Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN Faculty
Department of Informatics

Introduction to Robotics

Lecture 09

Michael Görner, Jianwei Zhang
[goerner, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

June 19, 2020

Outline

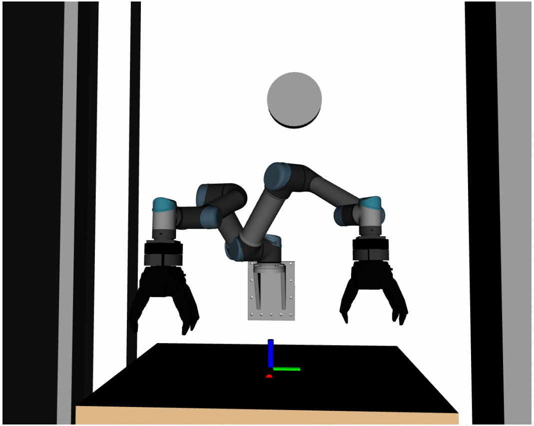
Outline (cont.)

From A to B

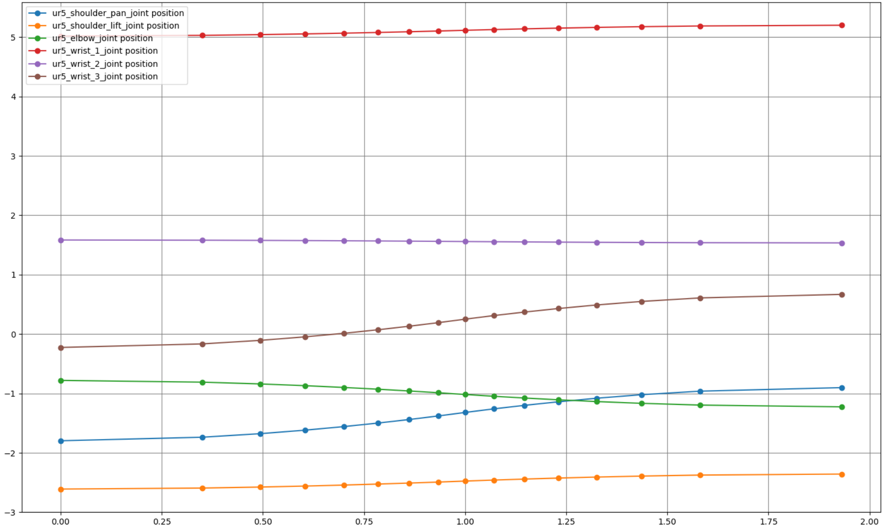
Problem: Generate a continuous trajectory from state A to state B

Approach from previous lectures:
Generate *quintic B-Splines* from A to B:

- Trapezoidal time parameterization
- Minimum jerk parameterization
- Time-optimal motion parameterization

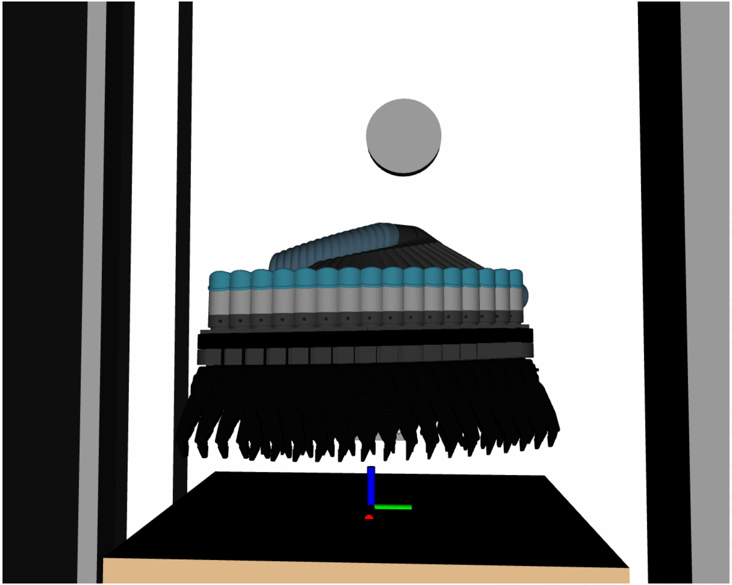


UR5 setup with exemplary start and goal states



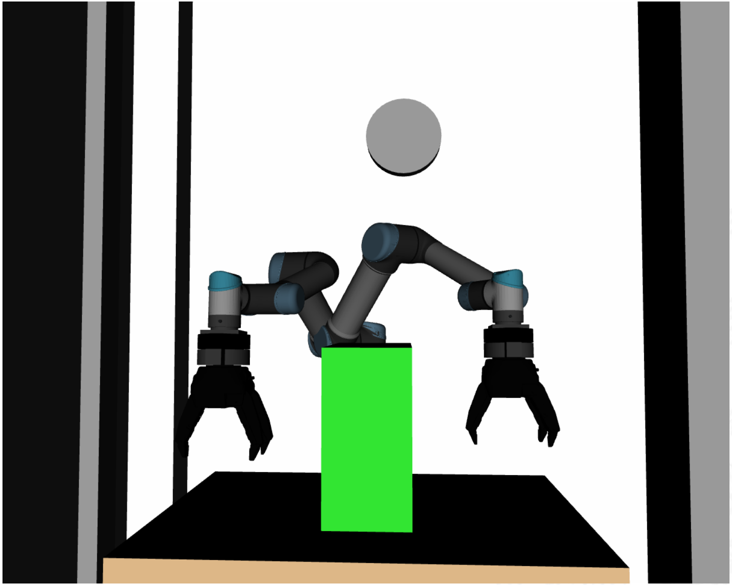
Generated splines of trapezoidal trajectory

From A to B - Trajectory Generation (2)



All waypoints of generated trapezoidal trajectory

From A to B?



Start and Goal state with box obstacle

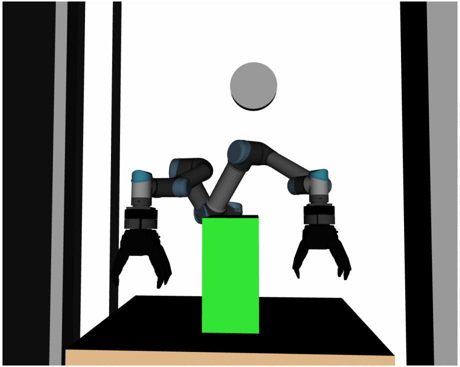
From A to B

If the path is **blocked**, the generated trajectory is **invalid/infeasible** and should not be executed!

Typical obstacles include:

- Walls / Tables
- Robot links
- Objects (to be manipulated)
- Humans

Getting this right is harder than it looks.

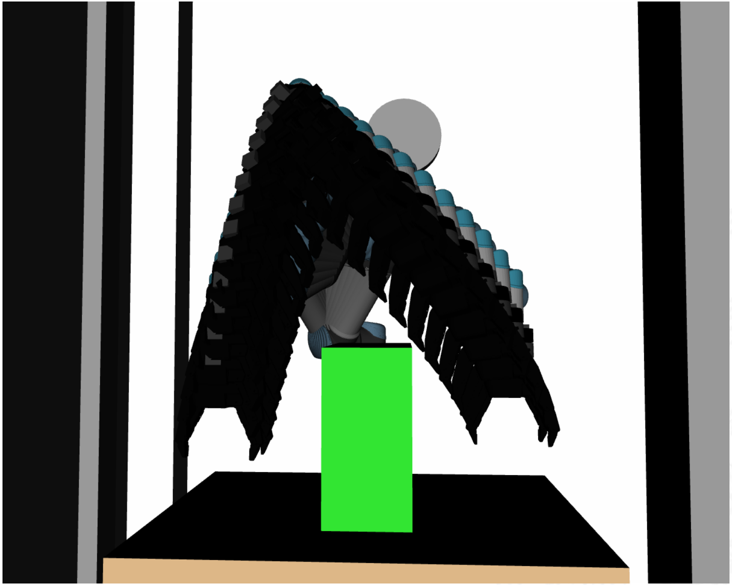


Start and Goal state with box obstacle

Infeasible Trajectories

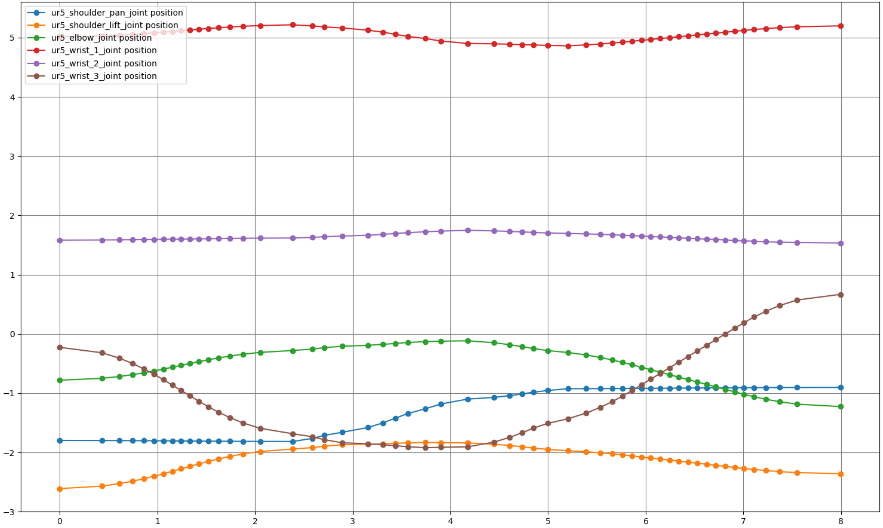
Shadow Hand rammed into styrofoam table

From A to B



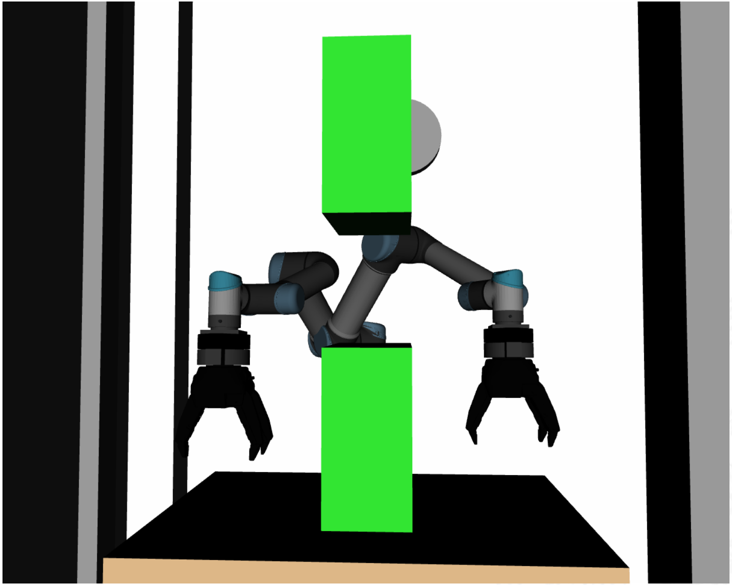
All waypoints of collision-free trajectory

From A to B



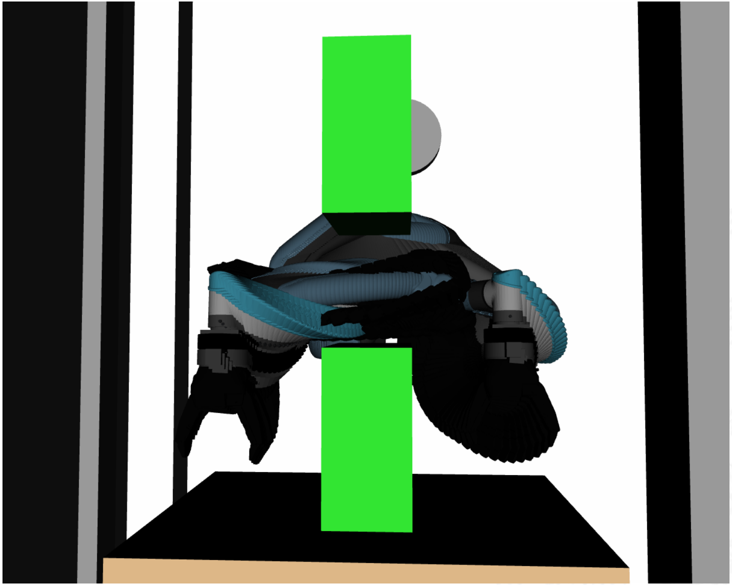
Splines of collision-free trajectory

From A to B



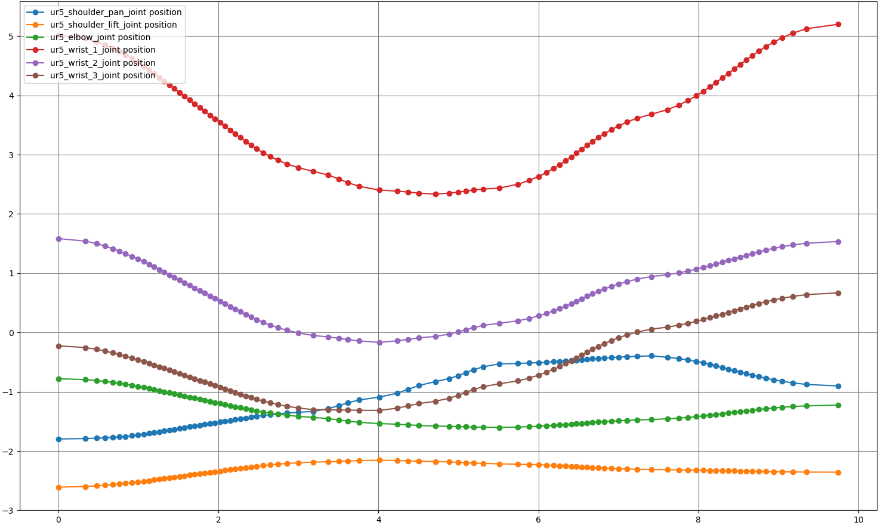
Workspace with two box obstacles

From A to B



All waypoints of collision-free trajectory

From A to B



Splines of collision-free trajectory

Some Criteria for Feasible Trajectories

Feasible trajectories have to satisfy hard geometric constraints.

The most important criterion is a **collision-free** trajectory.

- Collisions between parts of the robot (*self collisions*)
- Collisions with the environment

Countless other criteria can also be important:

- Carrying a container with liquid, no liquid must spill
- Spraying color on a workpiece, the nozzle must always point at the piece
- Getting close or moving directly towards humans

Most of these constraints define *Constraint Manifolds* in the full planning space.
This lecture focuses on collision-aware planning.

Some Criteria for Feasible Trajectories

Feasible trajectories have to satisfy hard geometric constraints.

The most important criterion is a **collision-free** trajectory.

- Collisions between parts of the robot (*self collisions*)
- Collisions with the environment

Countless other criteria can also be important:

- Carrying a container with liquid, no liquid must spill
- Spraying color on a workpiece, the nozzle must always point at the piece
- Getting close or moving directly towards humans

Most of these constraints define *Constraint Manifolds* in the full planning space.
This lecture focuses on collision-aware planning.

Some Criteria for Feasible Trajectories

Feasible trajectories have to satisfy hard geometric constraints.

The most important criterion is a **collision-free** trajectory.

- Collisions between parts of the robot (*self collisions*)
- Collisions with the environment

Countless other criteria can also be important:

- Carrying a container with liquid, no liquid must spill
- Spraying color on a workpiece, the nozzle must always point at the piece
- Getting close or moving directly towards humans

Most of these constraints define *Constraint Manifolds* in the full planning space.
This lecture focuses on collision-aware planning.

Path Planning

Feasible Trajectories

Geometry Representations

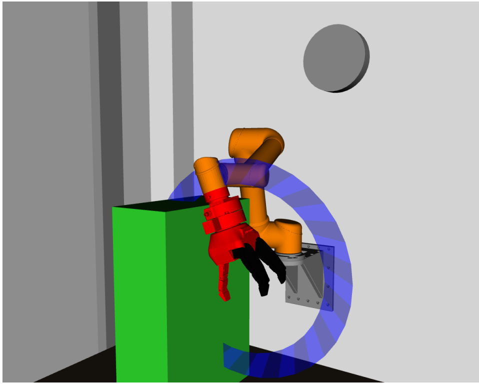
C-Space

Planner Approaches

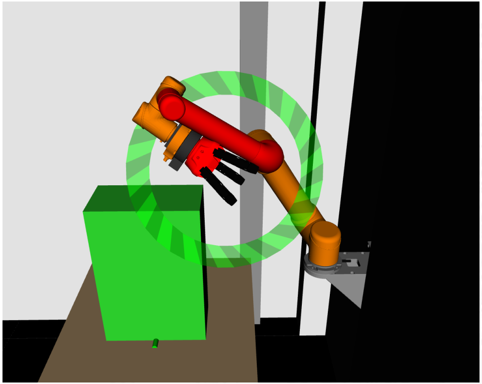
Detecting Collisions

In order to detect expected collisions, we need a geometric **Environment Model**.

- Need to represent all relevant collision shapes
- Trade-off between exact representations and computational load
- Collision tests should run as fast as possible



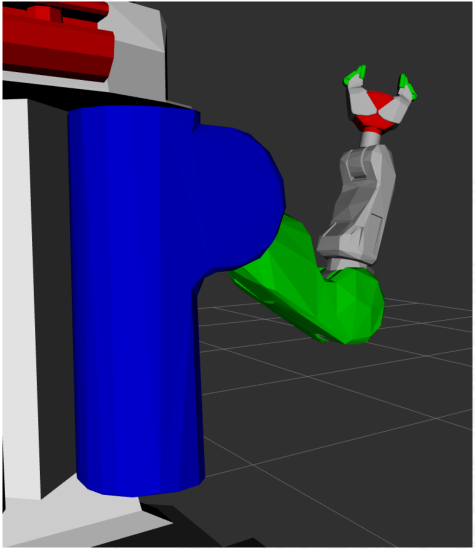
end-effector collision with box



end-effector collision with upper arm link

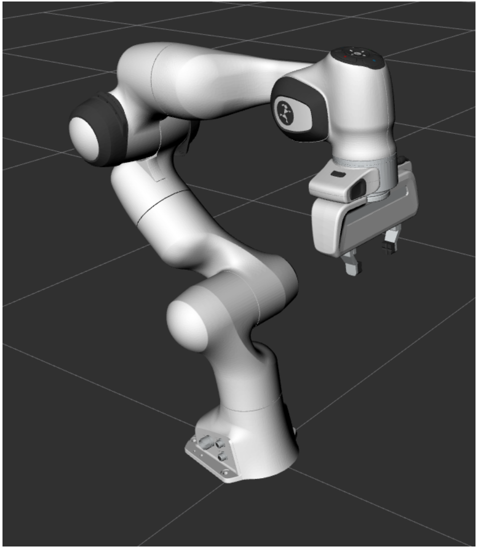
Triangle Meshes

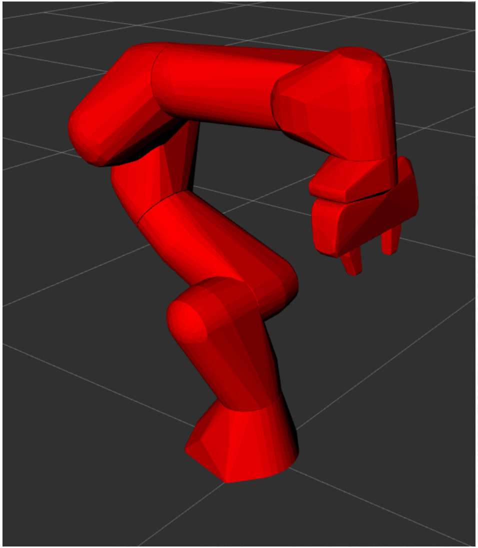
- Standard 3D representation for arbitrary shapes
- General collision checks are costly (Triangle intersection tests)
- Modelled details should depend on required accuracy
- Usually very coarse
- **Convex Meshes** are much more efficient to test. Non-colliding objects can always be separated by a plane.



PR2 left arm mesh representation

Convex Hull Collision Shapes

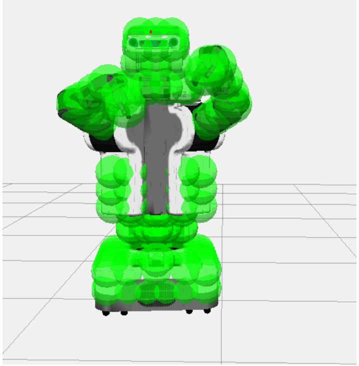


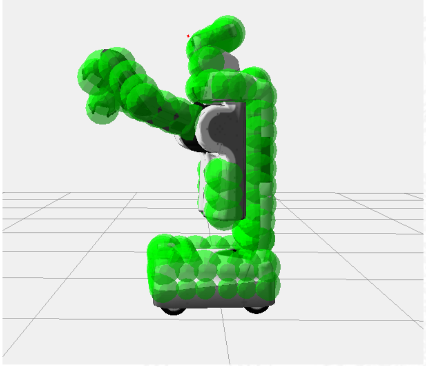


Visual model and convex collision representation of Panda robot arm

Parameters: center point c, radius r.

- Sphere/Sphere collisions afford the cheapest check:
  $\langle c_1, r1 \rangle$ and $\langle c_2, r_2 \rangle$ collided iff $|c_1 - c_2| < r_1 + r_2$
- Sufficient spheres can approximate any shape reasonably accurate:





Approximation of PR2 robot with 139 spheres with radius 10cm

Cowley 2013 [1]

Geometric Primitives

Primitive analytical shapes can be used for more accurate descriptions:

- **Cube**: pose p, scales for 3 axes
- **Cylinder**: pose p, radius r, height h
- **Cone**: pose p, radius r, height h
- **Plane**: pose p

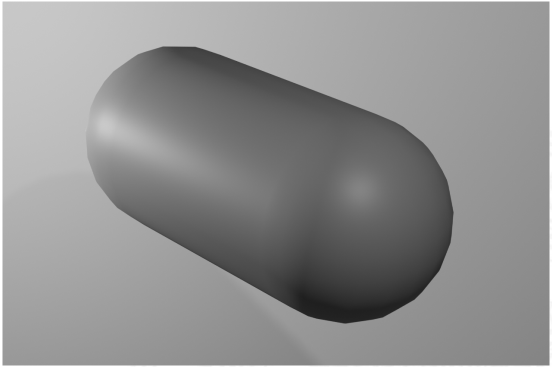
Many analytical shapes allow for faster collision checks.

Capsules

Capsules comprise two half-spheres and a connecting cylinder.

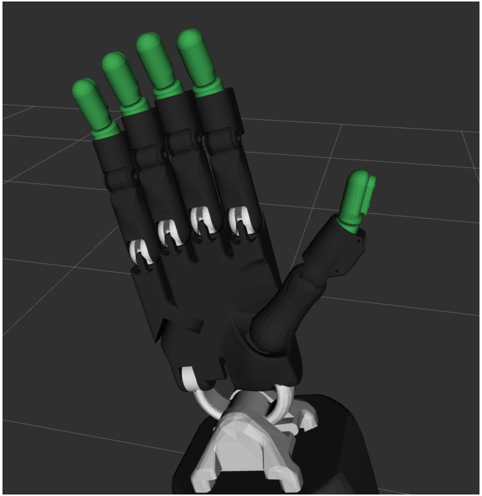
Less common analytical shape, supported in many robotics contexts.

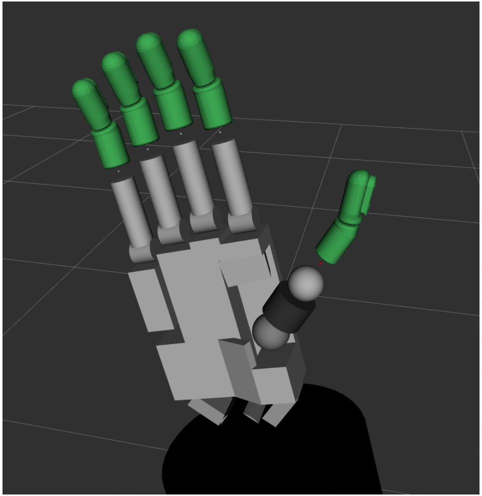
Parameters: pose p, radius r, height h, optionally scale parameters



A primitive capsule

Hybrid Models





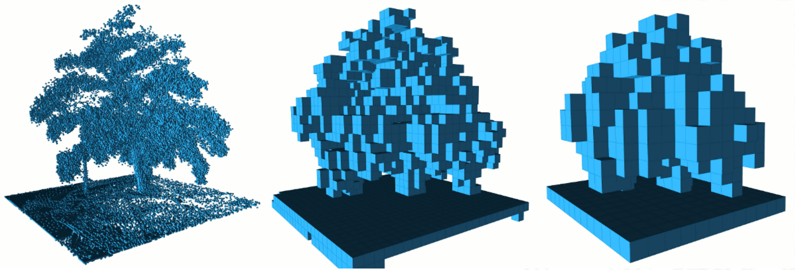
Visual and collision model of a Shadow Dexterous Hand with tactile fingertips

Voxelgrids / Octomaps

All analytical shapes require geometric knowledge about the scene.
Octomaps represent sensor data (depth measurements) directly

- Keeps geometric structure
- Sparse representation
- Efficient updates

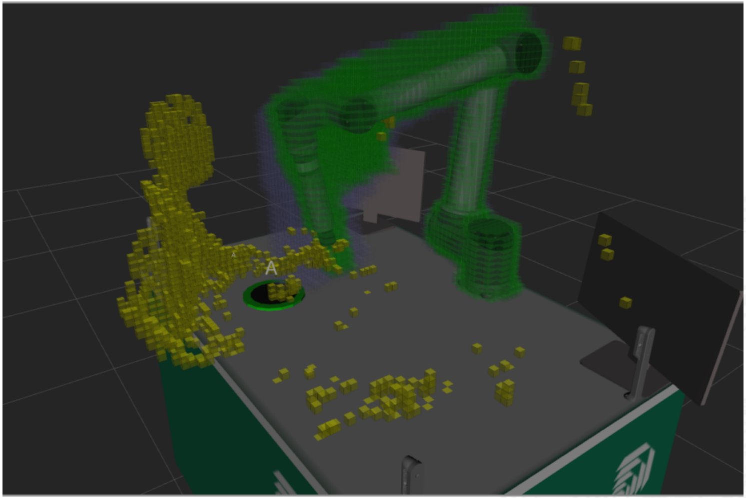
Parameters: pose p, minimal voxel resolution r, datapoints



Octomap representation of a tree at different resolutions

A. Hornung et.al. 2013 [2]

Voxelgrids / Octomaps (2)



Voxel representation of a human interacting with a UR10 robot

Collision Detection Summary

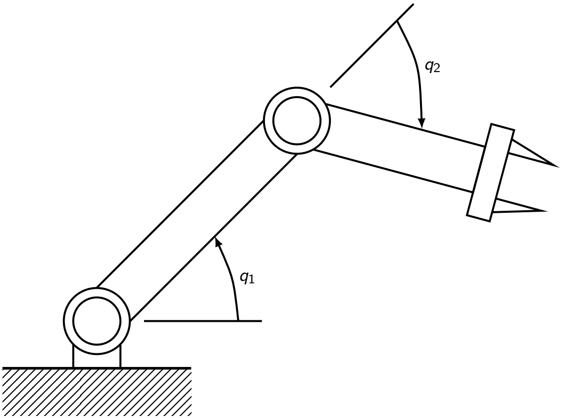
- Hybrid models allow to trade-off computation time and accuracy
- Requires collision checks between each pair of types of collision body

Huge amount of background literature and research in 3D Computer Graphics.
Collision checking in full scenes can be optimized much further optimization:

- Broadphase-collision checking
- Convex decompositions
- Hardware-accelerated checking
- ...

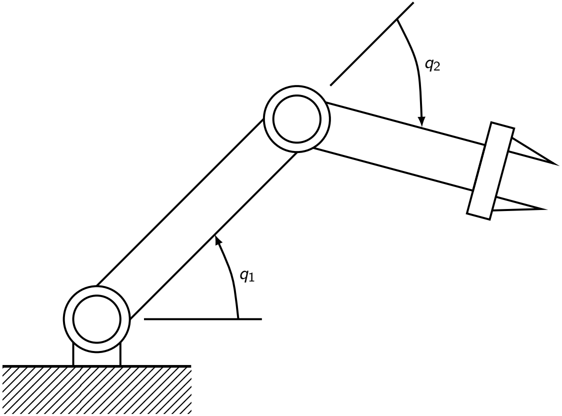
Path Planning

Workspace And Configuration Space – Illustration

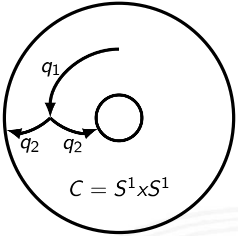


2dof robot model

Workspace And Configuration Space – Illustration

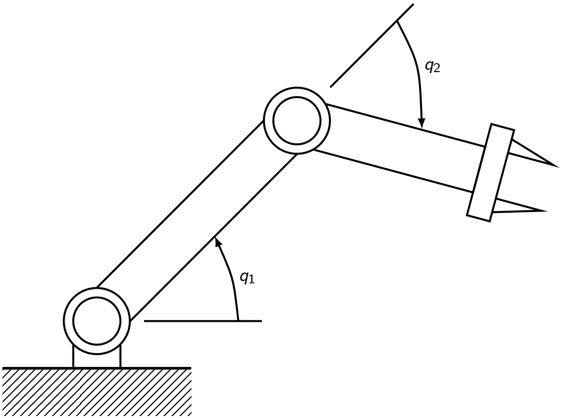


2dof robot model

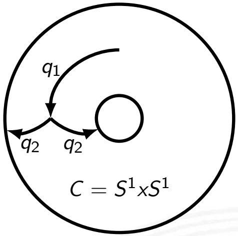


reachable workspace

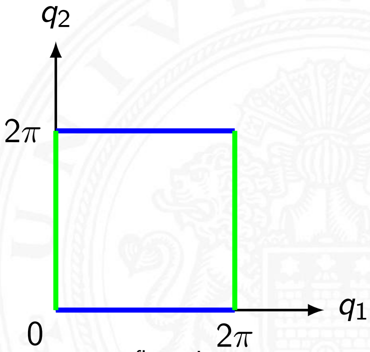
Workspace And Configuration Space – Illustration



2dof robot model



reachable workspace



configuration space

Definition

The parameters that define the configuration of the system are called **Generalized Coordinates**, and the vector space defined by these coordinates is called the **Configuration Space** $\mathcal{X}$.

In robotics, generalized coordinates include

- Joint positions for each controlled joint
- Cartesian poses for mobile robots

$\mathcal{X}_{obs} \subset \mathcal{X}$ describes the set of all configurations in collision.

$\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$ describes the collision-free planning space.

Configuration Space

Definition
The parameters that define the configuration of the system are called **Generalized Coordinates**, and the vector space defined by these coordinates is called the **Configuration Space** $\mathcal{X}$.

In robotics, generalized coordinates include

- Joint positions for each controlled joint
- Cartesian poses for mobile robots

$\mathcal{X}_{obs} \subset \mathcal{X}$ describes the set of all configurations in collision.

$\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$ describes the collision-free planning space.

Definition
The parameters that define the configuration of the system are called **Generalized Coordinates**, and the vector space defined by these coordinates is called the **Configuration Space** $\mathcal{X}$.

In robotics, generalized coordinates include

- Joint positions for each controlled joint
- Cartesian poses for mobile robots

$\mathcal{X}_{obs} \subset \mathcal{X}$ describes the set of all configurations in collision.

$\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$ describes the collision-free planning space.

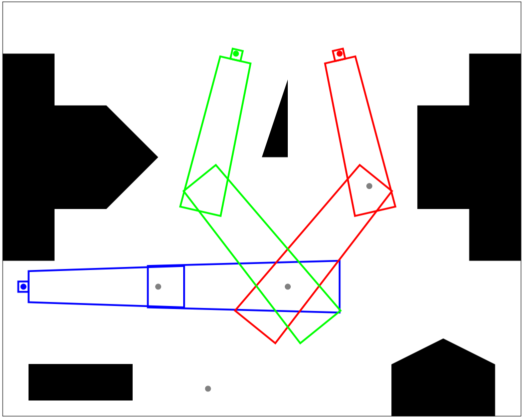
Workspace and Configuration Space

Whereas all intuitive reasoning and system description takes place in the **Workspace**, planning usually proceeds in the **C-space**.

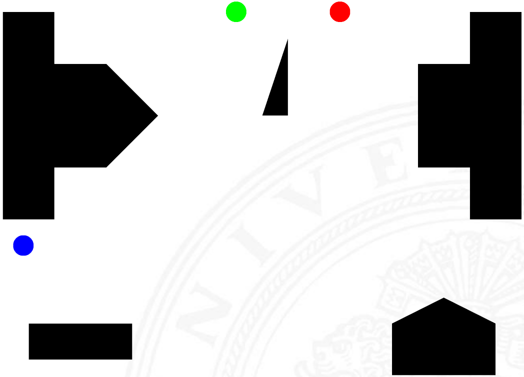
Confusing terminology:

- The workspace is often referred to as reachable **Cartesian space**.
- Configuration space is often shortened to **C-space**.
- For mobile robots, Cartesian poses can be (part of) the C-space.

Workspace to Configuration Space – Example

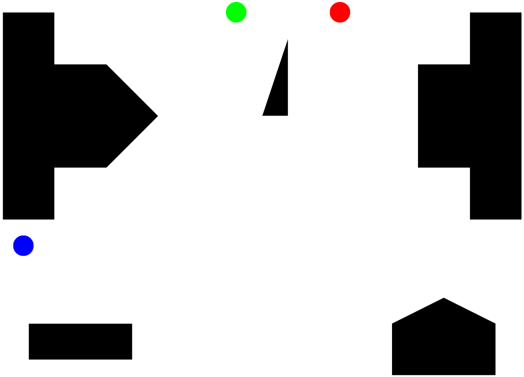


Workspace scheme with multiple states

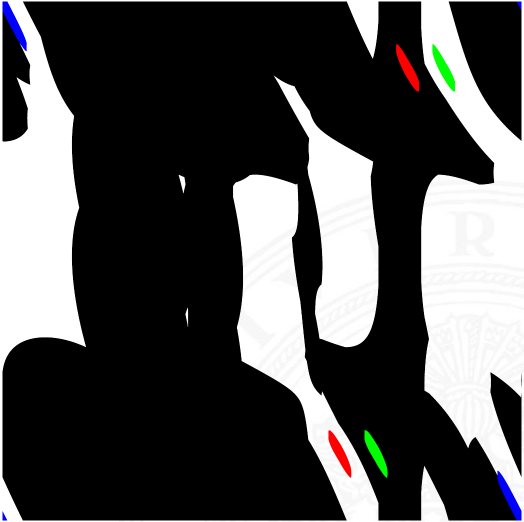


Workspace with target end-effector regions

Workspace to Configuration Space – Example



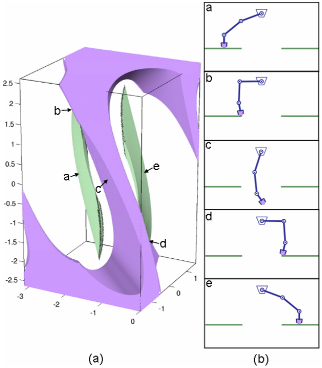
Workspace with target end-effector regions



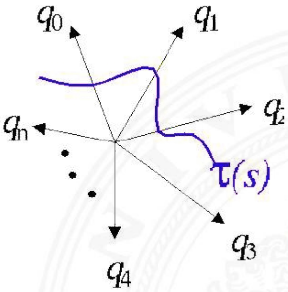
Configuration space with same target regions

On Dimensionality

- Workspaces (position-only) are described by 2 or 3 dimensions
- Effective C-spaces have 6 or more dimensions



C-space visualization for simulated 3dof arm



Trajectory in n-dimensional C-space

D. Berenson et.al. 2009 [3]

C-Space Summary

- The parameters of a system, i.e. **Generalized Coordinates**, span a vector space
- This space is called the **C-space** $\mathcal{X}$ of the system

- $\mathcal{X}_{free}$ describes the collision-free subspace of $\mathcal{X}$
- $x \in \mathcal{X}_{free}$ can be tested by collision-checking
- Usually the space is not parameterized (can not be easily described)

- Cartesian space and C-space can coincide in navigation tasks where only the pose of the robot is a parameter

Definition

A **Path Planning Problem** is described by a triple $\langle \mathcal{X}_{free}, x_{start}, \mathcal{X}_{goal} \rangle$, where

- $x_{start} \in \mathcal{X}_{free}$ is the start state
- $\mathcal{X}_{goal} \subset \mathcal{X}$ describes a goal region

Definition

A mapping $\tau : [0,1] \to R^n$ onto a C-space $\mathcal{R}^n$ is called a

- **Path** if it describes a finite, continous trajectory.
- **Collision-free Path** if $Range(\tau) \subseteq \mathcal{X}_{free}$
- **Feasible Path** if it is collision-free, $\tau(0) = x_{start}$, and $\tau(1) \in \mathcal{X}_{goal}$

adapted from S. Karaman et.al. 2011 [4]

Feasible Path Planning

Feasible Path Planning requires planners to find a **feasible path** for any given path planning problem. The ideal planner is

- **correct** - all reported paths are feasible
- **complete** - if a feasible path exist, it will be found
- performs with **bounded runtime** - if no path exists, it will fail

In practice,

- correctness is often traded for feasible runtime performance.
- *actual* correctness is defined by the real world, not by the planning model. If an object is not modelled, it will not be considered.
- most methods can not report failures and are only asymptotically complete.

Feasible Path Planning

Feasible Path Planning requires planners to find a **feasible path** for any given path planning problem. The ideal planner is

- **correct** - all reported paths are feasible
- **complete** - if a feasible path exist, it will be found
- performs with **bounded runtime** - if no path exists, it will fail

In practice,

- **correctness** is often traded for feasible runtime performance.
- *actual* correctness is defined by the real world, not by the planning model. If an object is not modelled, it will not be considered.
- most methods can not report failures and are only asymptotically complete.

Feasible Path Planning

Feasible Path Planning requires planners to find a **feasible path** for any given path planning problem. The ideal planner is

- **correct** - all reported paths are feasible
- **complete** - if a feasible path exist, it will be found
- performs with **bounded runtime** - if no path exists, it will fail

In practice,

- **correctness** is often traded for feasible runtime performance.
- *actual* correctness is defined by the real world, not by the planning model. If an object is not modelled, it will not be considered.
- most methods can not report failures and are only asymptotically complete.

Feasible Path Planning

Feasible Path Planning requires planners to find a **feasible path** for any given path planning problem. The ideal planner is

- **correct** - all reported paths are feasible
- **complete** - if a feasible path exist, it will be found
- performs with **bounded runtime** - if no path exists, it will fail

In practice,

- **correctness** is often traded for feasible runtime performance.
- *actual* correctness is defined by the real world, not by the planning model. If an object is not modelled, it will not be considered.
- most methods can not report failures and are only asymptotically complete.

Feasible Path Planning

Feasible Path Planning requires planners to find a **feasible path** for any given path planning problem. The ideal planner is

- **correct** - all reported paths are feasible
- **complete** - if a feasible path exist, it will be found
- performs with **bounded runtime** - if no path exists, it will fail

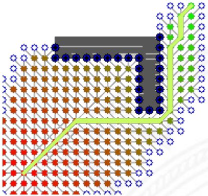
In practice,

- **correctness** is often traded for feasible runtime performance.
- *actual* correctness is defined by the real world, not by the planning model.
  If an object is not modelled, it will not be considered.
- most methods can not report failures and are only asymptotically complete.

Discretized Space Planning

Simple Idea: Discretize planning space & run A^* on the resulting grid

- Classical path search algorithm
- Returns optimal plan in grid
- Works well for planar path planning



A^* planner finding an optimal path in the grid

Not Tractable in 6+ dimensions

- Solutions limited to grid resolution
- Sufficiently high resolution required for correctness/completeness

- Discretization explicitly represents the whole space volume
- Curse-of-Dimensionality:
  - assuming 1 deg resolution and 360 deg joint range
  - 2 joints yield 129600 unique states
  - 3 joints yield 46656000 unique states
  - 6 joints yield $\sim 2.18e15$ unique states
- Explicit representation of the whole space is clearly not feasible.

Potential Field Method (PFM)

Alternative Idea: Represent space entirely through continuous function $f : R^n \rightarrow R$.

- No explicit space representation
- Can be evaluated as needed

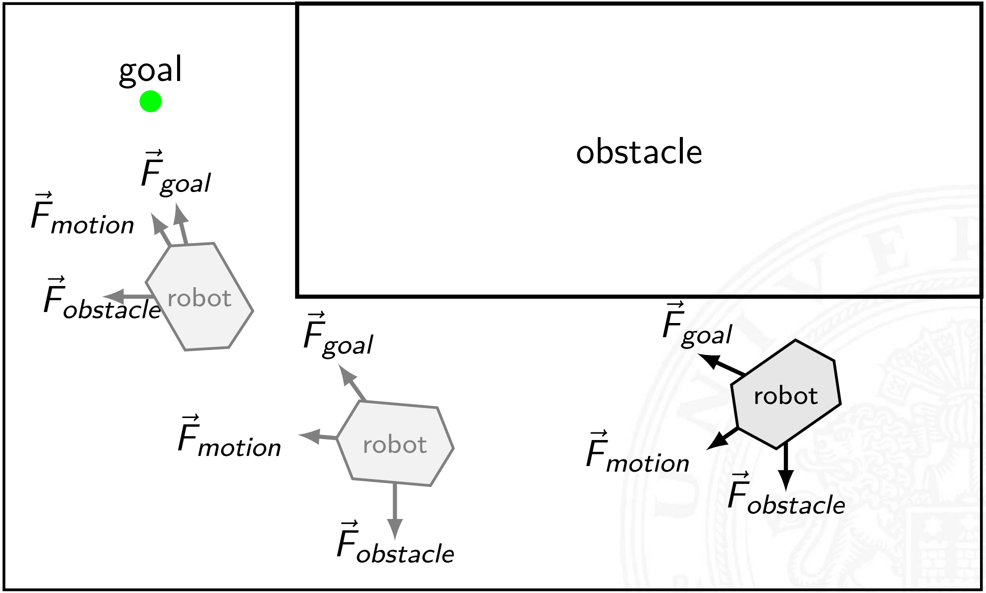
Khatib 1986:

> *The manipulator moves in a field of forces. The position to be reached is an attracting pole for the end effector and obstacles are repulsive surfaces for the manipulator parts.* [5]

Basic Principle

- Initially developed for real-time collision avoidance
- Potential field associates a scalar value $f(p)$ to every point p in space
- Robot moves along the negative gradient $-\nabla f(p)$, a "force" applied to the robot
- f's global minimum should be at the goal configuration
- An ideal field used for navigation should
  - be smooth
  - have only one global minimum
  - the values should approach ∞ near obstacles

Basic Principle (cont.)



Basic Principle (cont.)

- The attracting force (of the goal)

$$\vec{F}_{goal}(\mathbf{p}) = -\kappa_\rho(\mathbf{p} - \mathbf{p}_{goal})$$

  - where
    κ_ρ is a constant gain factor

Basic Principle (cont.)

- The potential field (of obstacles)

$$U(\mathbf{x}) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(\mathbf{p})} - \frac{1}{\rho_0}\right)^2 & \text{if } \rho(\mathbf{p}) \leq \rho_0 \\ 0 & \text{else} \end{cases}$$

  - where
    η is a constant gain factor
    $\rho(\mathbf{p})$ is the shortest distance to the obstacle O
    ρ_0 is a threshold defining the region of influence of an obstacle

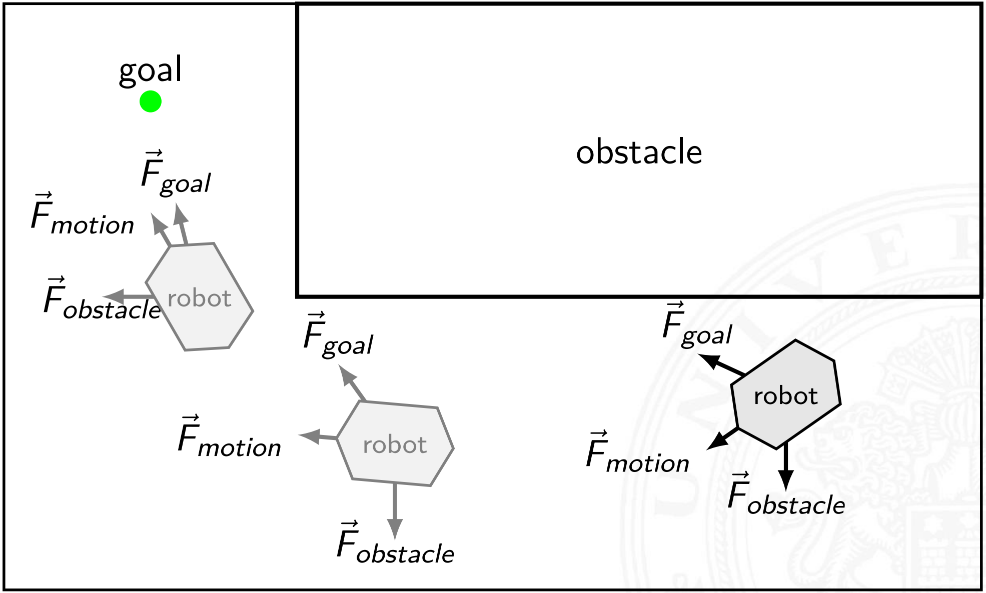
Basic Principle (cont.)

- The repulsive force of an obstacle

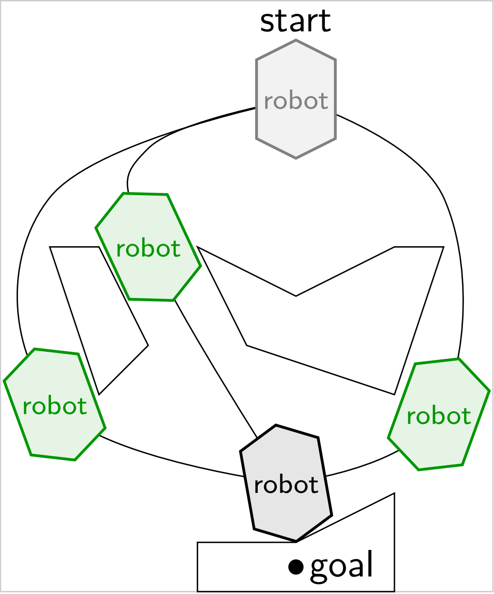
$$\vec{F}_{obstacle}(\mathbf{p}) = \begin{cases} \eta(\frac{1}{\rho(\mathbf{p})} - \frac{1}{\rho_0})\frac{1}{\rho(\mathbf{p})^2}\frac{d\rho(\mathbf{p})}{d\mathbf{p}} & \text{if } \rho(\mathbf{p}) \leq \rho_0 \\ 0 & \text{if } \rho(\mathbf{p}) > \rho_0 \end{cases}$$

  - where $\frac{d\rho(\mathbf{p})}{d\mathbf{p}}$ is the partial derivative vector of the distance from the point to the obstacle.

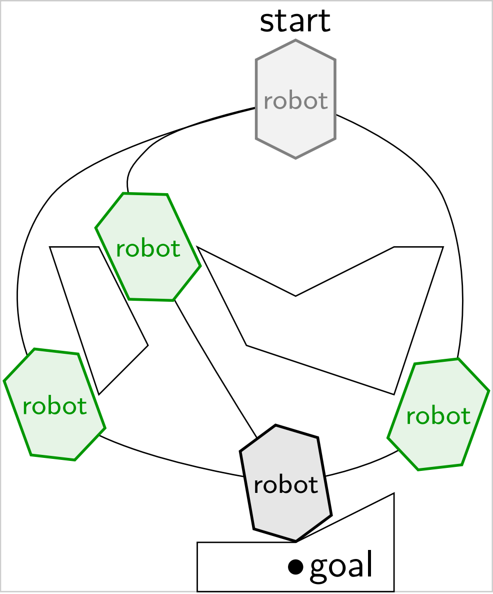
Basic Principle

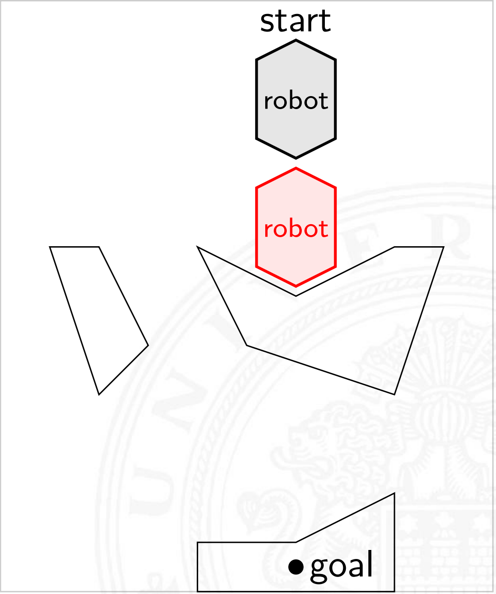


Local Minima of PFM



Local Minima of PFM





Advantages and Disadvantages of PFM

Advantages:

- Implicit State Representation
- Real-time capable

Advantages and Disadvantages of PFM

Advantages:

- Implicit State Representation
- Real-time capable

Disadvantages:

- Incomplete algorithm
  - Existing solution might not be found
  - Calculation might not terminate if no solution exists
- $\rho(p)$ is only intuitive in 2D and 3D
- Obstacles in 6D C-space have complex shapes