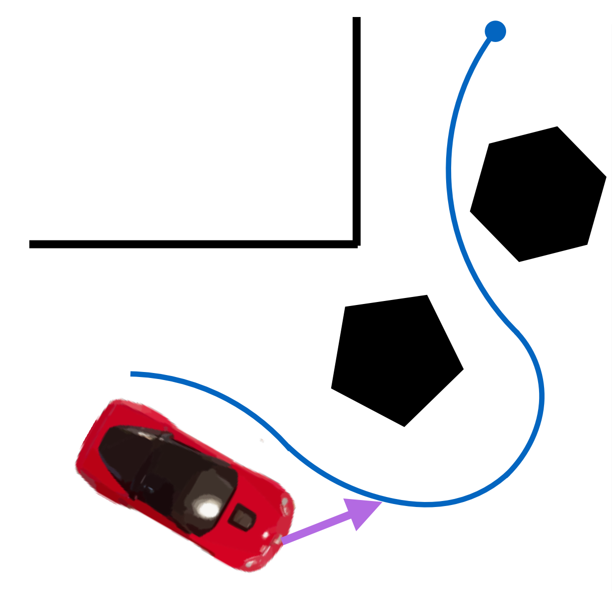
Estimate state



Plan a sequence of motions



Control robot to follow plan



A prospective grad student: “Is planning just A*?”

A prospective grad student:
“Is planning just A*?”

Challenge: Flying from Seattle to Pittsburgh?

(from Leslie Kaebling)

Challenge: Flying from Seattle to Pittsburgh?

(from Leslie Kaebling)

Piece 1: How do I get out of this classroom?

Challenge: Flying from Seattle to Pittsburgh?

(from Leslie Kaebling)

Piece 1: How do I get out of this classroom?

Piece 2: Even if we have an in-depth plan get to our terminal,
and some idea how to check-in and board plane,
do you
bother to plan your path through Pittsburgh terminal?

Challenge: Flying from Seattle to Pittsburgh?

(from Leslie Kaebling)

Piece 1: How do I get out of this classroom?

Piece 2: Even if we have an in-depth plan get to our terminal,
and some idea how to check-in and board plane,
do you
bother to plan your path through Pittsburgh terminal?

Piece 3: What if you wanted a rental car?
That's something you have to plan in advance right?

Challenge: Flying from Seattle to Pittsburgh?

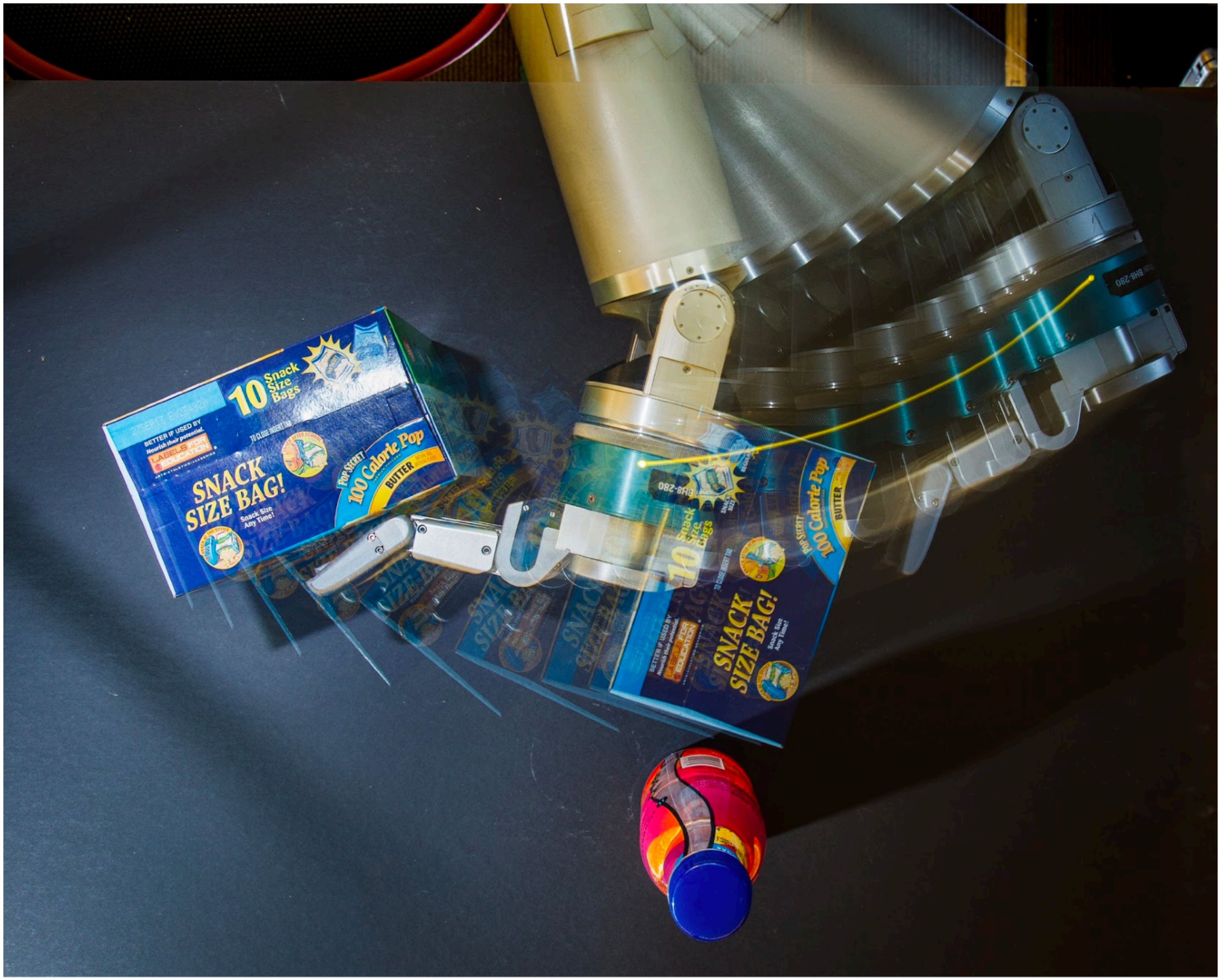
(from Leslie Kaebling)

Piece 1: How do I get out of this classroom?

Piece 2: Even if we have an in-depth plan get to our terminal,
and some idea how to check-in and board plane,
do you
bother to plan your path through Pittsburgh terminal?

Piece 3: What if you wanted a rental car?
That's something you have to plan in advance right?

Motion Planning



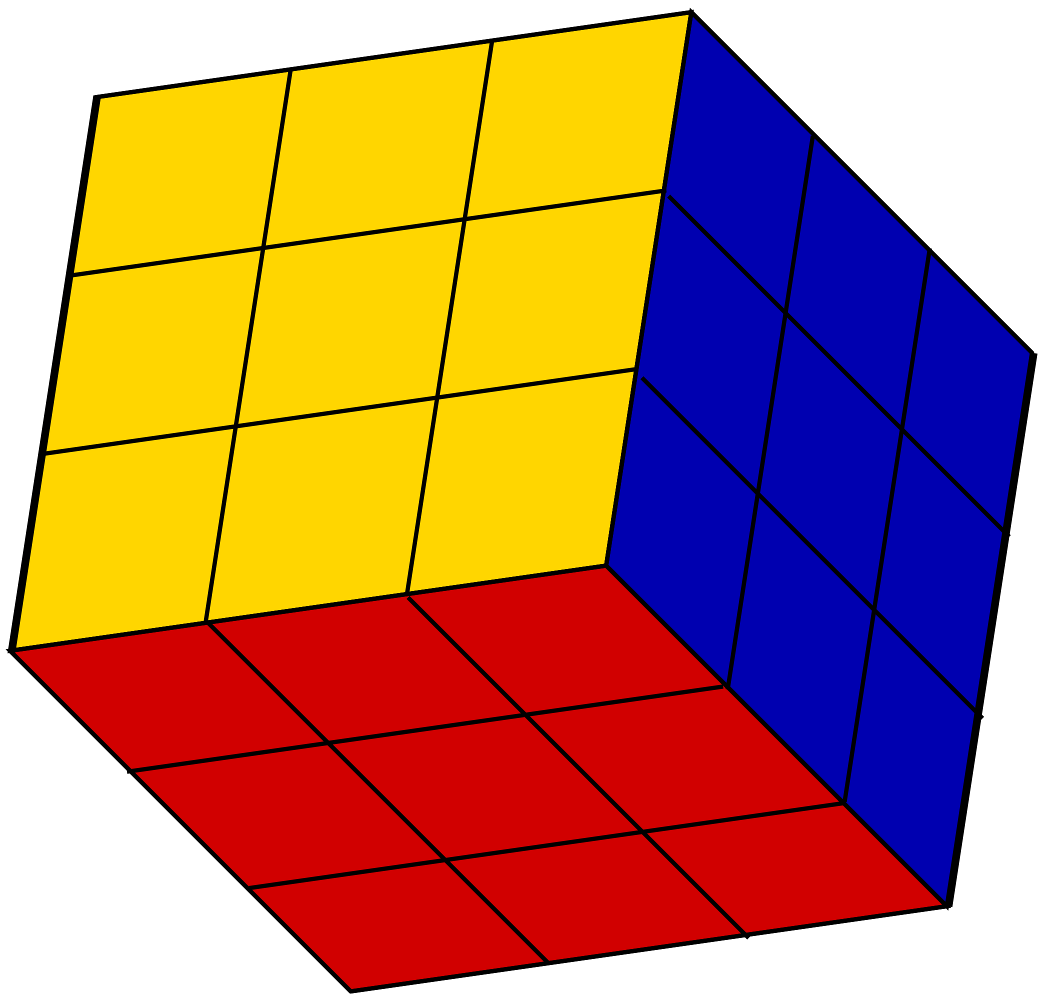
Today's objective

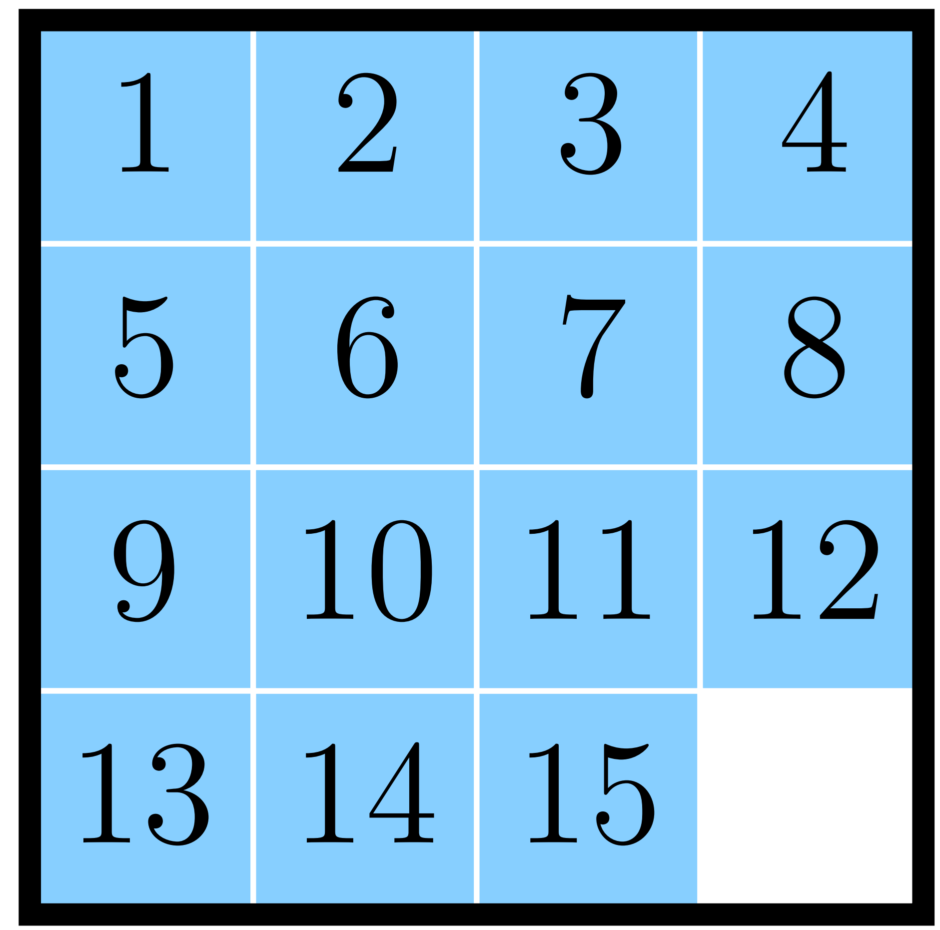
1. Broad scope and challenges in motion planning

2. Formalizing motion planning

3. Hardness of planning, extensions to differential constraints

Games





Discrete Feasible Planning

1. A nonempty *state space* X, which is a finite or countably infinite set of *states*.

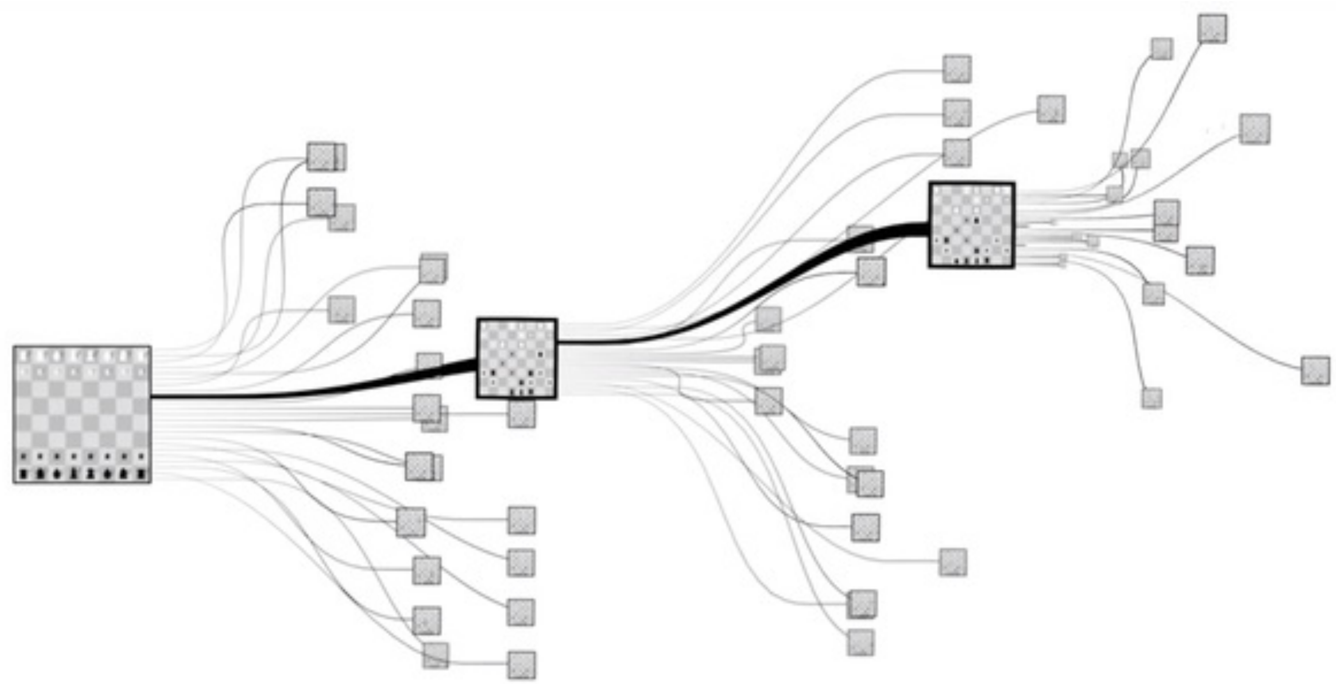
2. For each state $x \in X$, a finite *action space* $U(x)$.

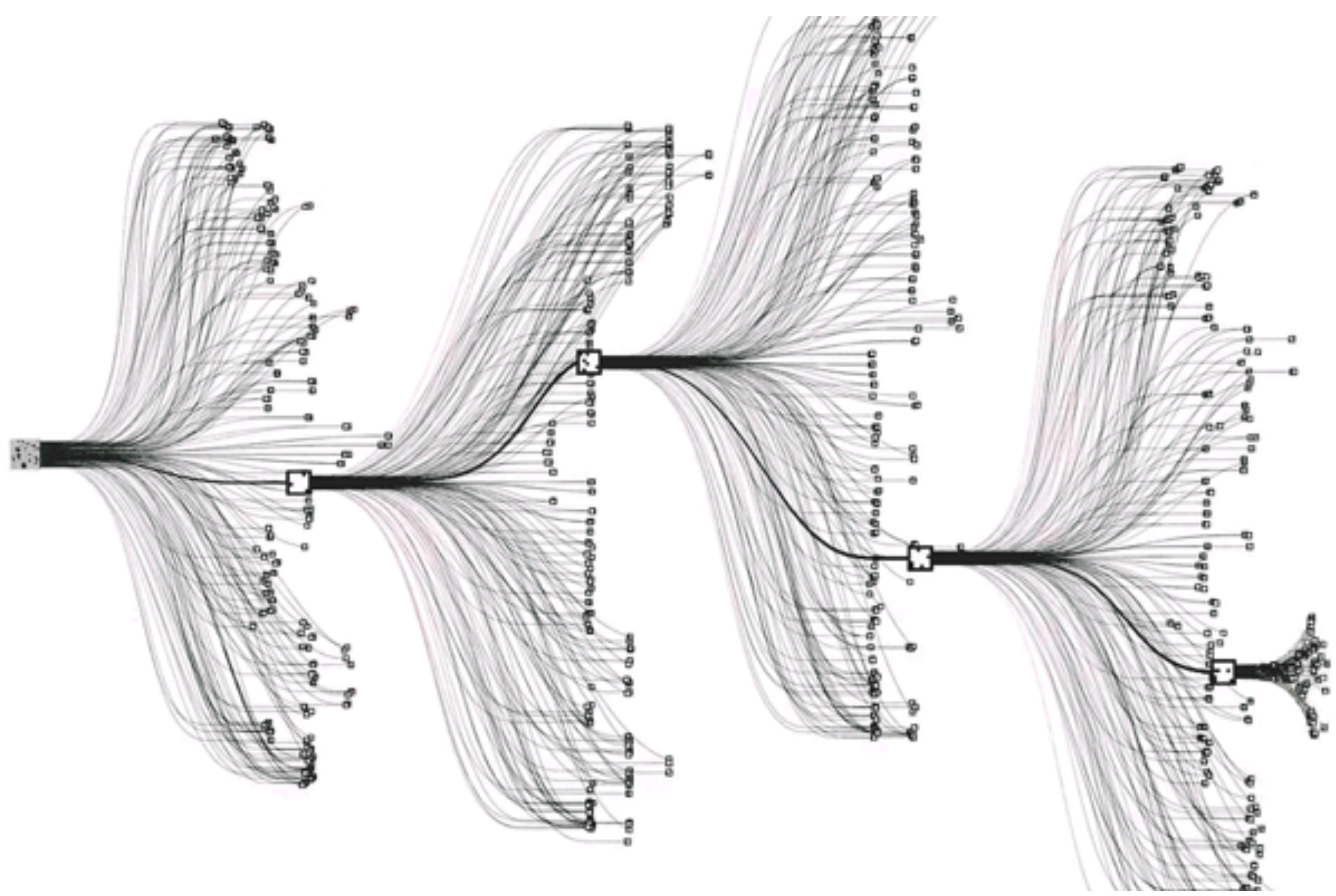
3. A *state transition function* f that produces a state $f(x, u) \in X$ for every $x \in X$ and $u \in U(x)$. The *state transition equation* is derived from f as $x' = f(x, u)$.

4. An *initial state* $x_I \in X$.

5. A *goal set* $X_G \subset X$.

Mastering the game of Go





Why it's not straight forward to extend?

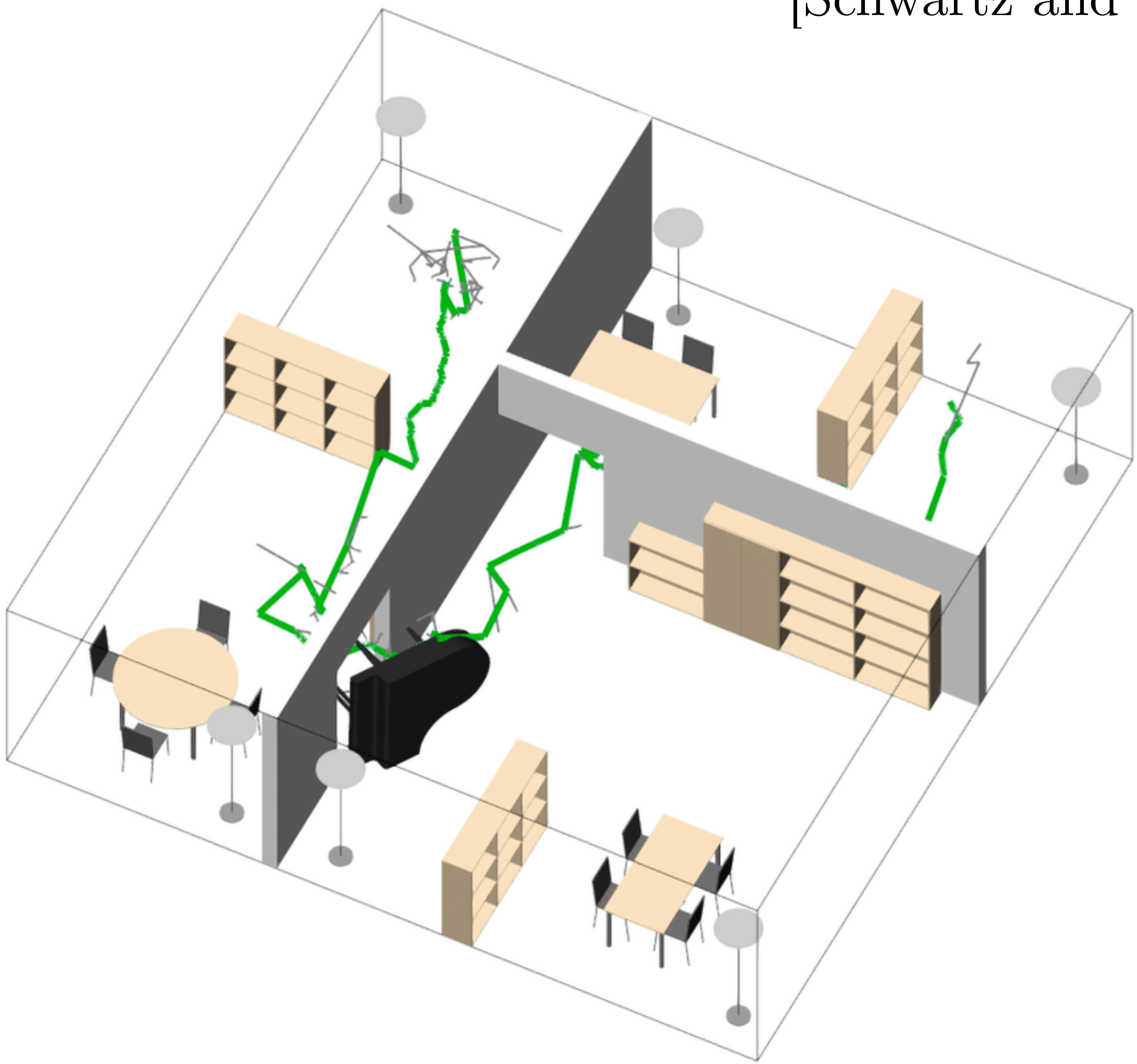
Discrete state space -
no recipe for going to continuous state action space

Easy to simulate moves -
no expensive physics / geometric computation

Rules of game already known -
no notion of model uncertainty

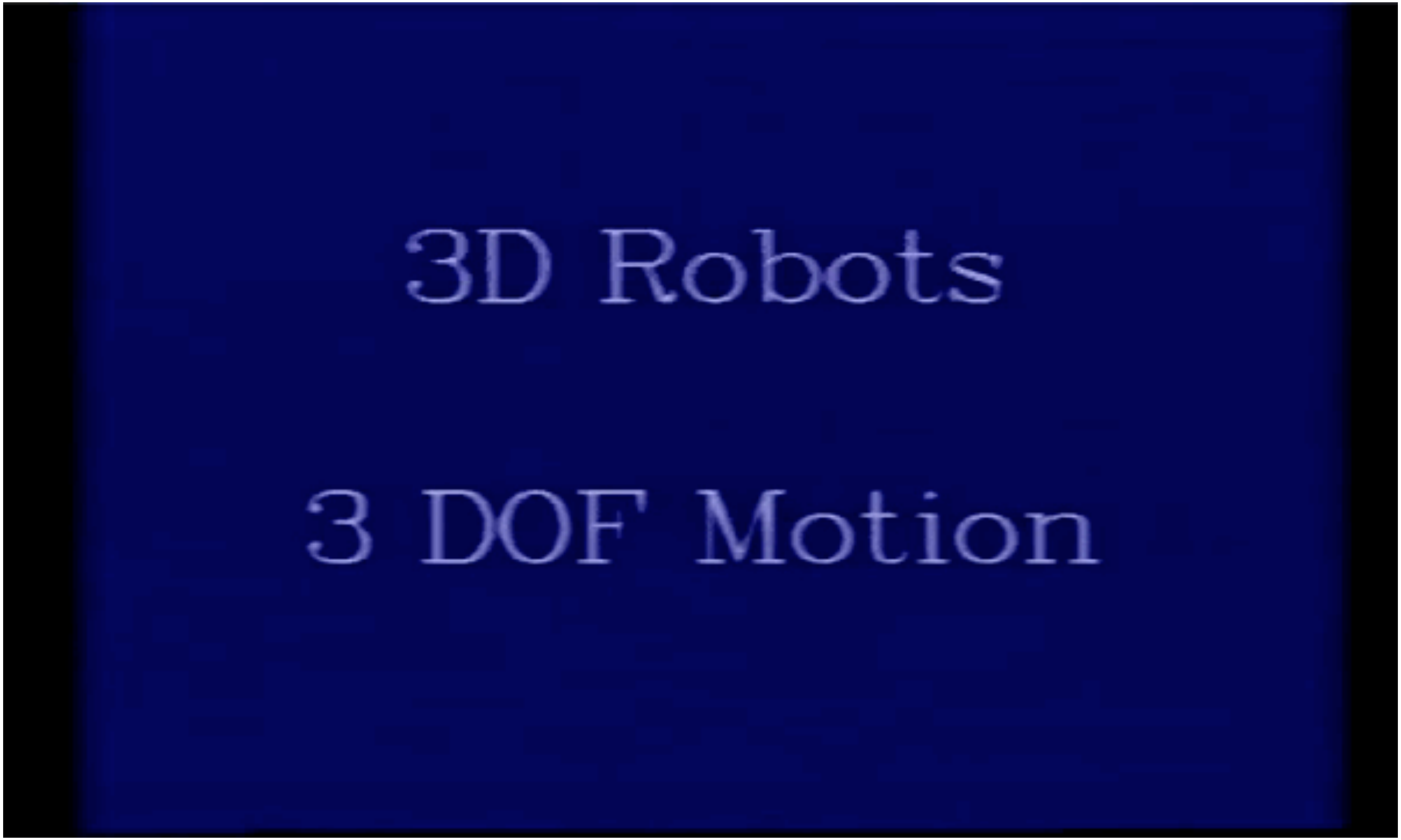
The Piano Mover's Problem

[Schwartz and Sharir, '83]



1990s!

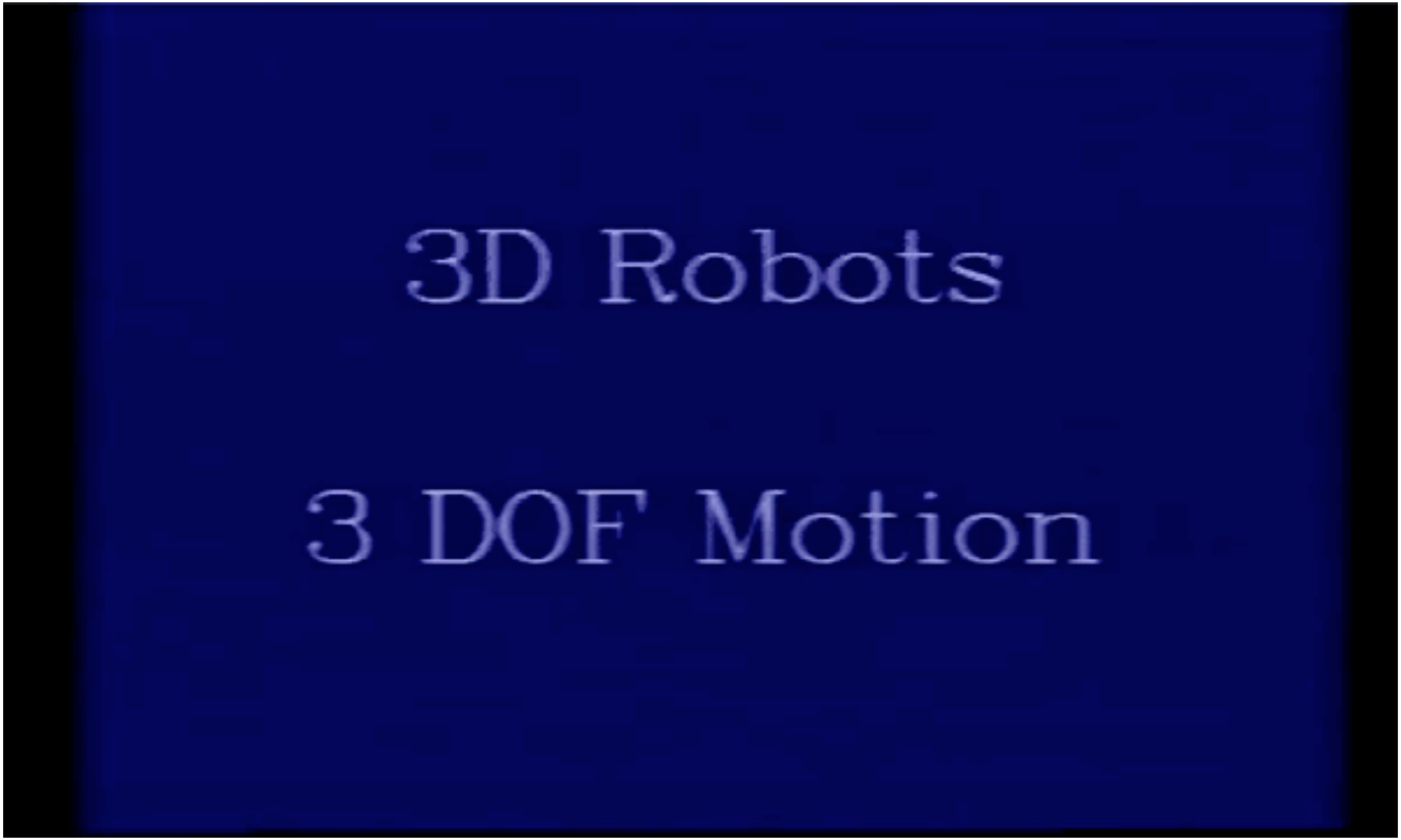
(Bruce Donald)



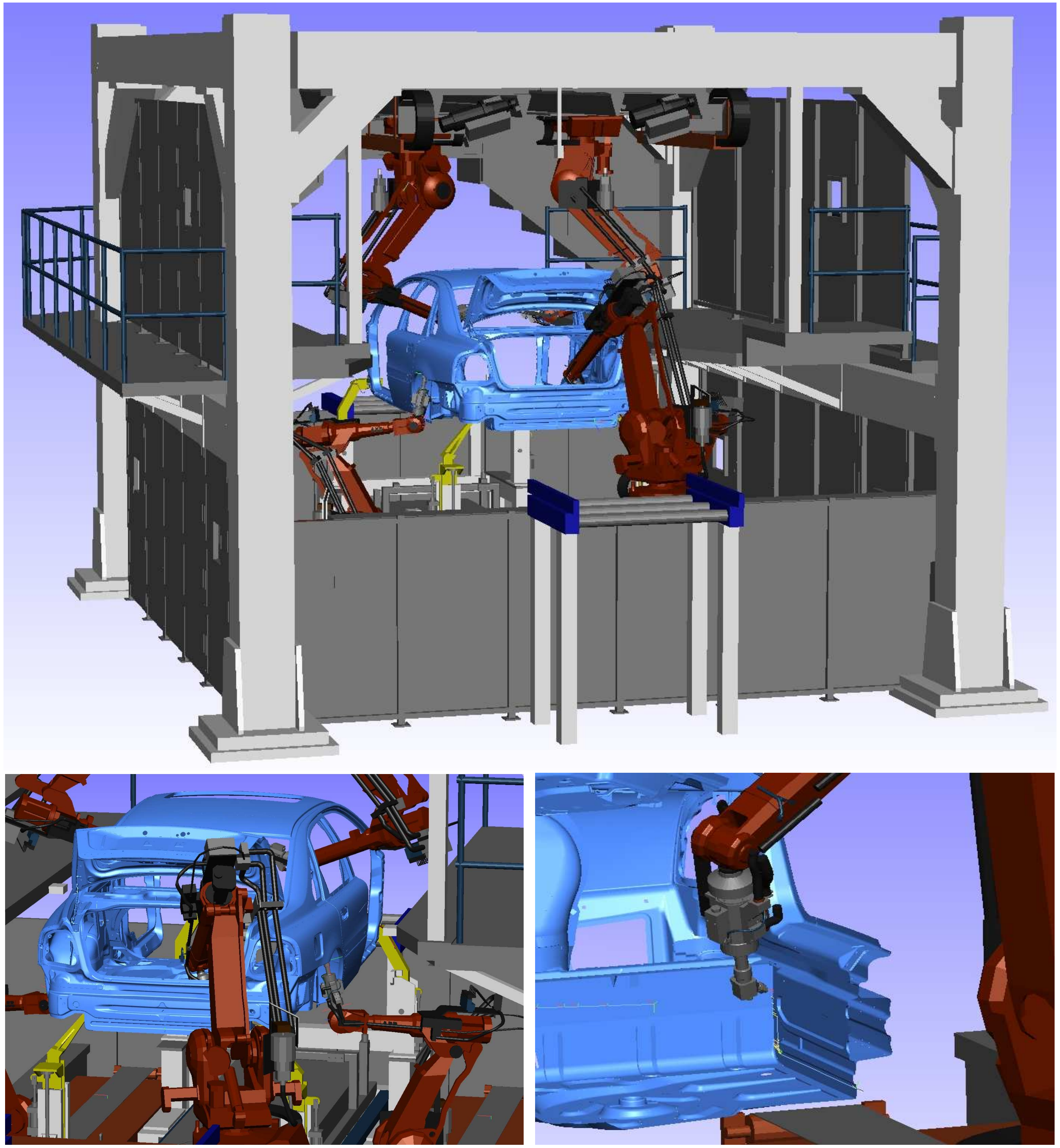
https://www.youtube.com/watch?v=UBAGTsnzAbk

1990s!

(Bruce Donald)



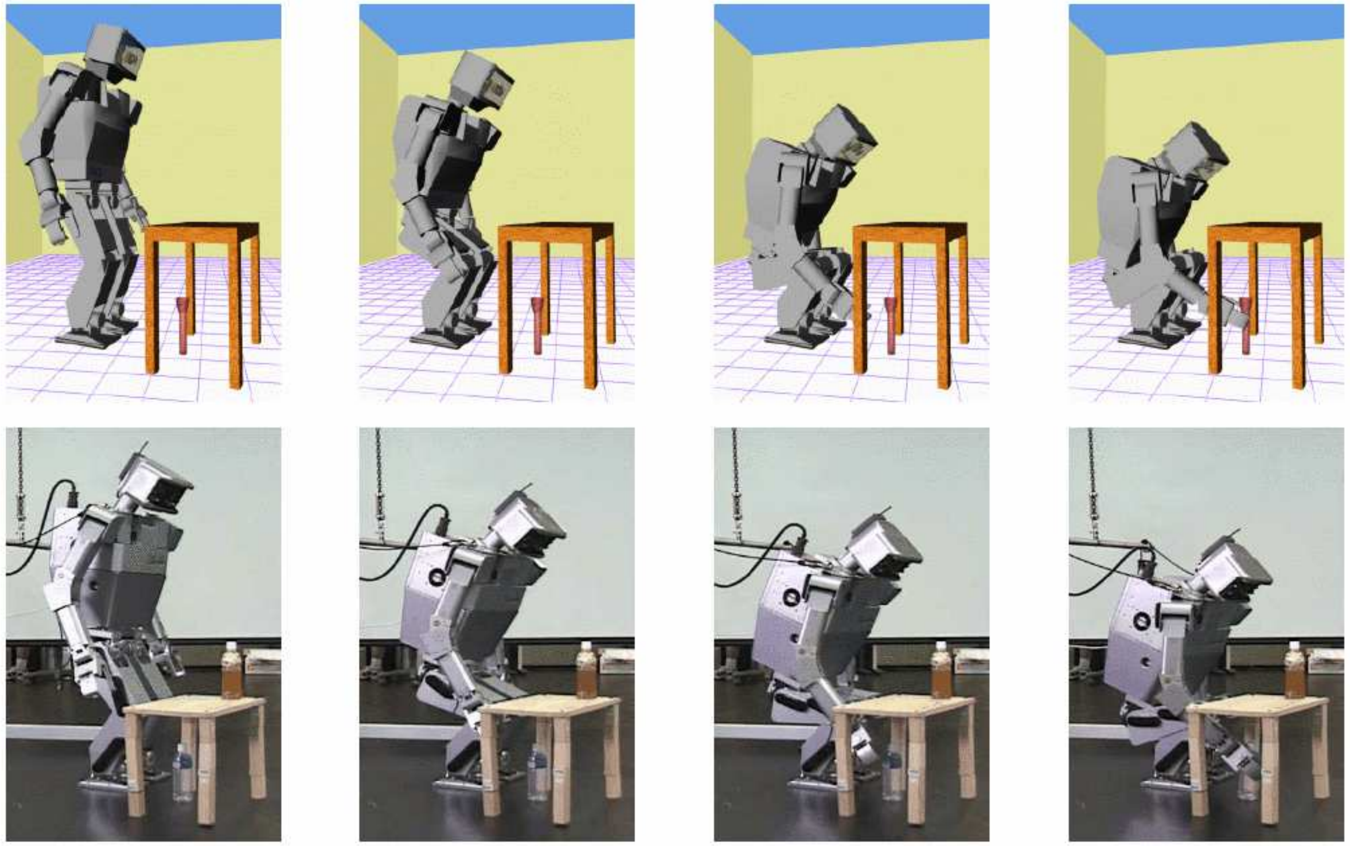
https://www.youtube.com/watch?v=UBAGTsnzAbk



Volvo Cars plant in Sweden (courtesy of Volvo Cars and FCC)

High dimension planning

(Lau and Kuffner, 2005)



Honda H7

(Kuffner, 2003)

Real-time planning

Willow garage, 2009

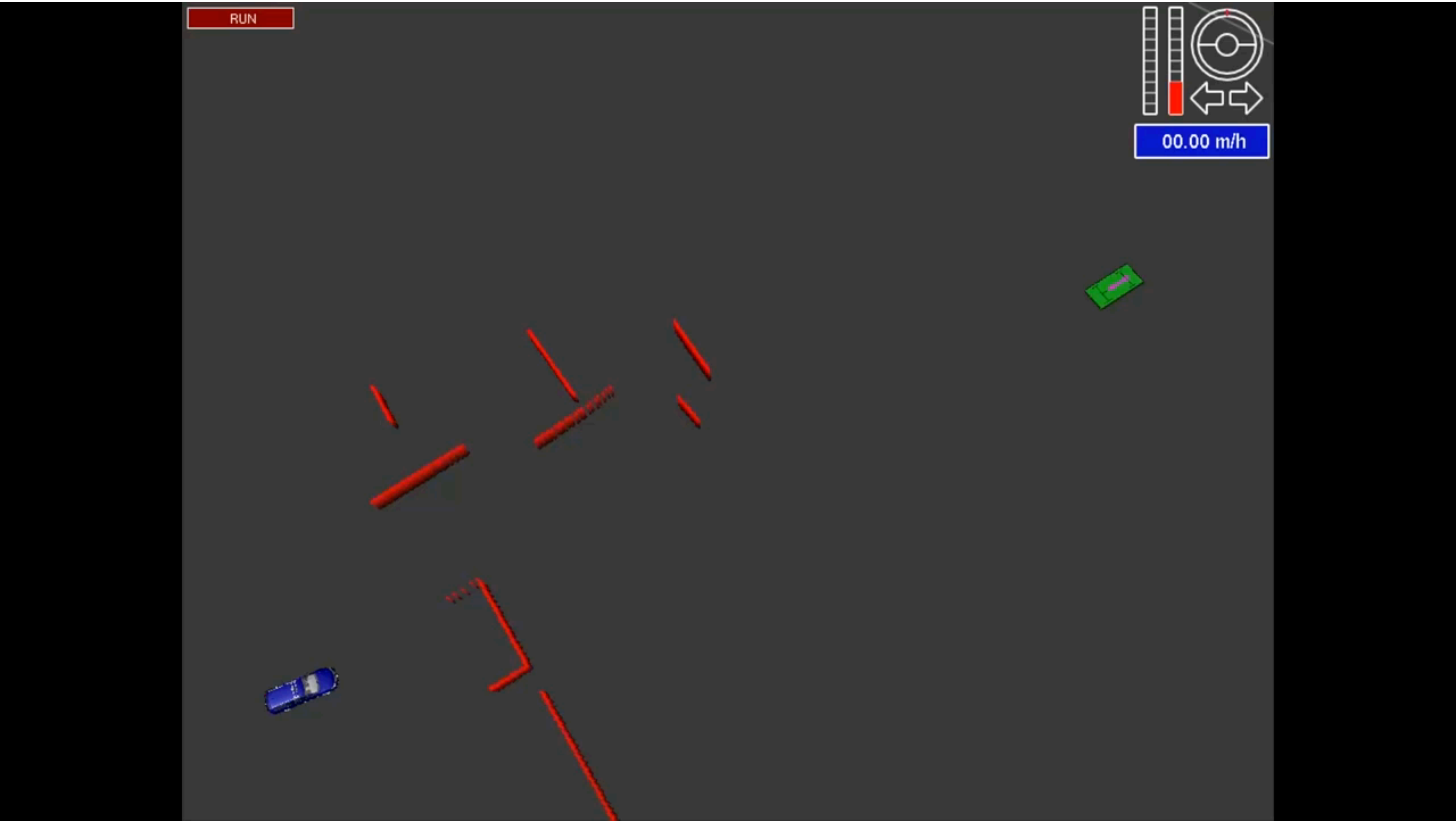
https://www.youtube.com/watch?v=qbQDJ1c_Nxk&feature=youtu.be

Real-time planning

Willow garage, 2009

https://www.youtube.com/watch?v=qbQDJ1c_Nxk&feature=youtu.be

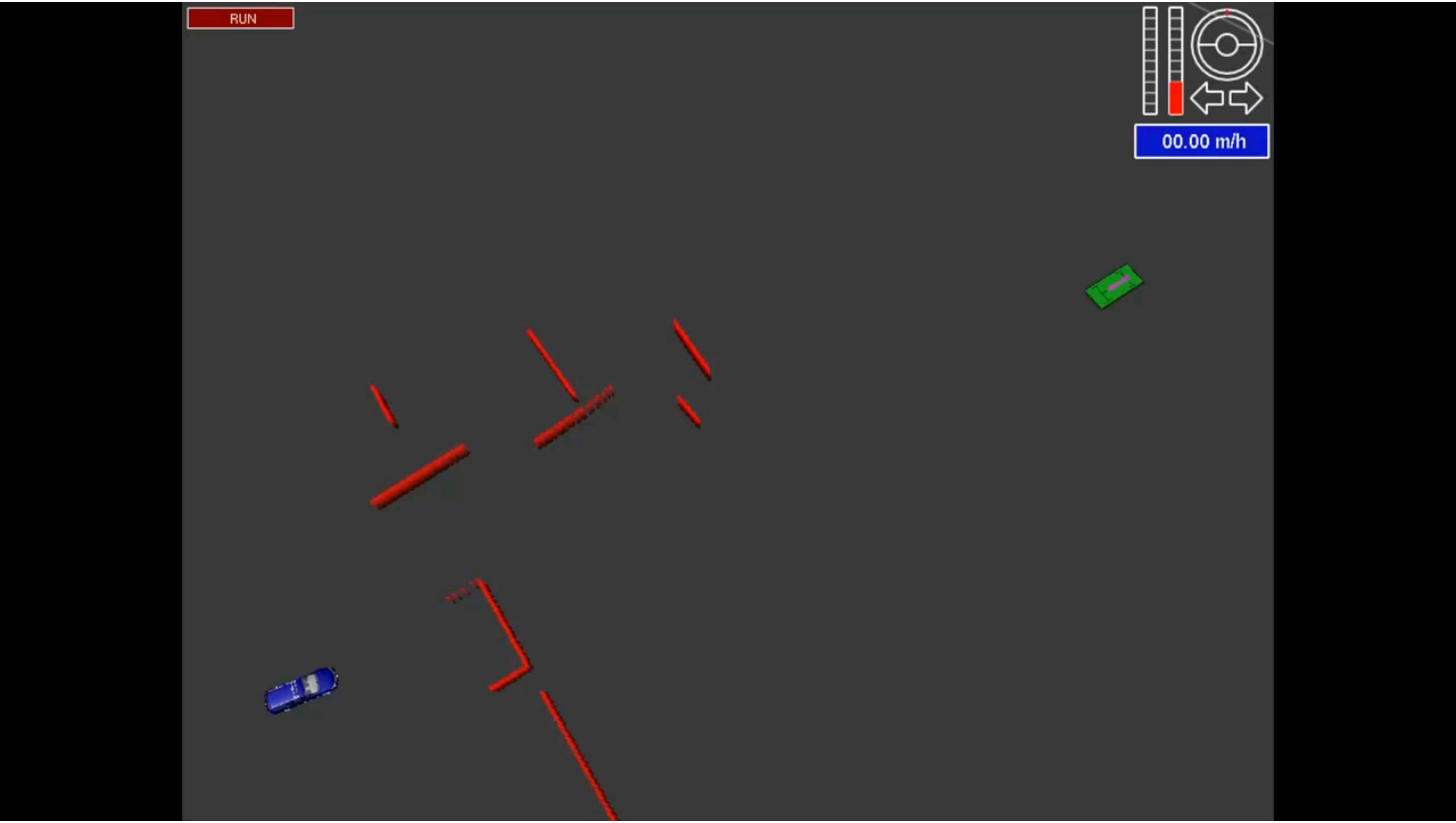
Real-time planning



Stanford DARPA Challenge, 2007

https://www.youtube.com/watch?v=qXZt-B7iUyw&feature=youtu.be

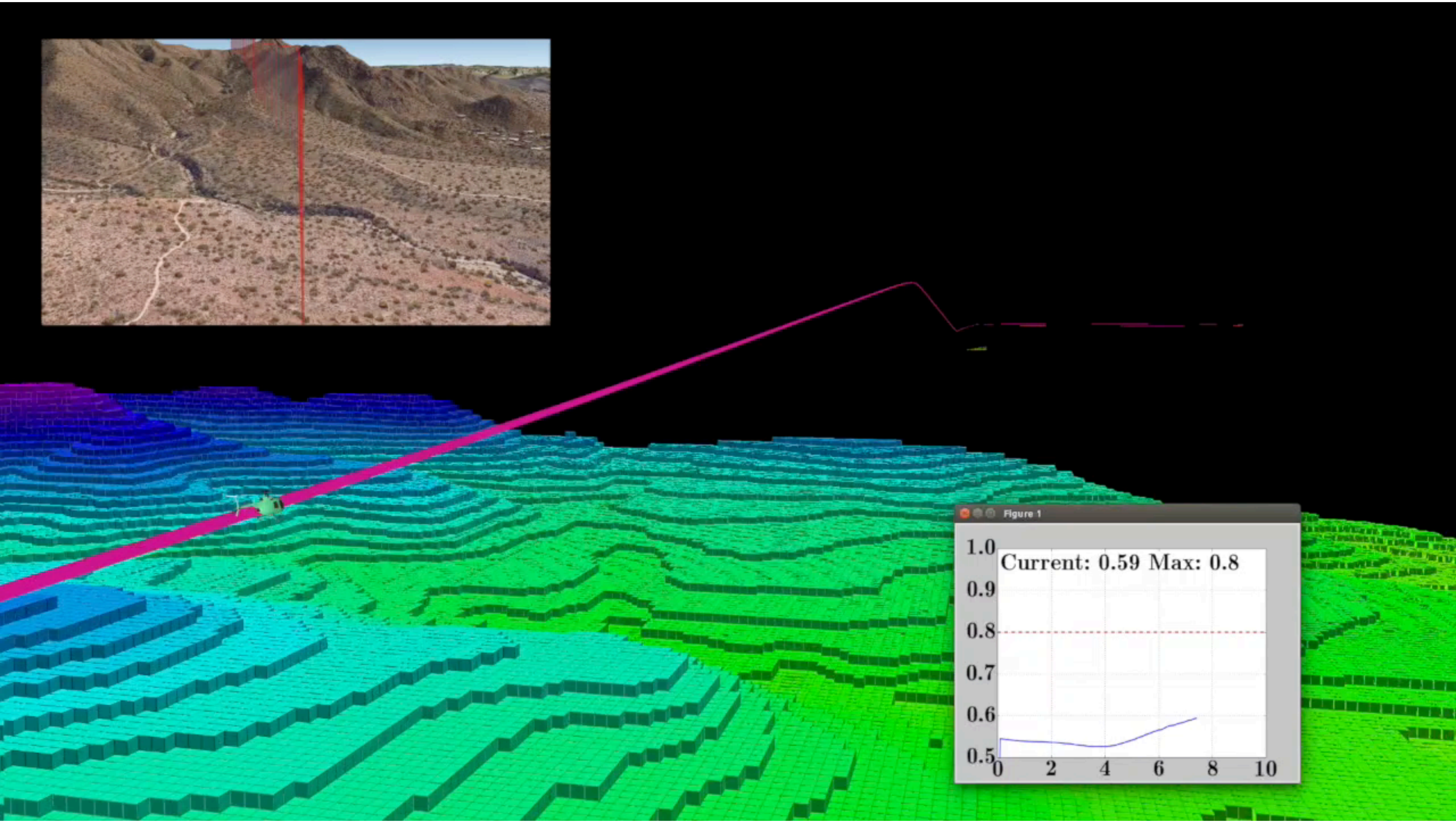
Real-time planning



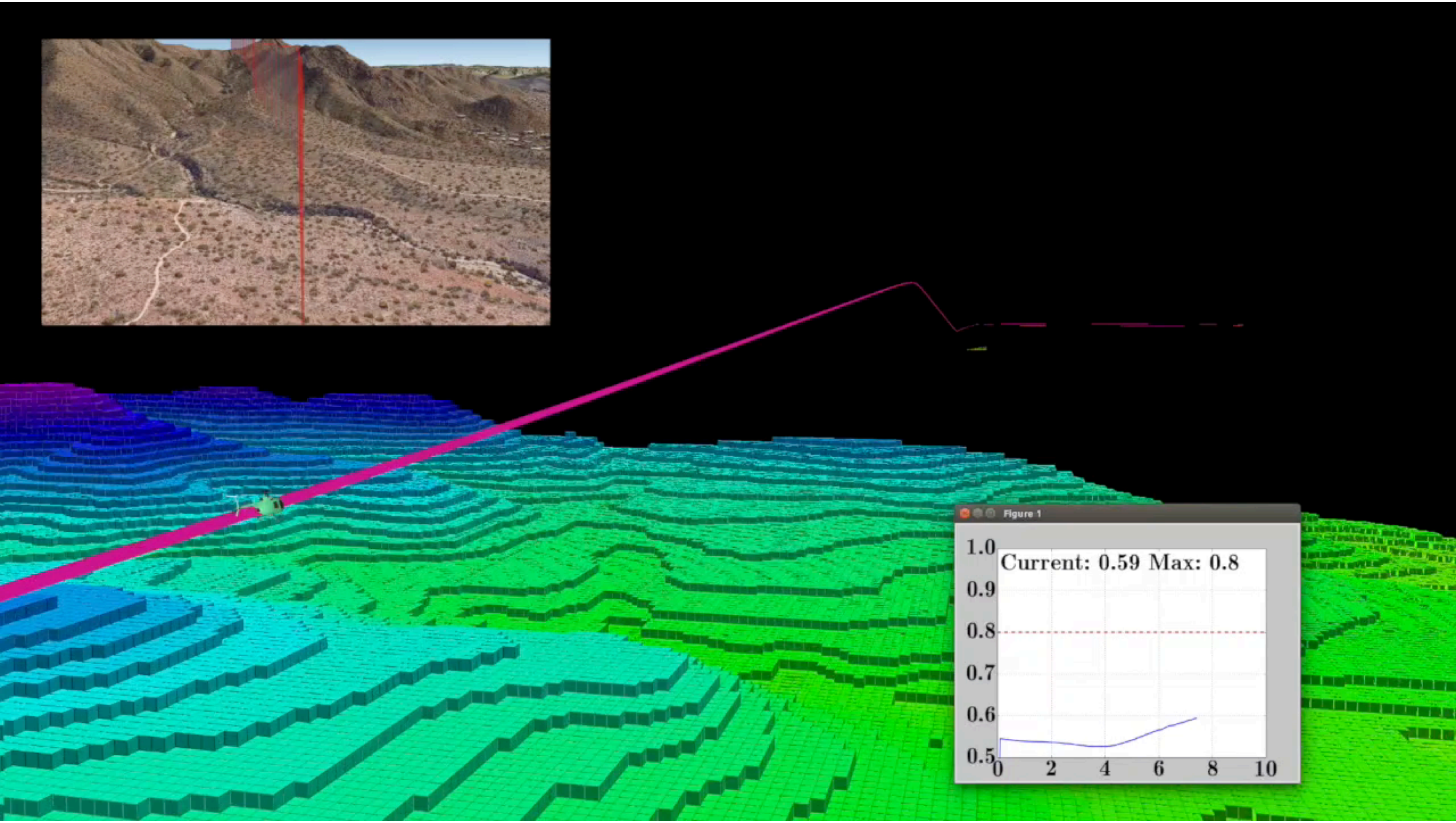
Stanford DARPA Challenge, 2007

https://www.youtube.com/watch?v=qXZt-B7iUyw&feature=youtu.be

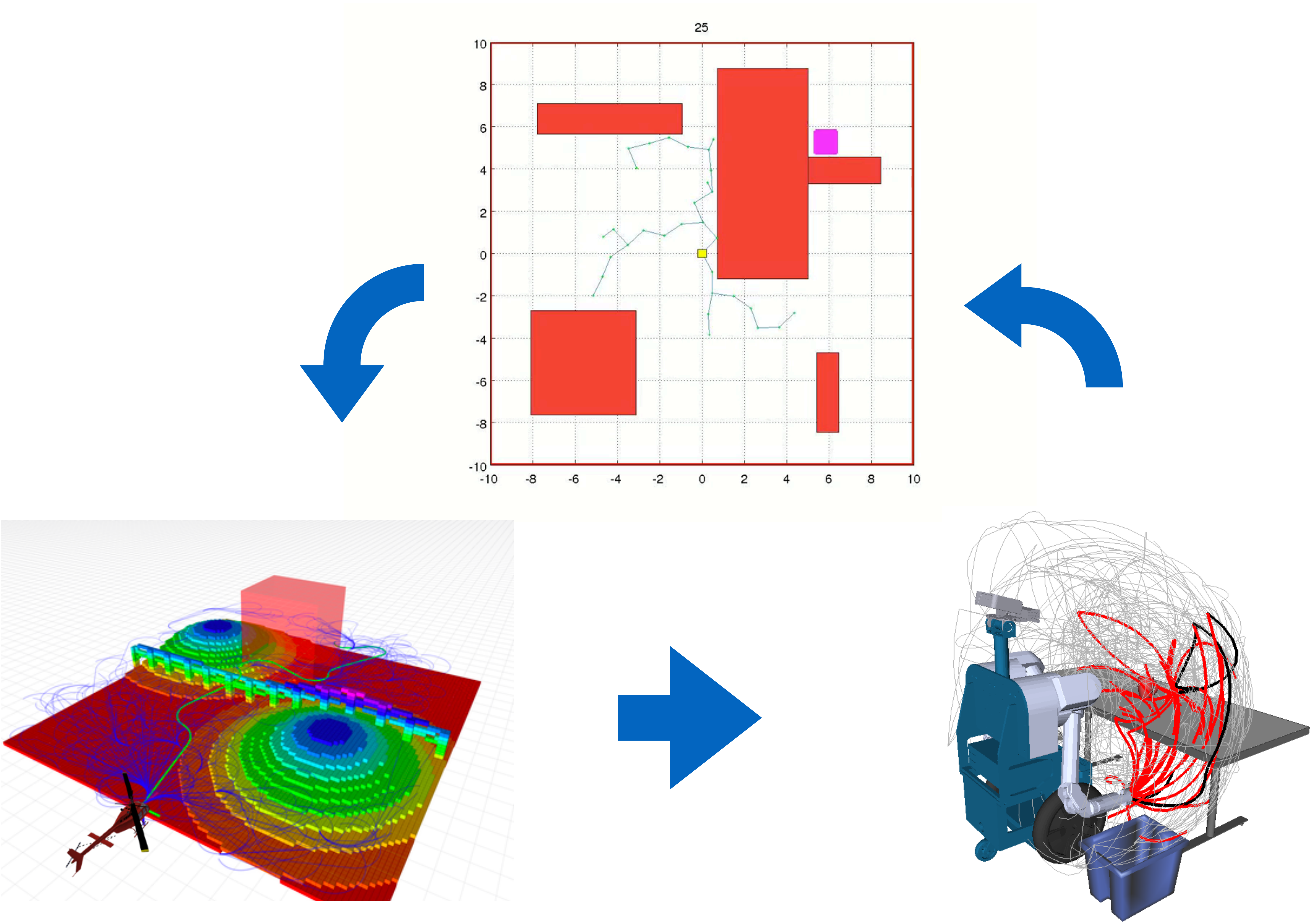
Real time helicopter planning



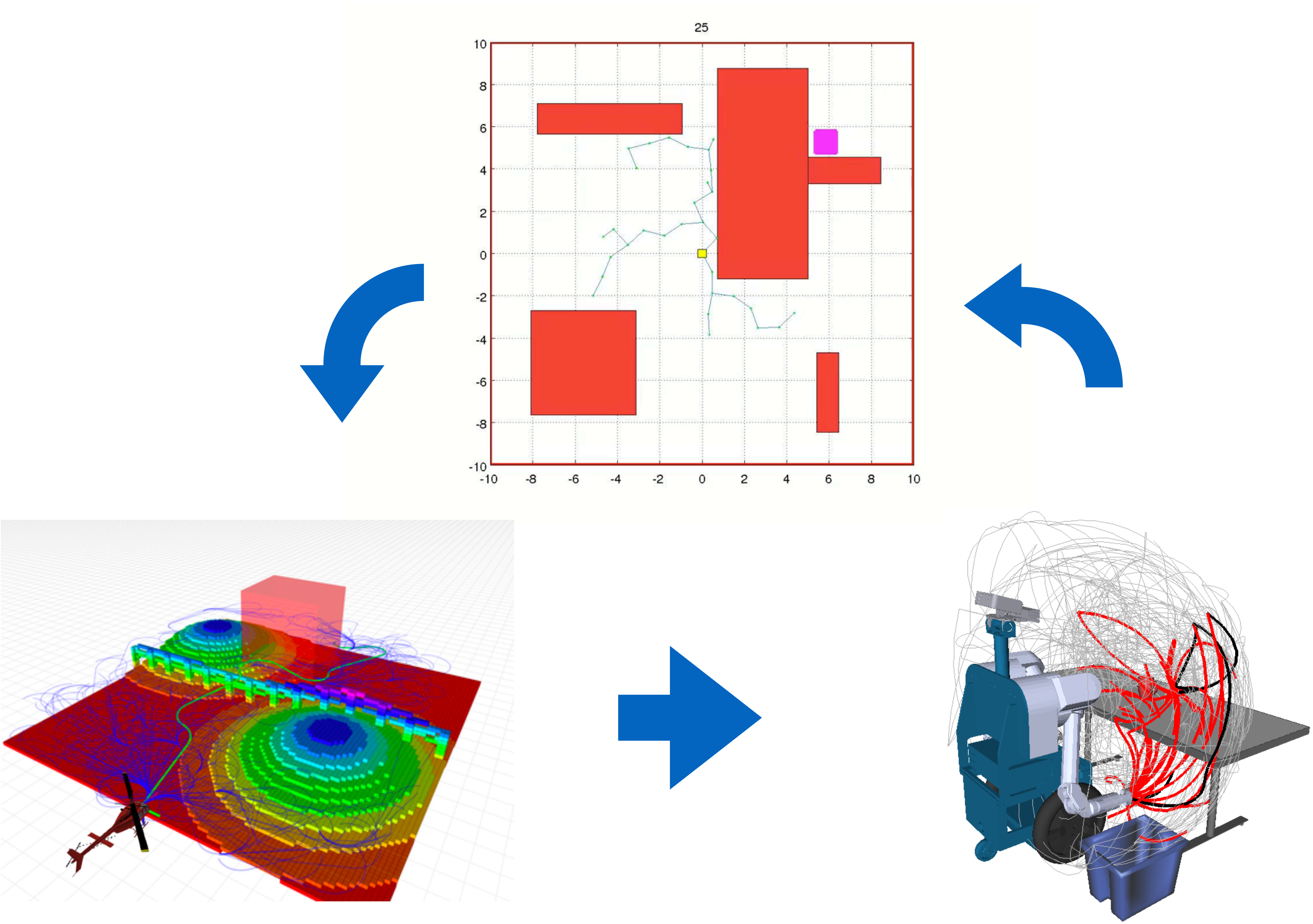
Real time helicopter planning



Generality of planning algorithms



Generality of planning algorithms



Challenges that we will focus on

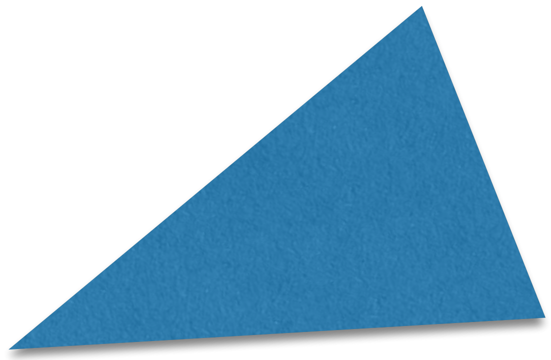
1. Search in continuous space such that a feasible path exists? optimal path?

2. Solve this problems in real-time

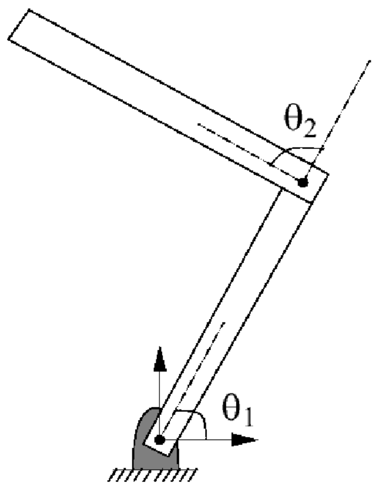
Planning ingredients

Configuration Space

The Configuration Space



(a) Translating Triangle



(b) 2-joint planar arm

(c) Racecar

(d) Manipulator

The Configuration Space

The configuration space or C-space is the manifold that contains the set of transformations achievable by the robot.

$$\mathcal{C}$$

Complete specification of the location of every point on robot geometry

The Configuration Space

The configuration space is a topological space

(Planning Algorithms, Ch 4.1.1)

The Configuration Space

The configuration space is a topological space

A set X is called a *topological space* if there is a collection of subsets of X called *open sets* for which the following axioms hold:

1. The union of any number of open sets is an open set.
2. The intersection of a finite number of open sets is an open set.
3. Both X and $\emptyset$ are open sets.

(Planning Algorithms, Ch 4.1.1)

The Configuration Space

The configuration space is a topological space

A set X is called a *topological space* if there is a collection of subsets of X called *open sets* for which the following axioms hold:

1. The union of any number of open sets is an open set.
2. The intersection of a finite number of open sets is an open set.
3. Both X and $\emptyset$ are open sets.

Intuition: Most general notion of space that allows for definition of continuity, connectedness and convergence

(Planning Algorithms, Ch 4.1.1)

The Configuration Space

The configuration space is a manifold

(Planning Algorithms, Ch 4.1.2)

The Configuration Space

The configuration space is a manifold

Manifold definition A topological space $M \subseteq \mathbb{R}^m$ is a *manifold*[4] if for every $x \in M$, an open set $O \subset M$ exists such that: 1) $x \in O$, 2) O is homeomorphic to $\mathbb{R}^n$, and 3) n is fixed for all $x \in M$. The fixed n is referred to as the *dimension* of the manifold, M. The second condition is the most important. It states that in the vicinity of any point, $x \in M$, the space behaves just like it would in the vicinity of any point $y \in \mathbb{R}^n$; intuitively, the set of directions that one can move appears the same in either case. Several simple examples that may or may not be manifolds are shown in Figure 4.4.

(Planning Algorithms, Ch 4.1.2)

The Configuration Space

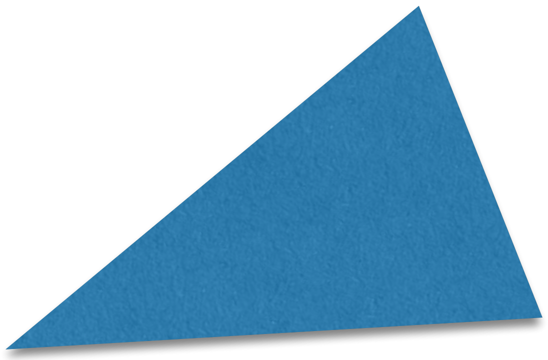
The configuration space is a manifold

Manifold definition A topological space $M \subseteq \mathbb{R}^m$ is a *manifold*[4] if for every $x \in M$, an open set $O \subset M$ exists such that: 1) $x \in O$, 2) O is homeomorphic to $\mathbb{R}^n$, and 3) n is fixed for all $x \in M$. The fixed n is referred to as the *dimension* of the manifold, M. The second condition is the most important. It states that in the vicinity of any point, $x \in M$, the space behaves just like it would in the vicinity of any point $y \in \mathbb{R}^n$; intuitively, the set of directions that one can move appears the same in either case. Several simple examples that may or may not be manifolds are shown in Figure 4.4.

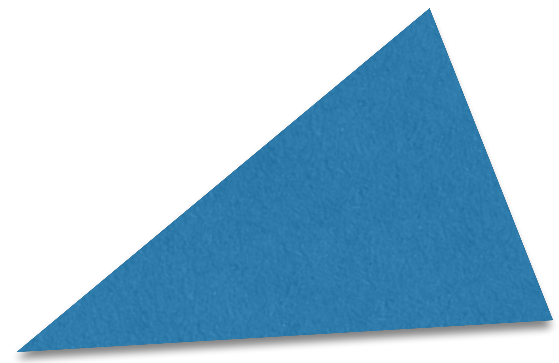
Intuition: Manifold is a nice topological space that locally behaves like a surface

(Planning Algorithms, Ch 4.1.2)

Example 1: Translating triangle



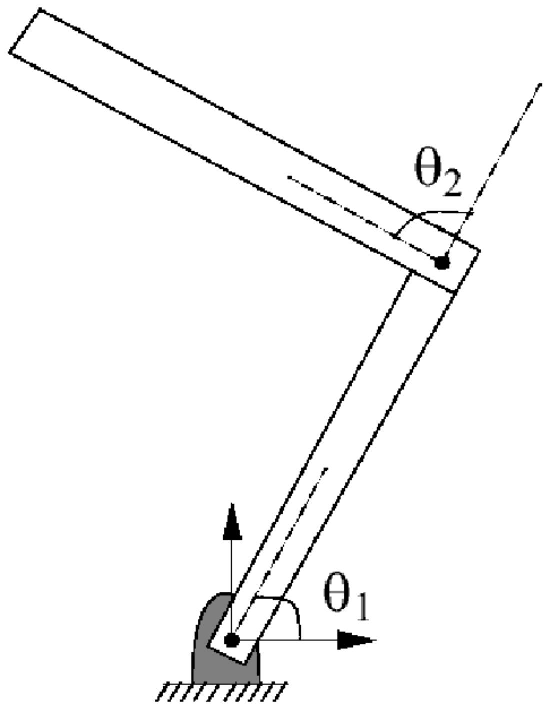
Example 1: Translating triangle



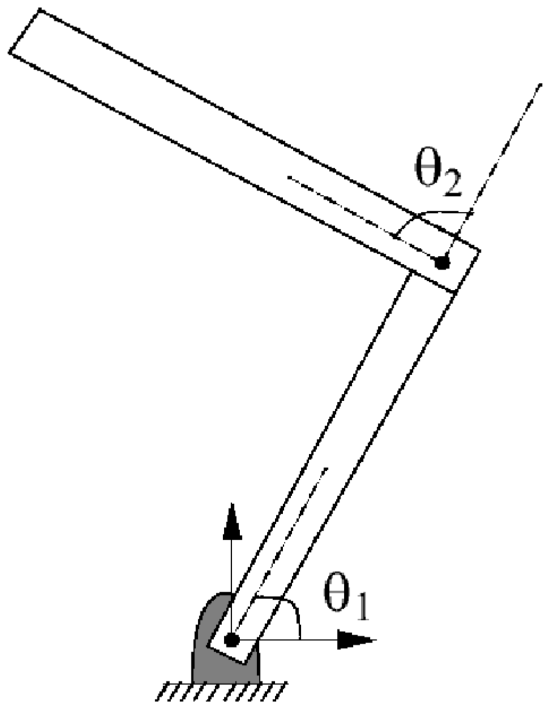
$$\mathbb{R} \times \mathbb{R} = \mathbb{R}^2$$

(cartesian product)

Example 2: 2-joint planar arm

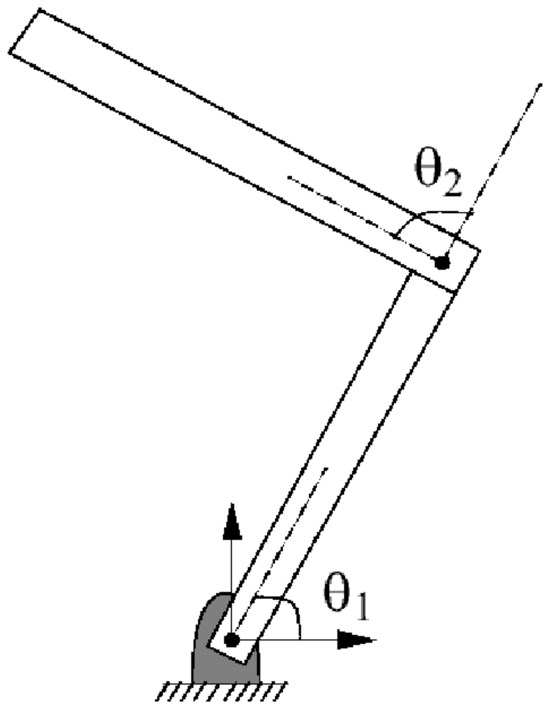


Example 2: 2-joint planar arm



$$\mathbb{S}^1 \times \mathbb{S}^1$$

Example 2: 2-joint planar arm

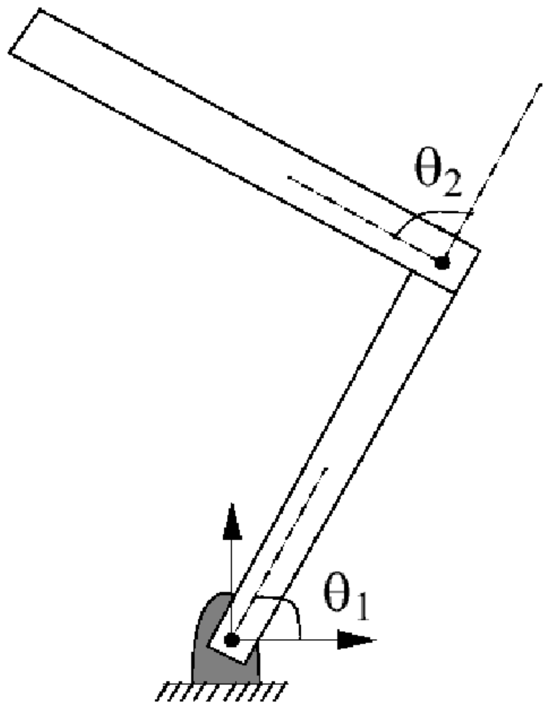


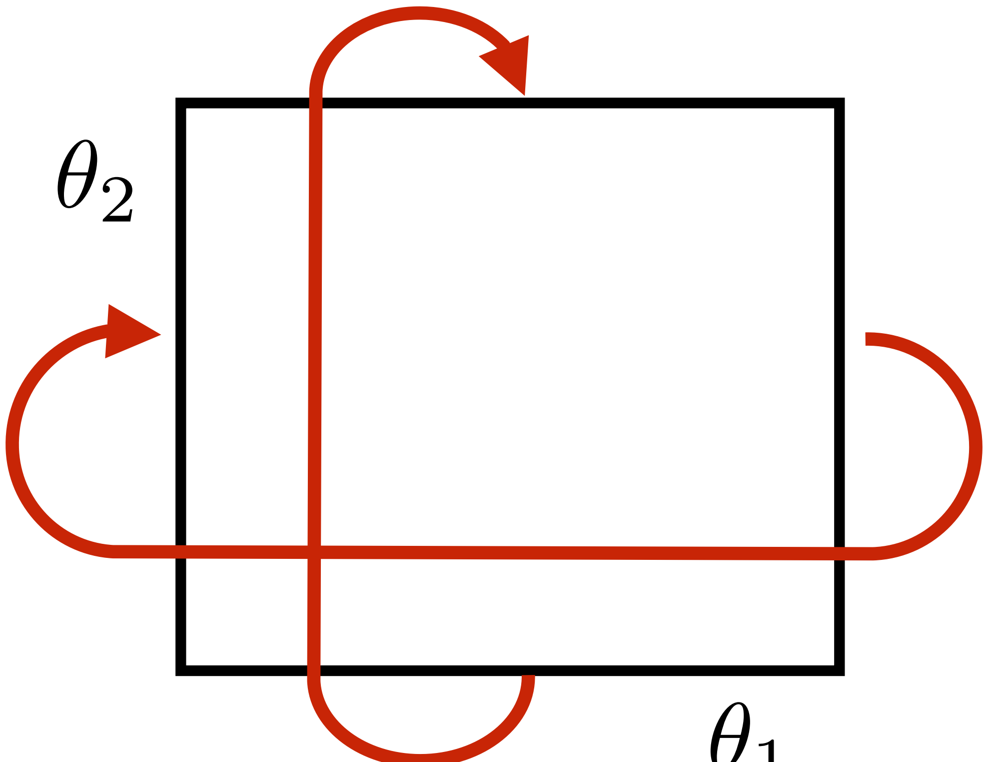
$$\mathbb{S}^1 \times \mathbb{S}^1$$

Circle

$$\mathbb{S}^1 = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1\}.$$

Example 2: 2-joint planar arm



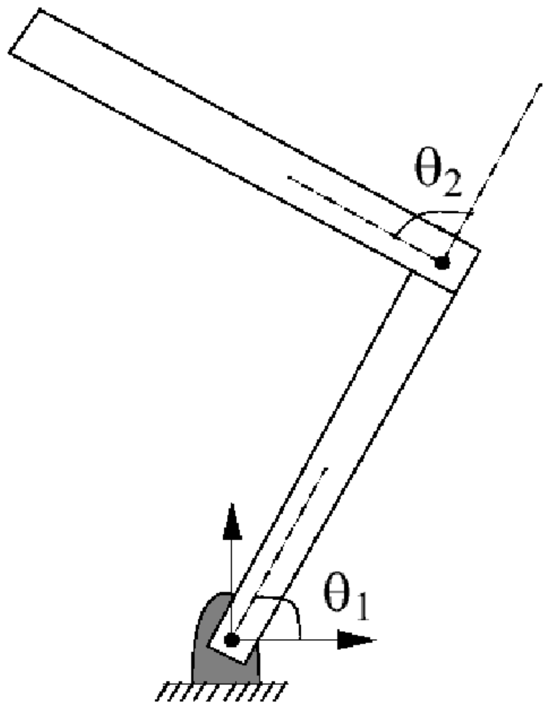


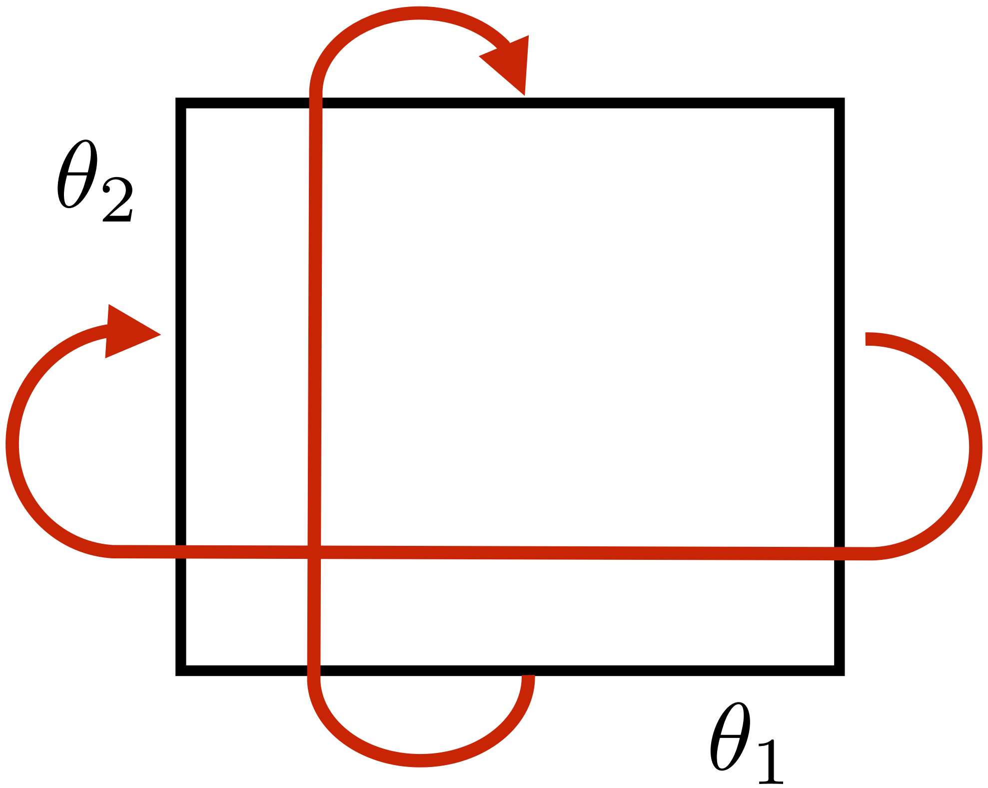
$$\mathbb{S}^1 \times \mathbb{S}^1$$

Circle

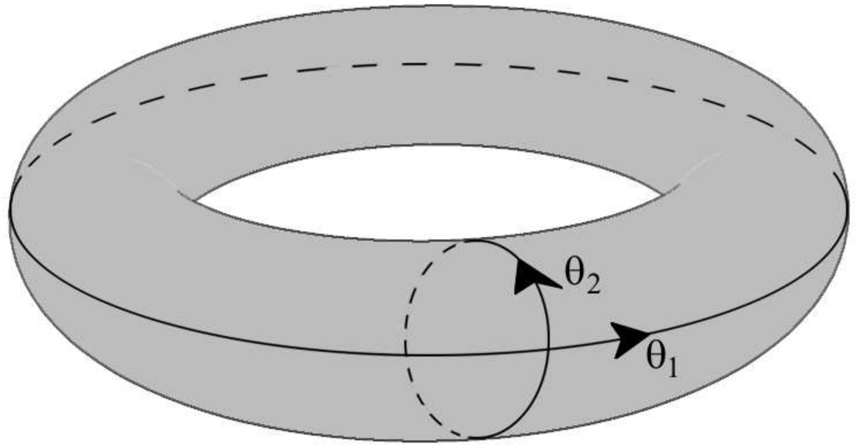
$$\mathbb{S}^1 = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1\}.$$

Example 2: 2-joint planar arm





$$\mathbb{S}^1 \times \mathbb{S}^1 = \mathbb{T}^2$$



Circle

$$\mathbb{S}^1 = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1\}.$$

Example 3: Racecar

Example 3: Racecar

$$\mathbb{R}^2 \times \mathbb{S}^1$$

Example 3: Racecar

$$\mathbb{R}^2 \times \mathbb{S}^1$$

special euclidean group $SE(2)$

Common C-spaces

Type of Robot	**$\mathcal{C}$-space Representation**
Mobile robot translating in the plane	$\mathbb{R}^2$
Mobile robot translating and rotating in the plane	$SE(2)$ or $\mathbb{R}^2 \times S^1$
Rigid body translating in the three-space	$\mathbb{R}^3$
A spacecraft	$SE(3)$ or $\mathbb{R}^3 \times SO(3)$
An n-joint revolute arm	T^n
A planar mobile robot with an attached n-joint arm	$SE(2) \times T^n$

(Kavraki and LaValle)

Obstacles

Obstacle specification

Obstacle specification

Robot operates in a 2D / 3D workspace $\mathcal{W} = \mathbb{R}^2 \text{ or } \mathbb{R}^3$

Obstacle specification

Robot operates in a 2D / 3D workspace $\mathcal{W} = \mathbb{R}^2 \text{ or } \mathbb{R}^3$

Subset of this space is obstacles $\mathcal{O} \subset \mathcal{W}$

semi-algebraic models (polygons, polyhedra)

Obstacle specification

Robot operates in a 2D / 3D workspace $\mathcal{W} = \mathbb{R}^2 \text{ or } \mathbb{R}^3$

Subset of this space is obstacles $\mathcal{O} \subset \mathcal{W}$

semi-algebraic models (polygons, polyhedra)

Geometric shape of the robot
(set of points occupied by robot at a config) $\mathcal{A}(q) \subset \mathcal{W}$

Obstacle specification

Robot operates in a 2D / 3D workspace $\mathcal{W} = \mathbb{R}^2 \text{ or } \mathbb{R}^3$

Subset of this space is obstacles $\mathcal{O} \subset \mathcal{W}$

semi-algebraic models (polygons, polyhedra)

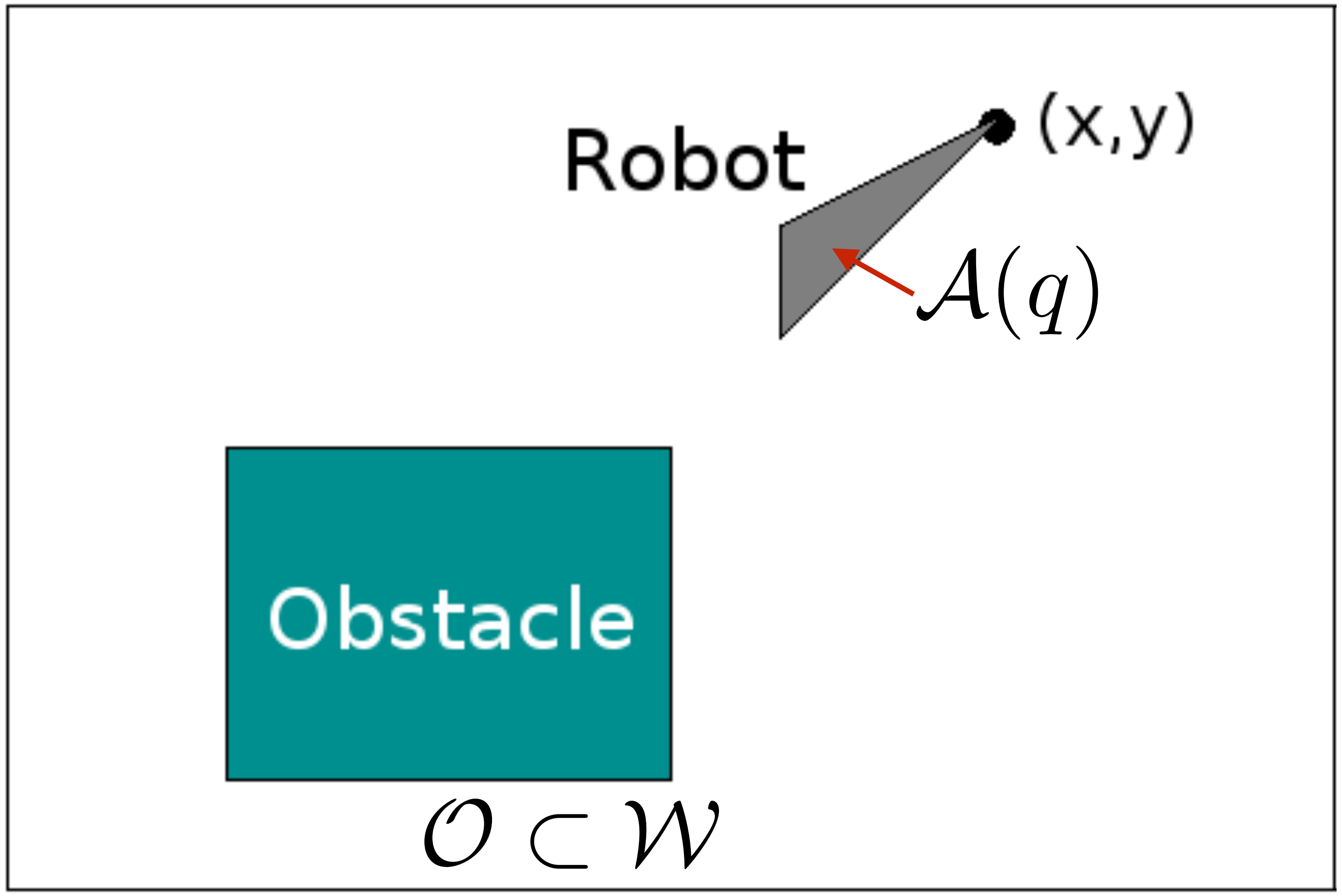
Geometric shape of the robot
(set of points occupied by robot at a config) $\mathcal{A}(q) \subset \mathcal{W}$

C-space obstacle region

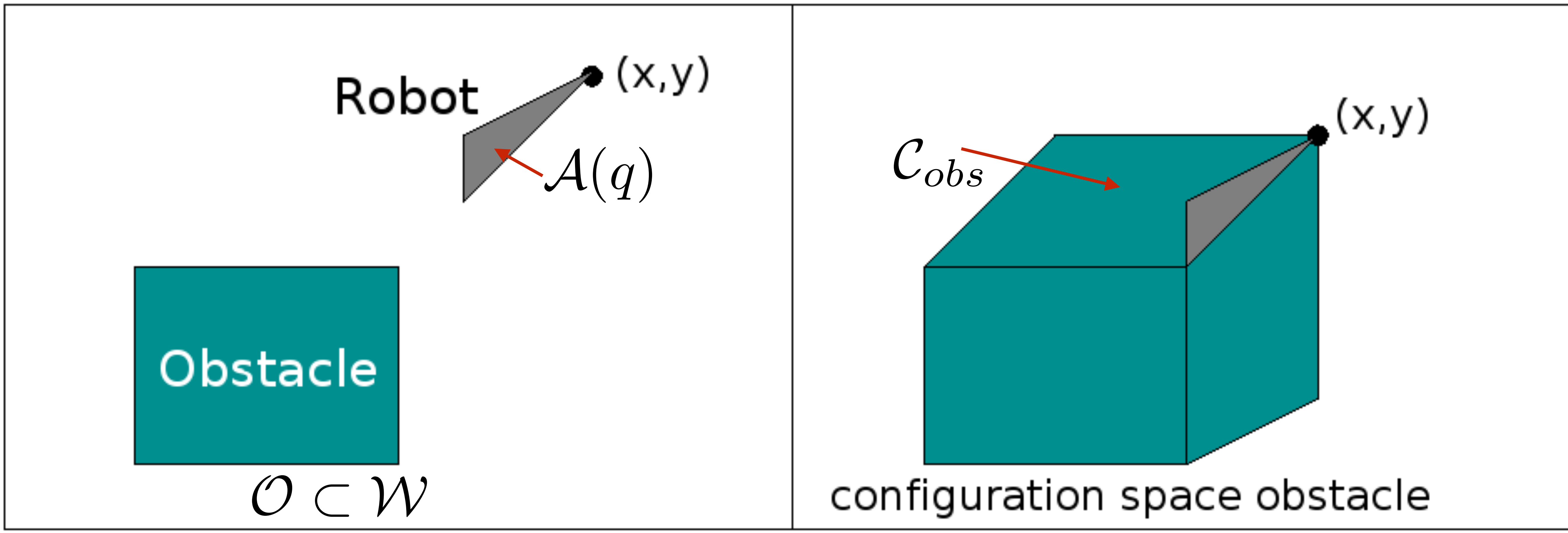
$$\mathcal{C}_{obs} = \{\boldsymbol{q} \in \mathcal{C} \mid \mathcal{A}(\boldsymbol{q}) \cap \mathcal{O} \neq \emptyset\}$$

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$$

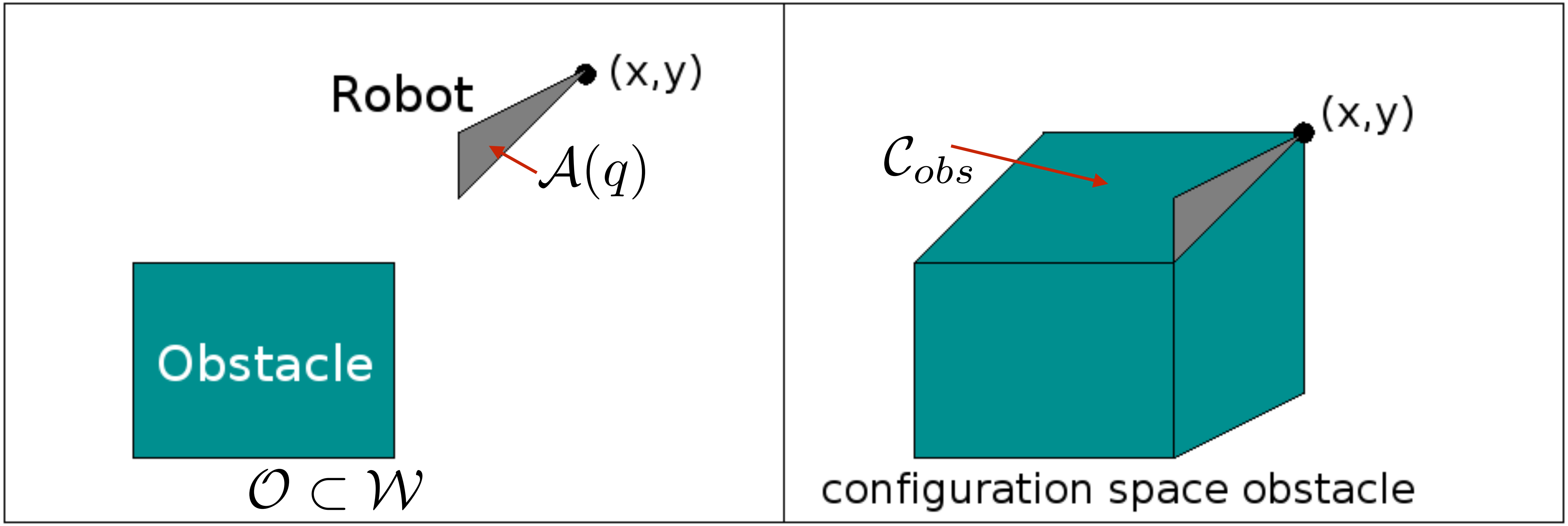
Example 1: Translating triangle



Example 1: Translating triangle

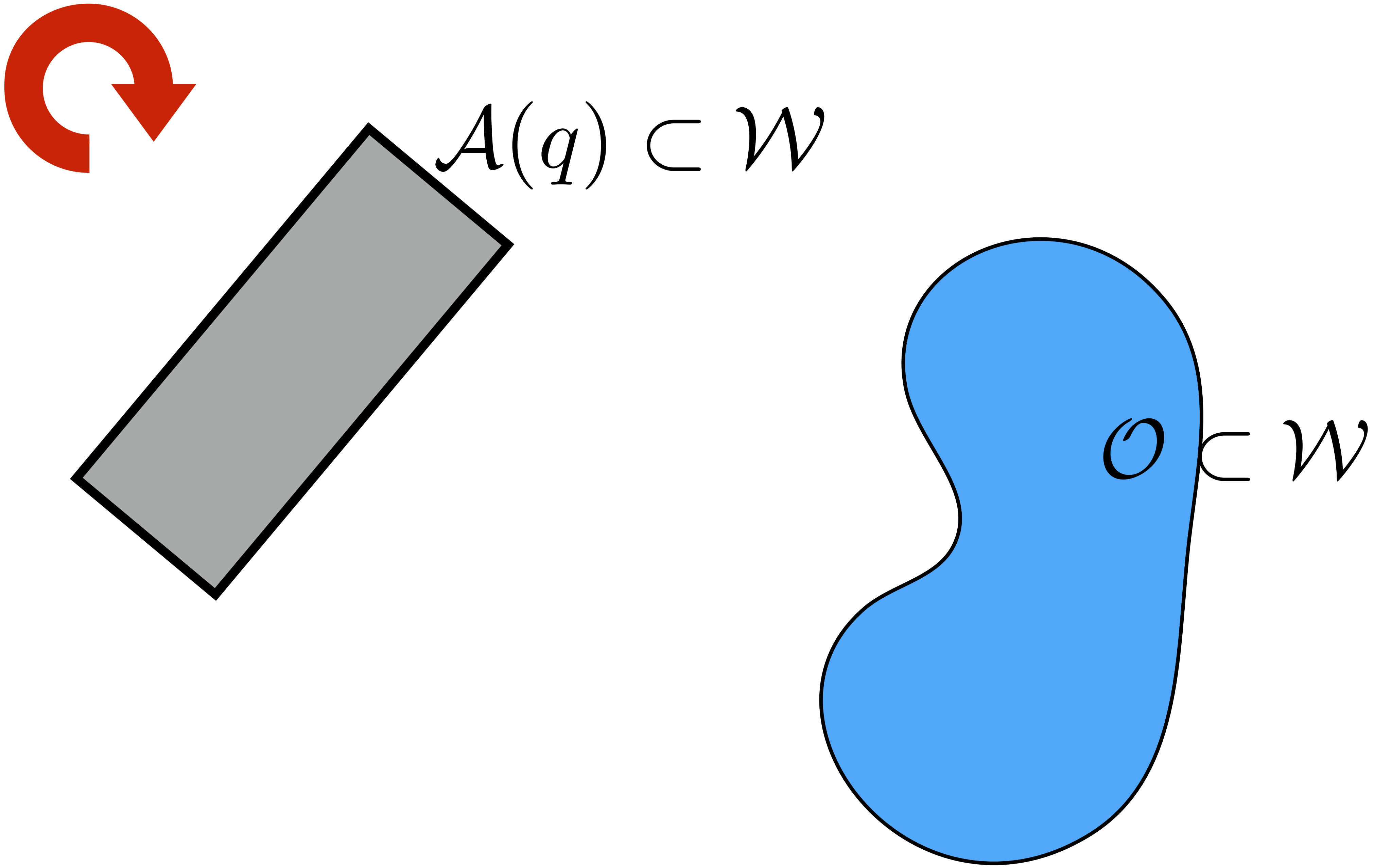


Example 1: Translating triangle



Can be efficiently computed using Minkowski sum

Example 2: SE(2) robot

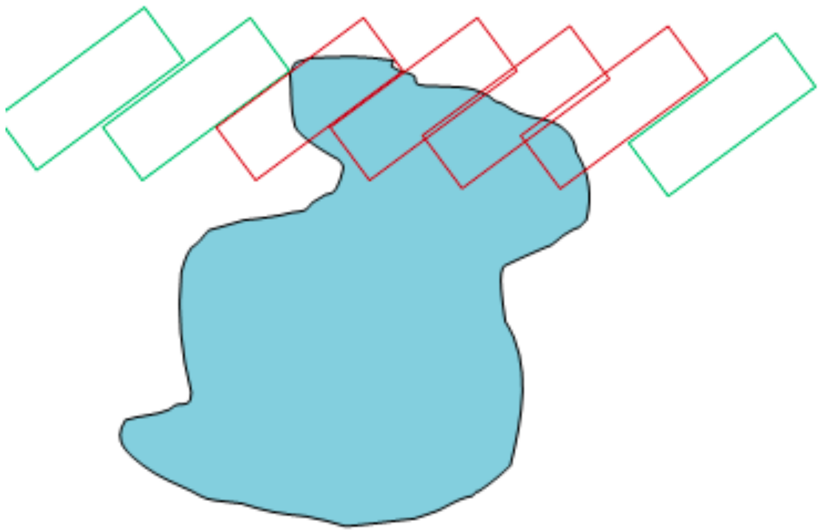


Example 2: SE(2) robot

(Courtesy Matt Klingensmith)

https://www.gamasutra.com/blogs/MattKlingensmith/20130907/199787/Overview_of_Motion_Planning.php

Example 2: SE(2) robot



(Courtesy Matt Klingensmith)

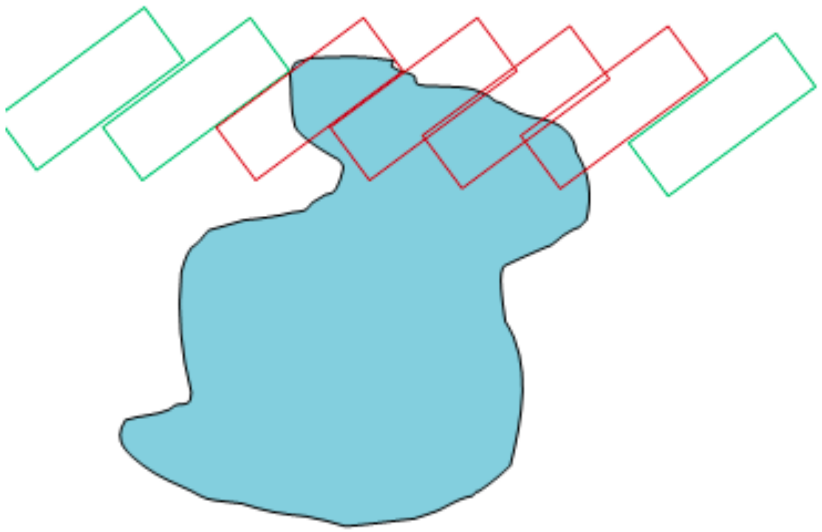
https://www.gamasutra.com/blogs/MattKlingensmith/20130907/199787/Overview_of_Motion_Planning.php

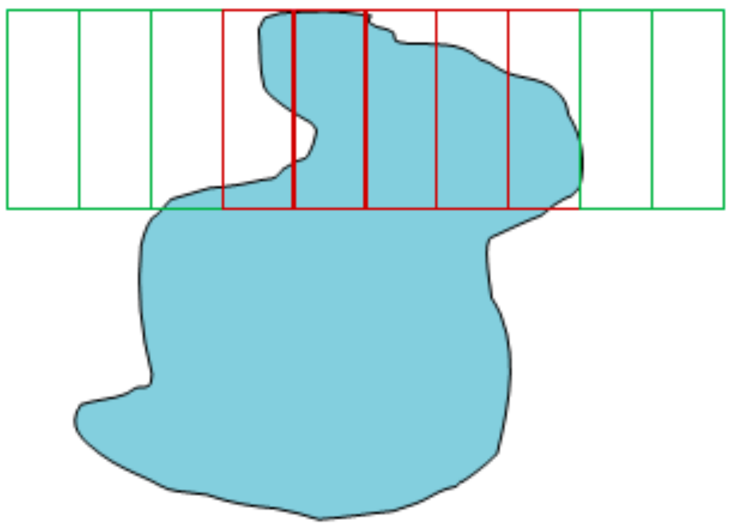
Example 2: SE(2) robot

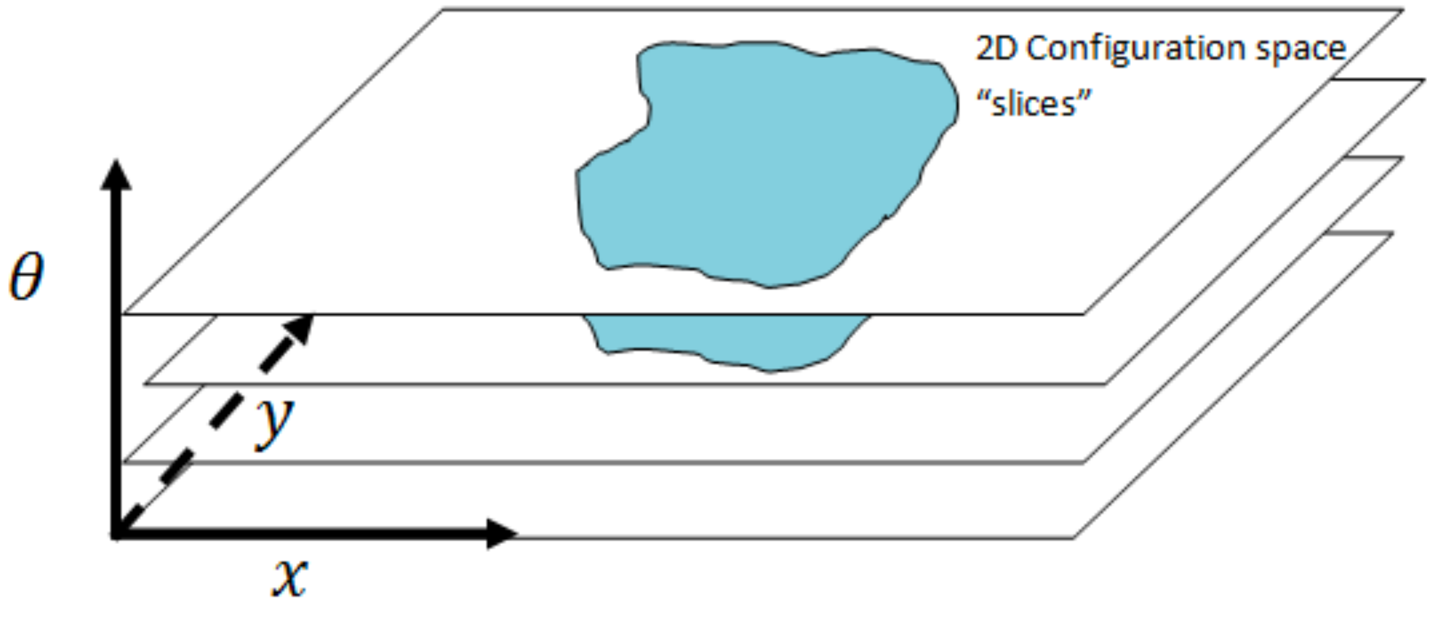
(Courtesy Matt Klingensmith)

https://www.gamasutra.com/blogs/MattKlingensmith/20130907/199787/Overview_of_Motion_Planning.php

Example 2: SE(2) robot



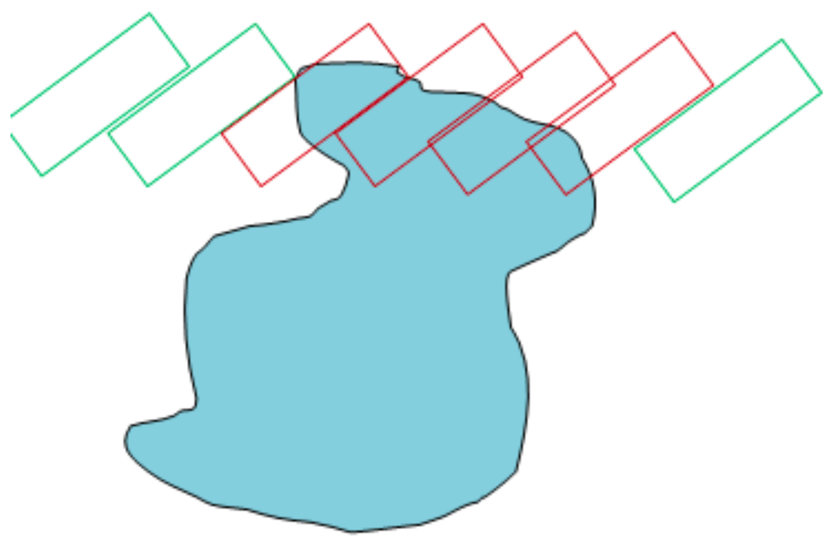


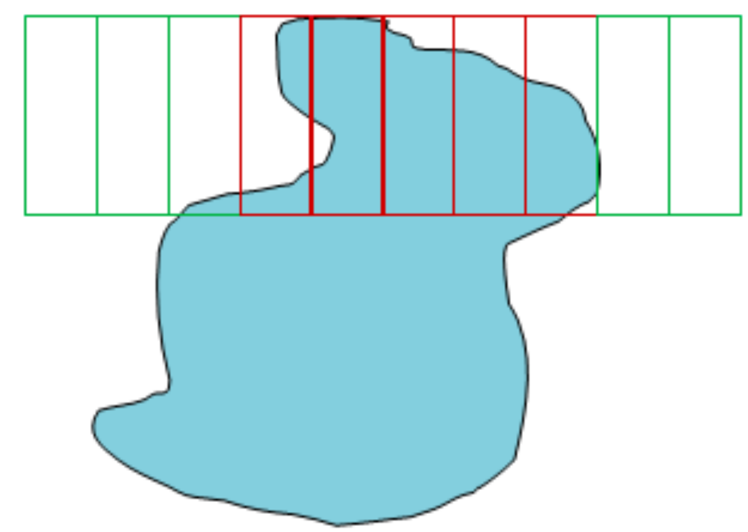


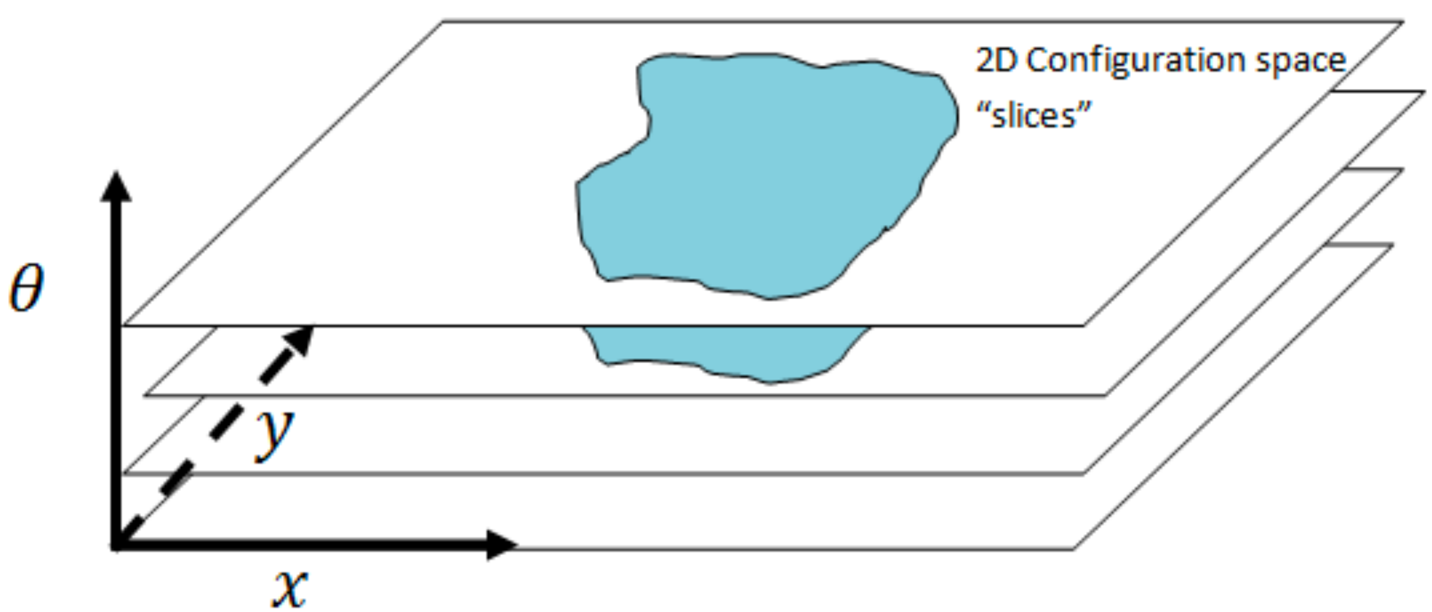
(Courtesy Matt Klingensmith)

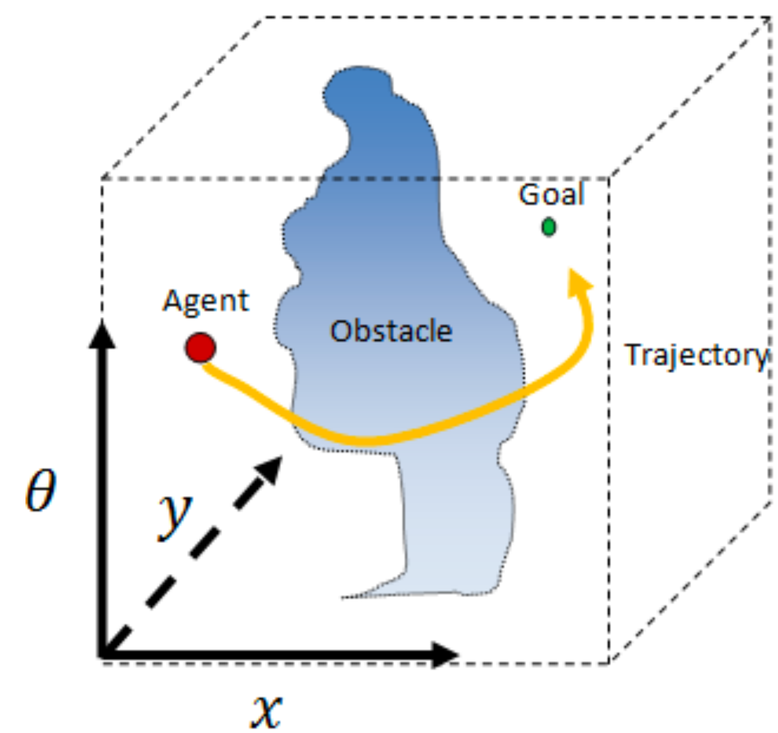
https://www.gamasutra.com/blogs/MattKlingensmith/20130907/199787/Overview_of_Motion_Planning.php

Example 2: SE(2) robot





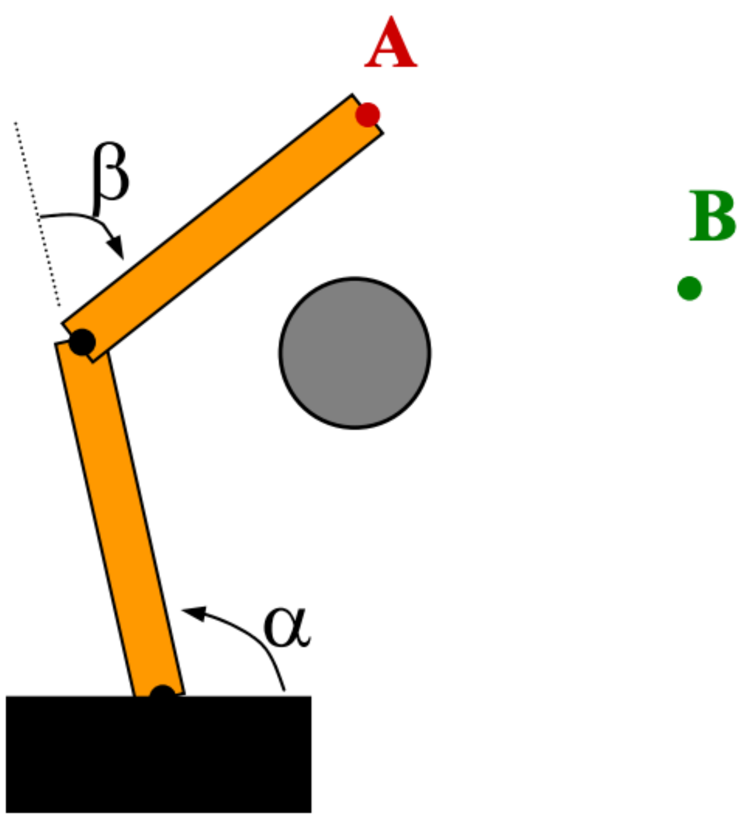


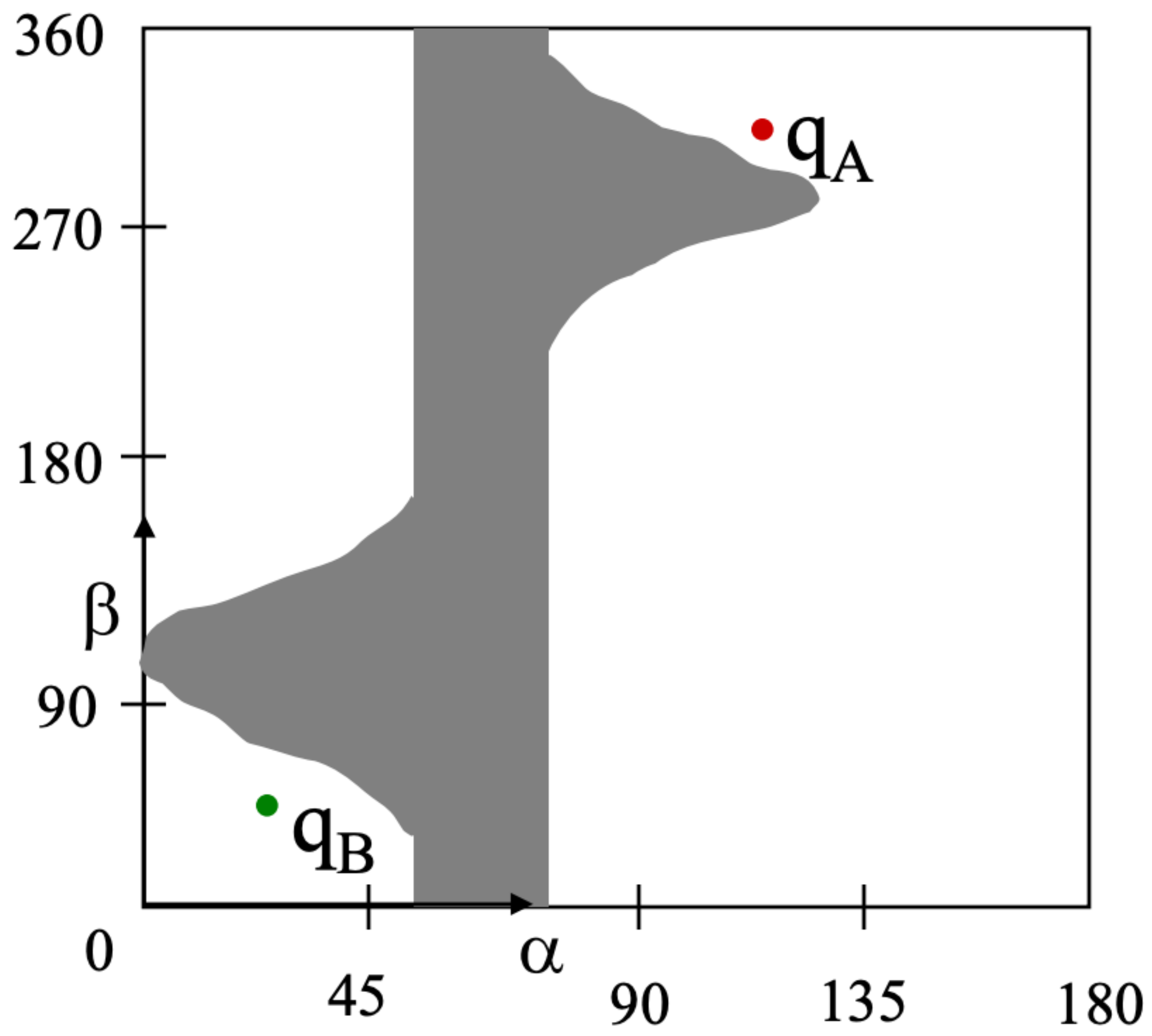


(Courtesy Matt Klingensmith)

https://www.gamasutra.com/blogs/MattKlingensmith/20130907/199787/Overview_of_Motion_Planning.php

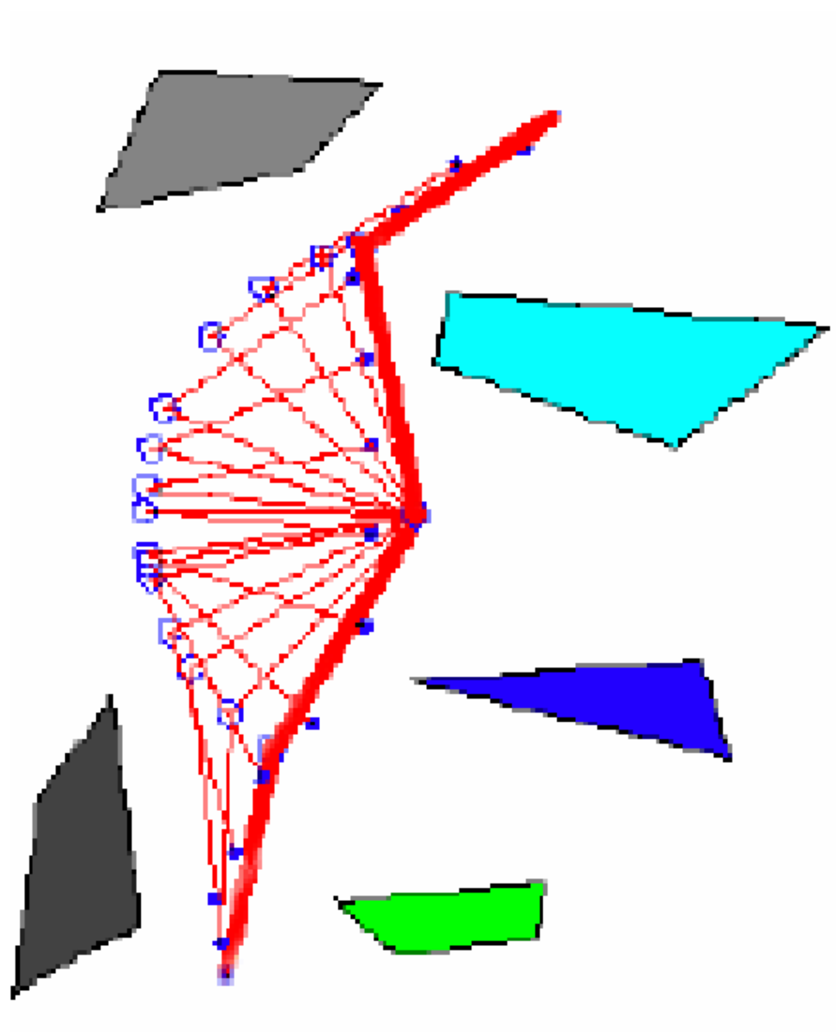
Example 3: 2-link planar arm

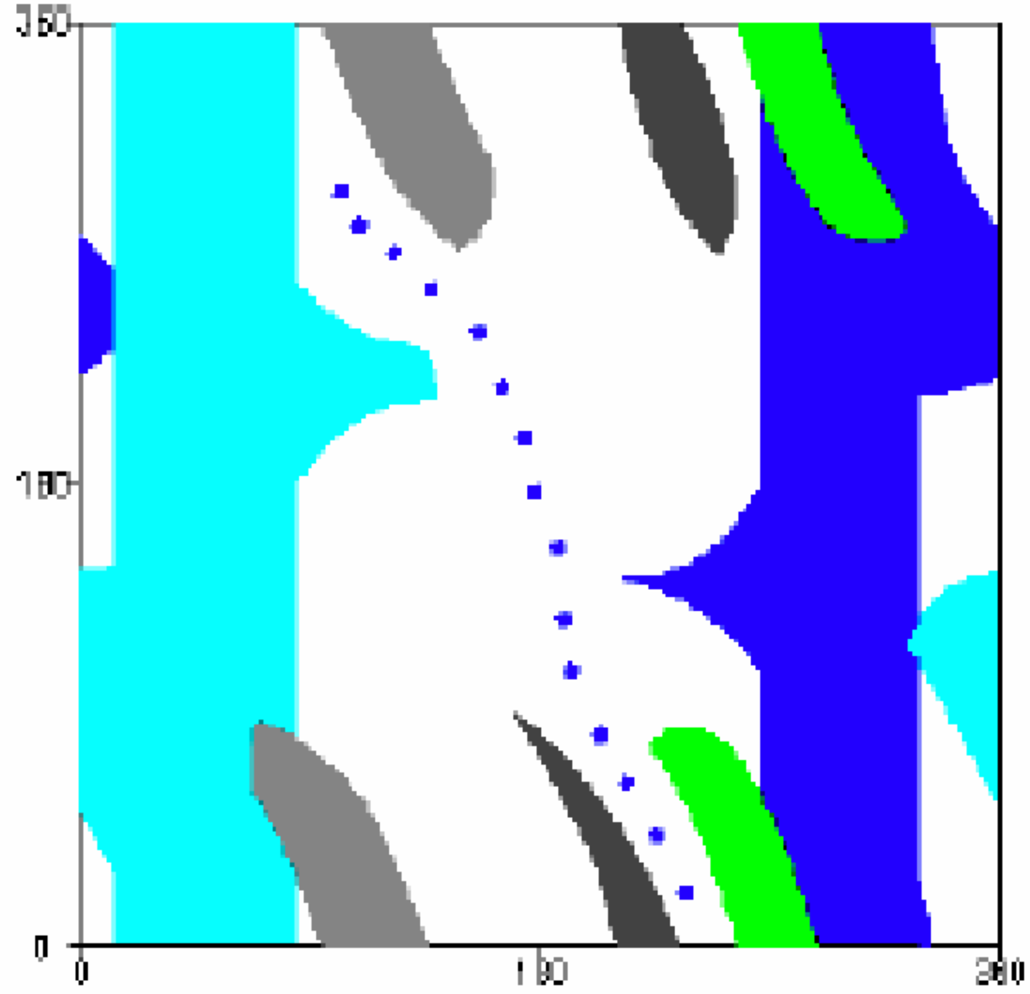




Courtesy Tapomayukh Bhattacharya

Example 3: 2-link planar arm

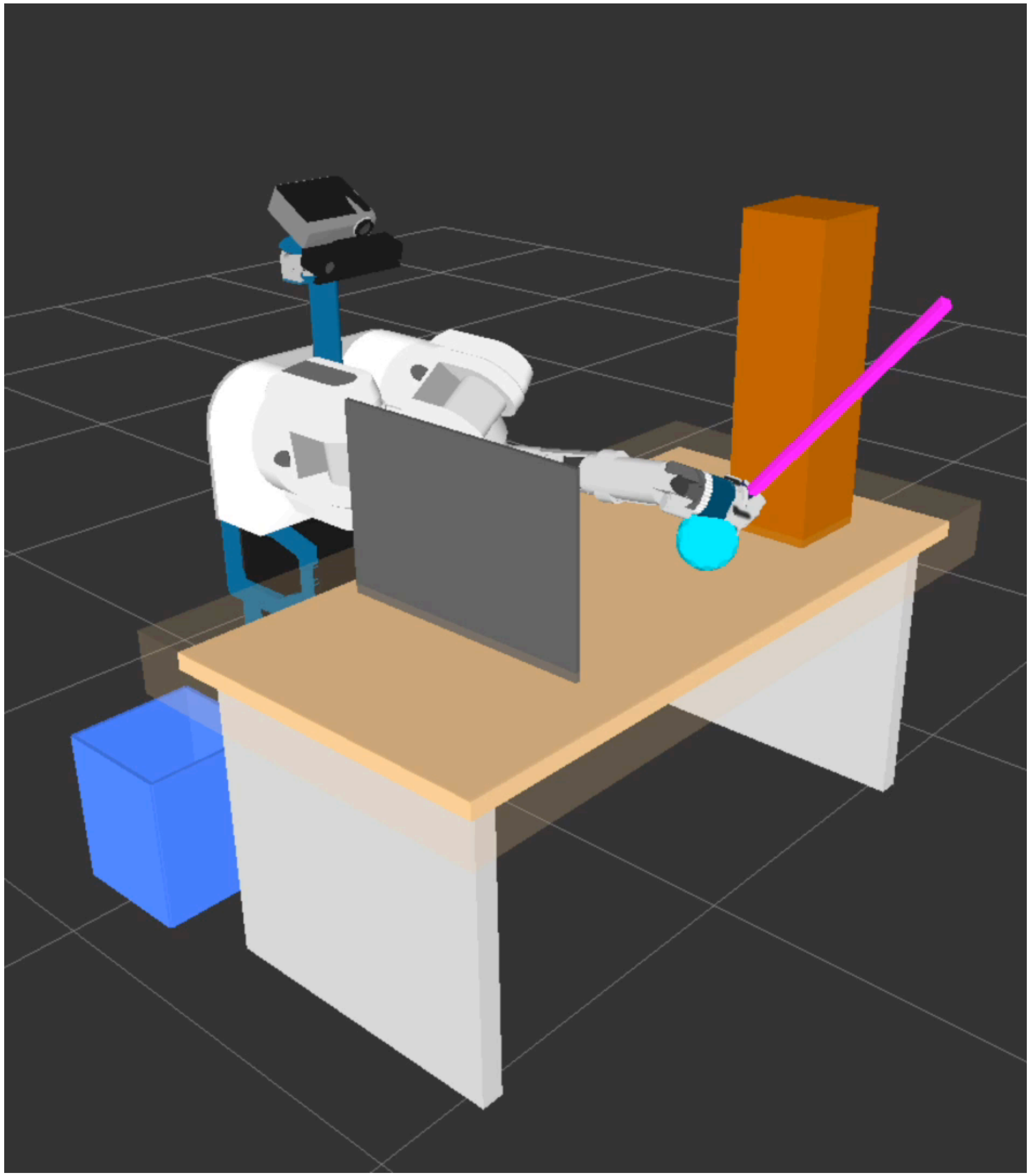




Courtesy Tapomayukh Bhattacharya

Geometric Path Planning Problem

Geometric Path Planning Problem



Also known as
Piano Mover's Problem (Reif 79)

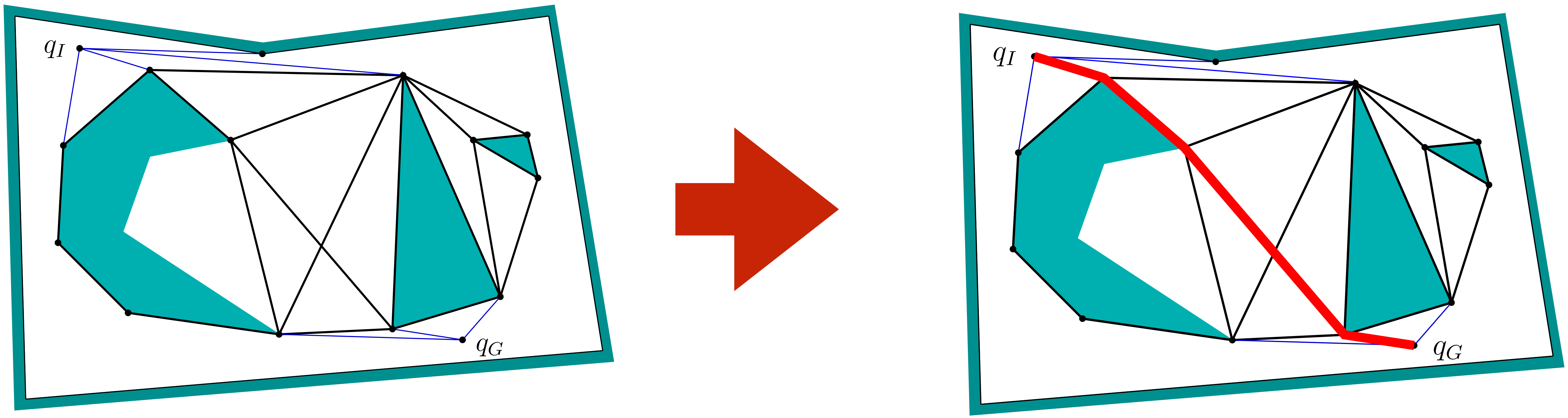
Given:

1. A *workspace* $\mathcal{W}$, where either $\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$.
2. An *obstacle region* $\mathcal{O} \subset \mathcal{W}$.
3. A *robot* defined in $\mathcal{W}$. Either a rigid body $\mathcal{A}$ or a collection of m links: $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$.
4. The *configuration space* $\mathcal{C}$ (C_{obs} and C_{free} are then defined).
5. An *initial configuration* $\boldsymbol{q_I} \in \mathcal{C}_{free}$.
6. A *goal configuration* $\boldsymbol{q_G} \in \mathcal{C}_{free}$. The initial and goal configuration are often called a *query* $(\boldsymbol{q_I}, \boldsymbol{q_G})$.

Compute a (continuous) path, $\tau : [0,1] \rightarrow \mathcal{C}_{free}$, *such that* $\tau(0) = \boldsymbol{q_I}$ *and* $\tau(1) = \boldsymbol{q_G}$.

Also may want to minimize cost $c(\tau)$

Can we solve this for some problems?



Yes! E.g. 2D polygon robots / obstacles can be solved with visibility graphs

So, are we done?

No! Planning is hard

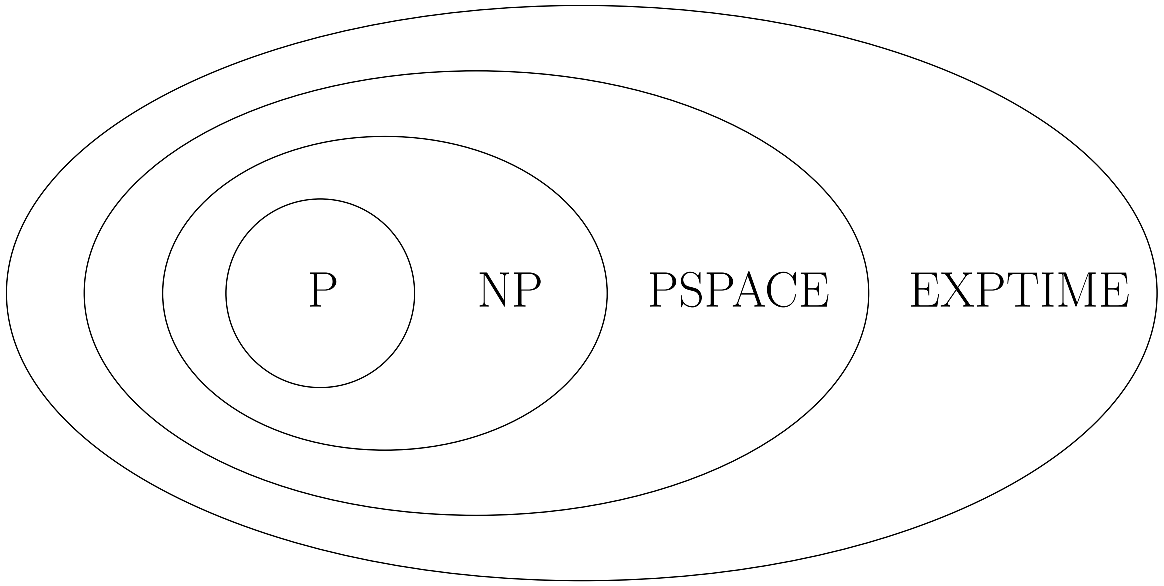
Hardness of motion planning

Hardness of motion planning

Piano Mover's problem is PSPACE-hard (Reif et al.)

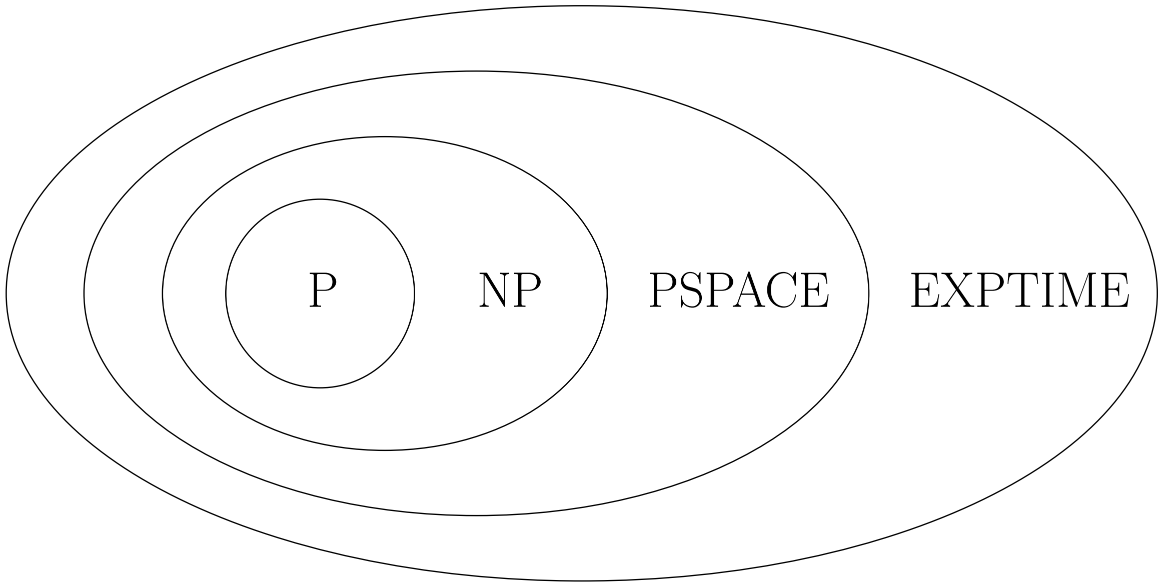
Hardness of motion planning

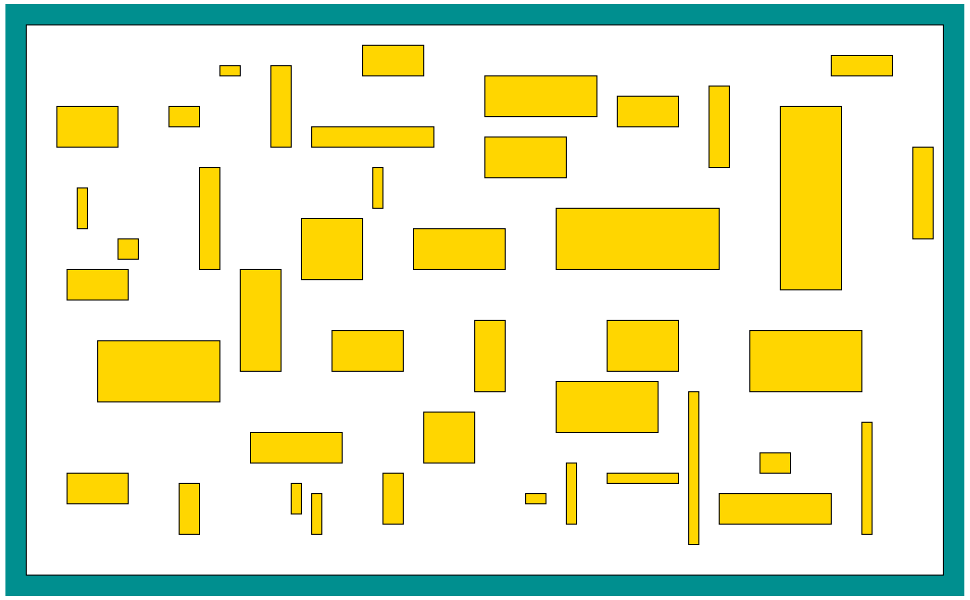
Piano Mover's problem is PSPACE-hard (Reif et al.)



Hardness of motion planning

Piano Mover's problem is PSPACE-hard (Reif et al.)



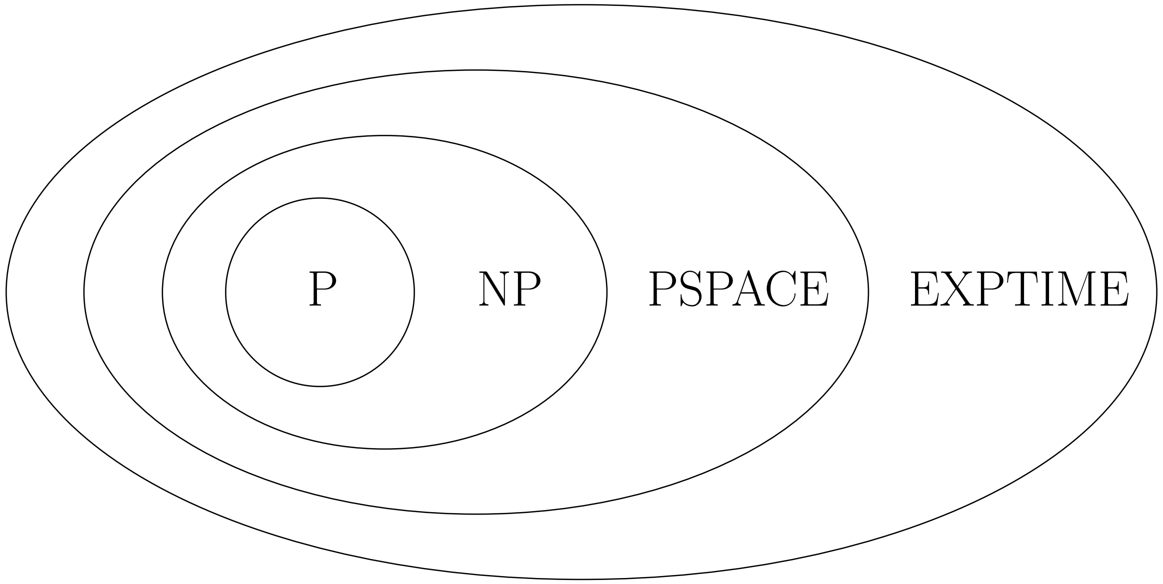


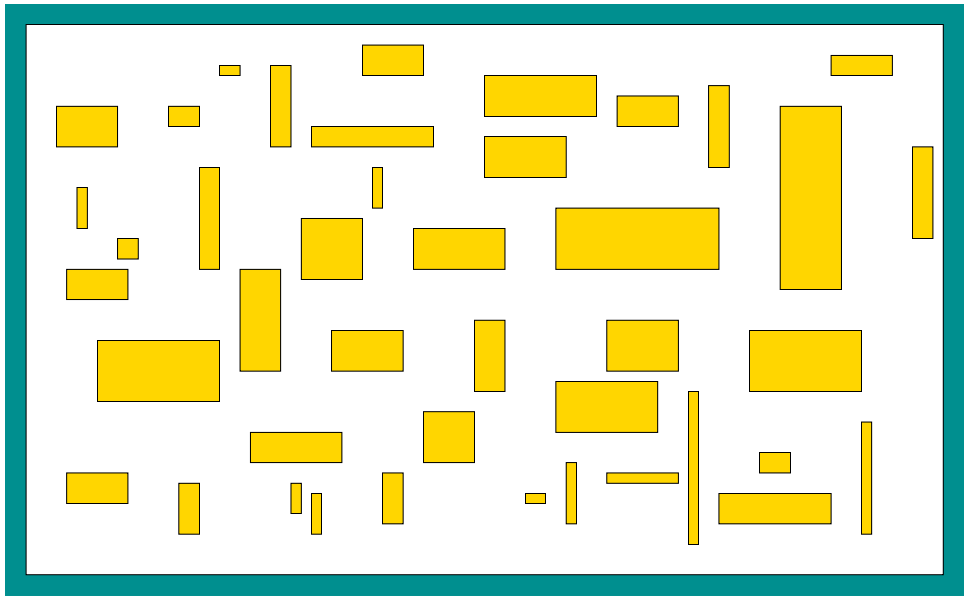
Even planning for translating rectangles is PSPACE-hard!

(Hopcroft et al. 84)

Hardness of motion planning

Piano Mover's problem is PSPACE-hard (Reif et al.)





Even planning for translating rectangles is PSPACE-hard!

(Hopcroft et al. 84)

Certain 3D robot planning under uncertain is NEXPTIME-hard!

(Canny et al. 87)

Why is it hard?

1. Computing the C-space obstacle is hard

2. Planning in continuous high-dimension space is hard

Exponential dependency on dimension

Research in Motion Planning:

Tractable approximations with provable guarantees

Differential constraints

In geometric path planning, we were only dealing with C-space

$$q \in \mathcal{C}$$

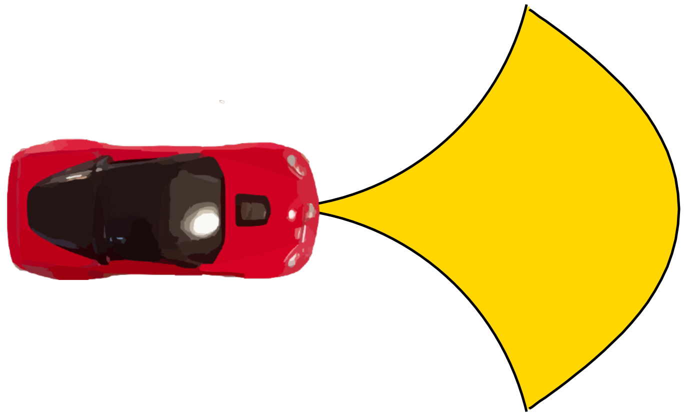
Differential constraints

In geometric path planning, we were only dealing with C-space

$$q \in \mathcal{C}$$

We now introduce differential constraints

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = f(\begin{bmatrix} q \\ q \end{bmatrix}, u)$$



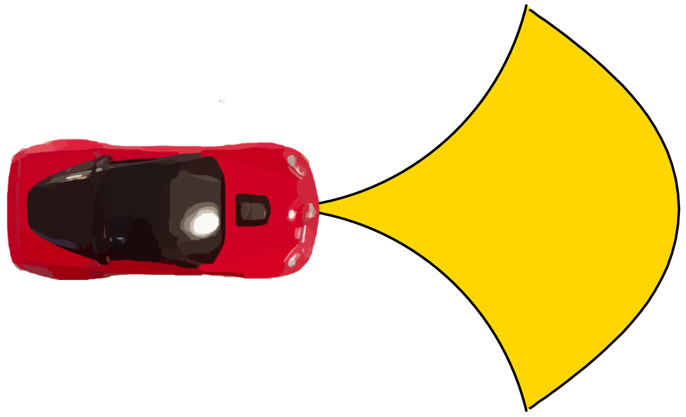
Differential constraints

In geometric path planning, we were only dealing with C-space

$$q \in \mathcal{C}$$

We now introduce differential constraints

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = f(\begin{bmatrix} q \\ q \end{bmatrix}, u)$$



Let the state space x be the following augmented C-space

$$x = (q, \dot{q}) \qquad \dot{x} = f(x, u)$$

Motion planning under differential constraints

1. Given world, obstacles, C-space, robot geometry (same)

2. Introduce state space X. Compute free and obstacle state space.

3. Given an action space U

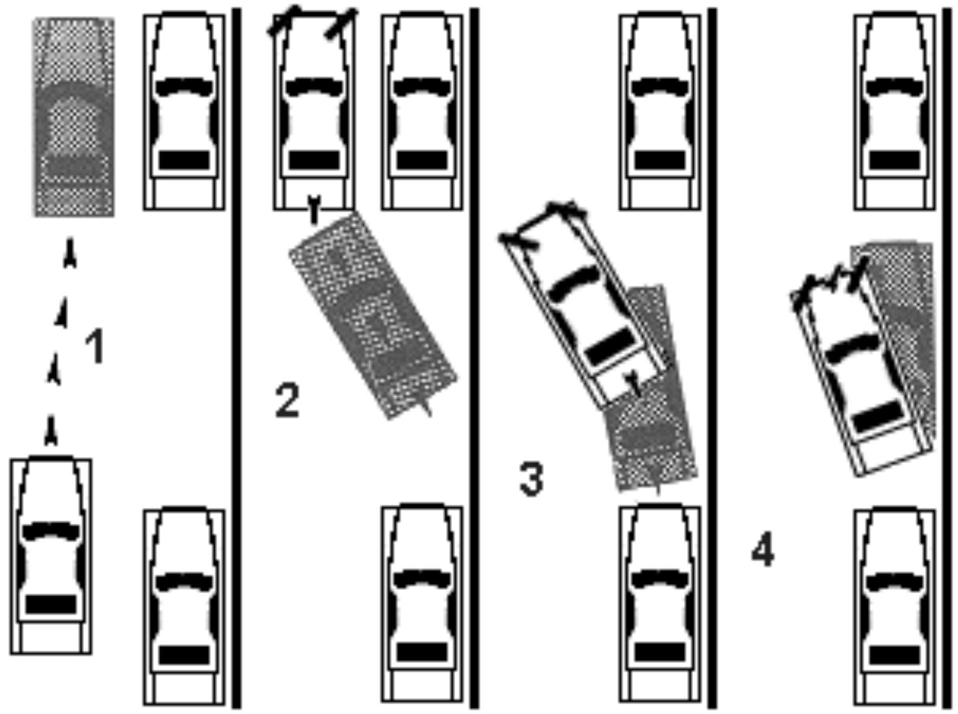
4. Given a state transition equations $\dot{x} = f(x, u)$

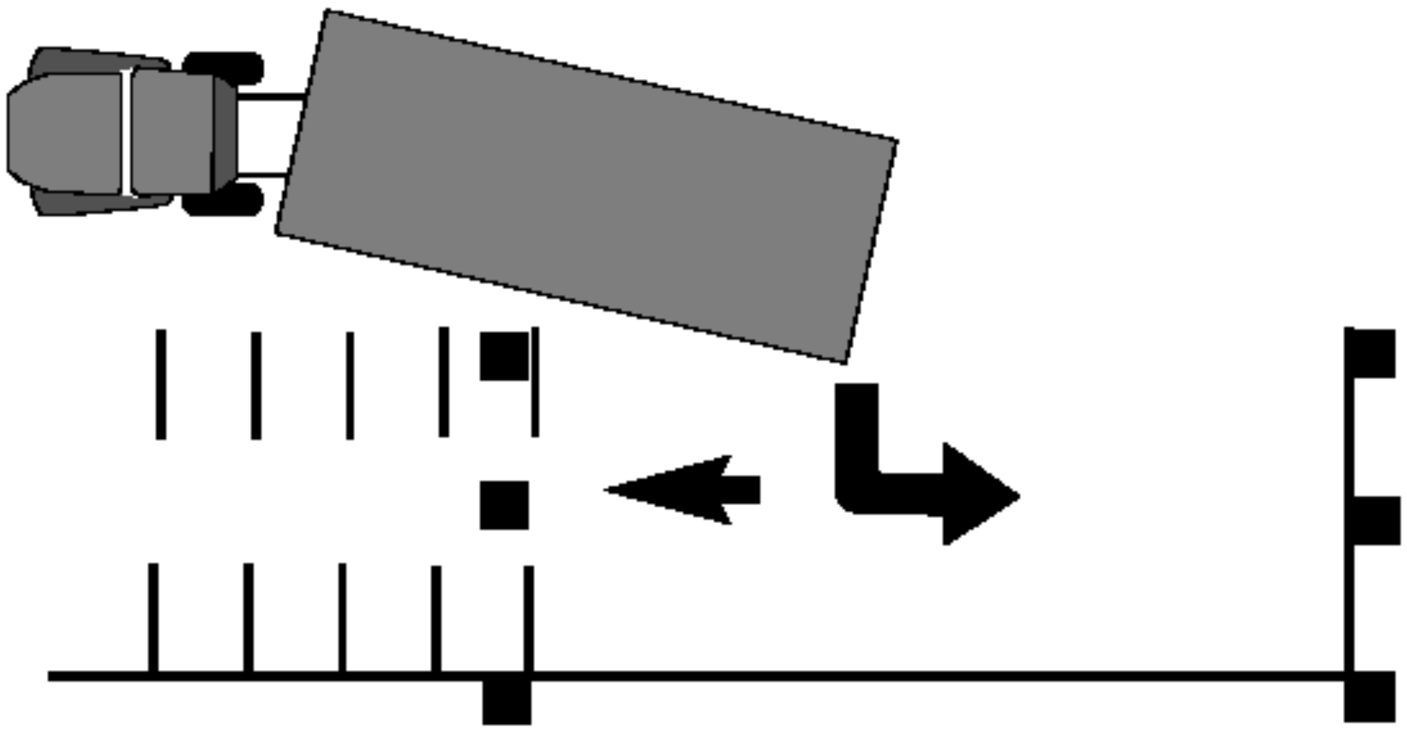
5. Given initial and final state, cost function

$$J(x(t), u(t)) = \int c(x(t), u(t))dt$$

6. Compute action trajectory that satisfies boundary conditions, stays in free state space and minimizes cost.

Differential constraints make things even harder





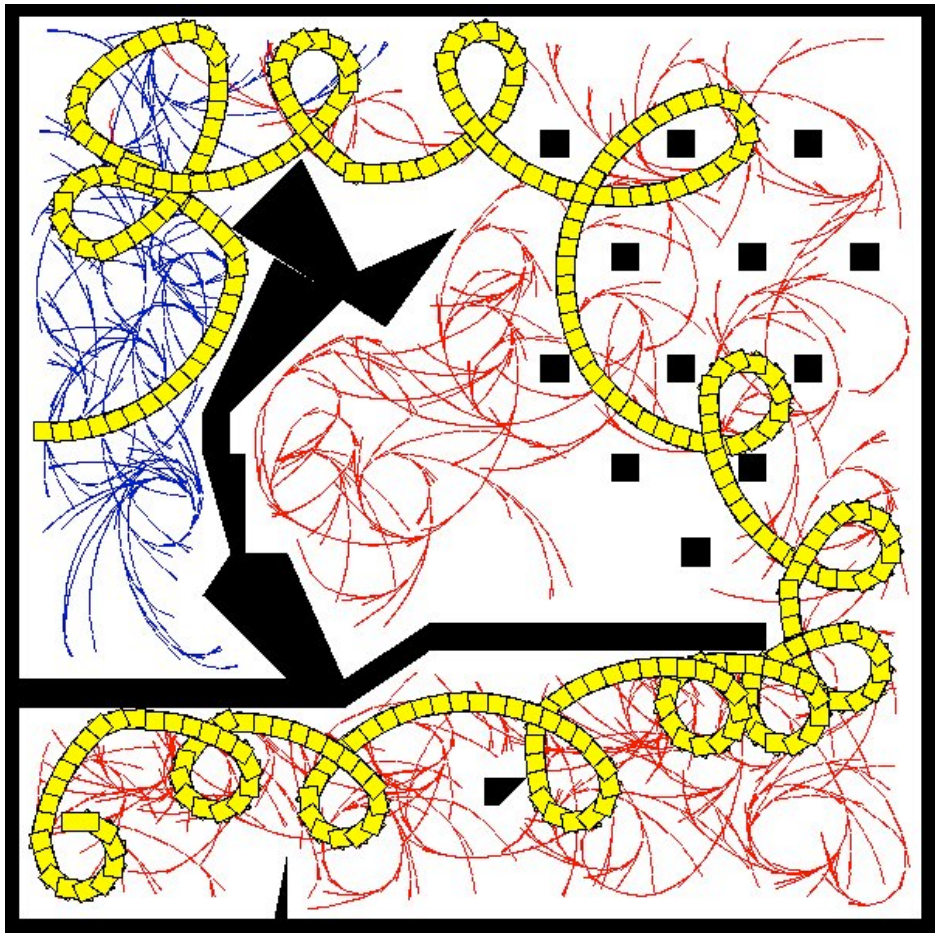
These are examples of non-holonomic system

non-holonomic differential constraints are not completely integrable

i.e. the system is trapped in some sub-manifold of the config space

Differential constraints make things even harder

Emergency landing where UAV can only turn left



"Left-turning-car"

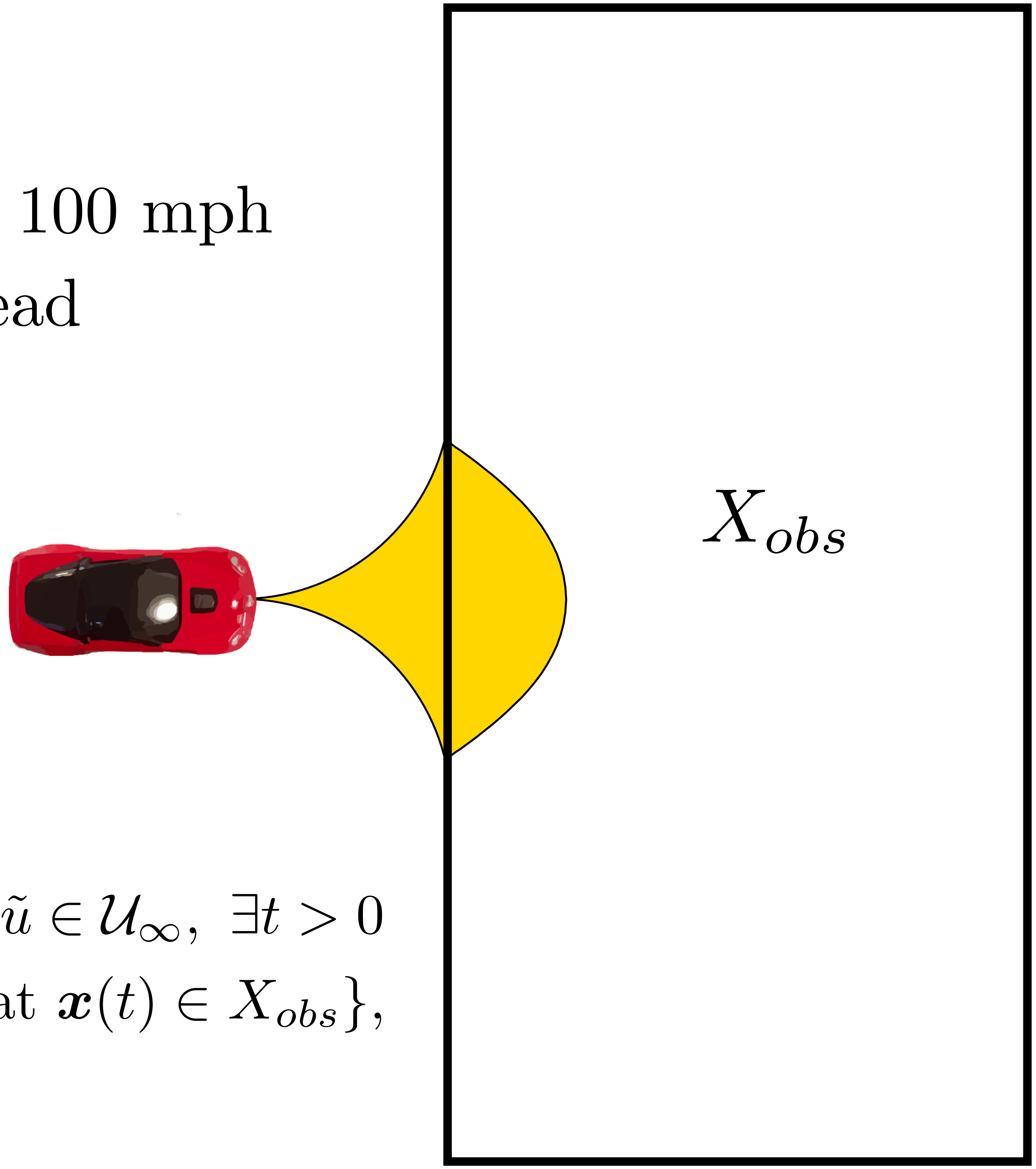
These are examples of non-holonomic system

non-holonomic differential constraints are not completely integrable

i.e. the system is trapped in some sub-manifold of the config space

Regions of inevitable collision

Consider a car going at 100 mph towards a wall 10 m ahead



$$X_{ric} = \{\boldsymbol{x}(0) \in X \mid \text{ for any } \tilde{u} \in \mathcal{U}_\infty, \ \exists t > 0 \text{ such that } \boldsymbol{x}(t) \in X_{obs}\},$$

Research in Motion Planning:

Tractable approximations with provable guarantees