Motion Planning

Organization, Introduction, Problem Formulation

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

April 17, 2024

Intro Lecturer: Wolfgang Hönig

- Assistant Professor (CS; since 04/2023)
  - Group: Intelligent Multi-Robot Coordination Lab (`https://imrclab.github.io`)
- Research on multi-robot systems, aerial robots, and motion planning

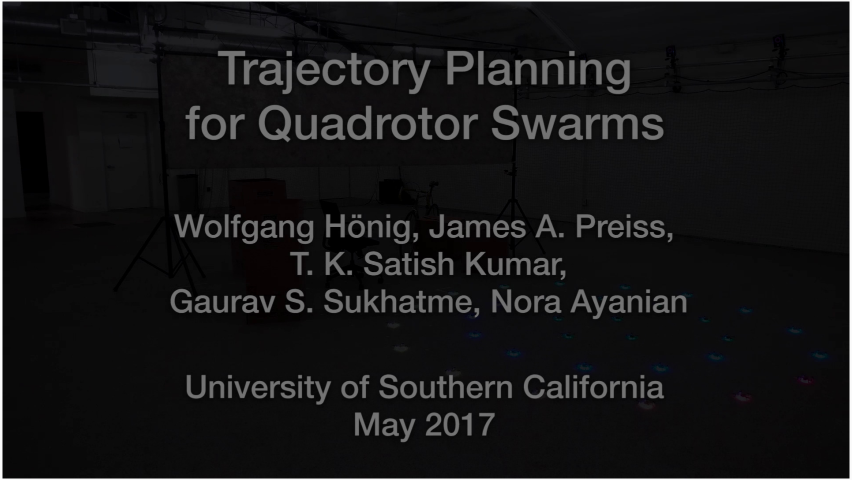
Academic Experience

Technische Universität Dresden	Diploma CS
USC University of Southern California	MS/PhD CS (Robotics)
Caltech	Postdoc Aerospace
Technische Universität Berlin	Group Leader CS

Industry Experience

NVIDIA	Intern Software Engineer
amazon robotics	Intern
bitcraze	Visiting Researcher

Video



https://youtu.be/7KIa9FlmbRc

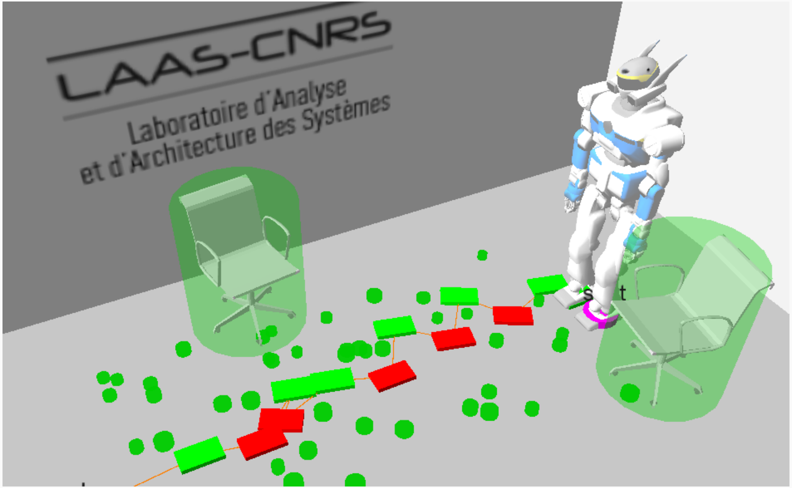
Intro Lecturer: Andreas Orthey

- Staff Robotics Scientist at Realtime Robotics (since 07/2021)
- Research on humanoid robotics, and abstractions for motion planning

Academic/Industrial Experience

Technische Universität Berlin	Bsc/Msc CS (Technische Informatik)	Alexander von Humboldt Stiftung/Foundation	Postdoc Fellow
LAAS CNRS	PhD Robotics	Max-Planck-Institut für Intelligente Systeme	Postdoc
AIST	Postdoc Fellow	realtime robotics	Staff Robotics Scientist

Video



https://www.youtube.com/watch?v=moAXBesmL8A

Intro Teaching Assistants: Akmaral Moldagalieva, Khaled Wahba, Pia Hanfeld

- PhD students in my group (since 2022)
- Research on motion planning, controls, and safety for multirotor teams

Organization

Class Format

- Weekly 90 min lectures (Wednesday 8:15am – 9:45am)
  - Prepared jointly, held by one of the lecturers
- Weekly exercises
  - Theoretical part and occasionally programming part
  - 10 % of the grade; exam preparation
- Weekly 90 min discussion sessions (Thursday 4:15pm – 5:45pm)
  - Discuss exercise problems
  - Interactive (rather than presenting solutions), so come prepared
- Four programming assignments
  - Individual work
  - 40 % of the grade
- Office hours by appointment

Class material, communication, feedback, submission via ISIS and GitLab.

Grading

- Portfolio format
- Exercises (0, 0.5, or 1 % per exercise; up to 10 % total), by
  - Presenting your solution of a task in the discussion session (1 %); -OR-
  - Uploading your individual solution via ISIS (0, 0.5, or 1 %)
  - The first option is less work and encourages active participation during the discussion session.
- Programming assignments (40 %)
  - Due every 3 weeks (Wednesday, 6pm, via pushing a GitLab tag)
  - Individual work
  - Topics: Collision checking, sampling-based motion planning, OMPL, optimization-based motion planning
- Written exam (50 %)
  - 1h, at the end of the semester
  - Covers topics from the exercises and programming assignments

Prerequisites

Programming

- Experience in Python or C++
- Ability to read pseudo code, e.g.,

```
def Astar(G, d, v_s, v_z, h):
  O = queue()
  while O ≠ ∅:
    # smallest f-value
    v = O.pop()
    if v = v_z:
      return solution
    for n in v.neighbor:
      # ...
```

Math

- Set theory, e.g.

$$\mathcal{W} \setminus (\cup_{i=1}^{N} \mathcal{O}_i)$$

- Probability Theory

$$P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)}$$

- Linear Algebra and Calculus

Not sure?

Prof. Toussaint has a great tutorial.

Syllabus

Foundations

2 Weeks (problem formulation, terminology, collision checking)

Search-based

2 Weeks (A* and variants; state-lattice-based planning)

Sampling-based

5 Weeks (RRT, PRM, OMPL, Sampling Theory)

Optimization-based

2 Weeks (SCP, TrajOpt)

Current and Advanced Topics

2 Weeks (Comparative Analysis, Machine Learning and Motion Planning, Hybrid- and Multi-Robot approaches)

Two holidays: Wednesday (May 1st), Thursday (May 9th)

Related Classes

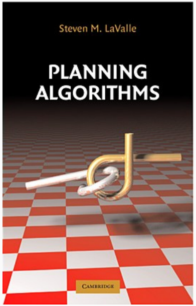
Classes at TU Berlin

- Optimization Algorithms, Prof. Toussaint (MOSES)
  - Focuses on how continuous optimization problems are solved
- Robot Learning, Prof. Toussaint/Hönig (MOSES)
  - Focuses on how to apply learning for robotics problems, including motion planning
- Robotics I+II, Prof. Brock (MOSES)
  - Provides foundation of entire robotics pipeline (sensing, decision making, control)

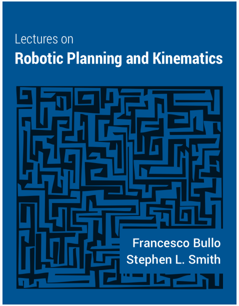
Classes elsewhere

- Motion Planning, Prof. Berenson, University of Michigan (Web)
  - Based on paper reading and projects
- Algorithmic Robotics and Motion Planning, Prof. Kleinbort, Tel Aviv University (Web)
  - Focus on computational geometry

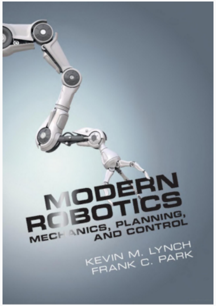
Text Books / Reading Material



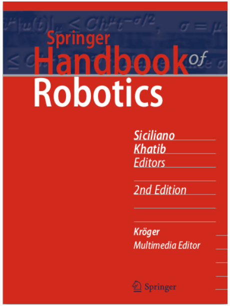
LaValle [1]



Bullo and Smith [2]



Lynch and Park [3, Chapter 10]



Siciliano and Khatib [4, Chapter 7]

Desired Learning Outcome

After the class, you should have/be able to:

- Knowledge of current state-of-the-art algorithms in motion planning
- Decide (theoretically and empirically) which algorithm(s) to use for a given problem
- Implement (basic versions) of the algorithms
- Use current academic and industrial tools such as the Open Motion Planning Library (OMPL)

Introduction

Autonomy Becomes Ubiquitous

Source: Robotics Business Review

Industry 4.0

Source: Bloomberg

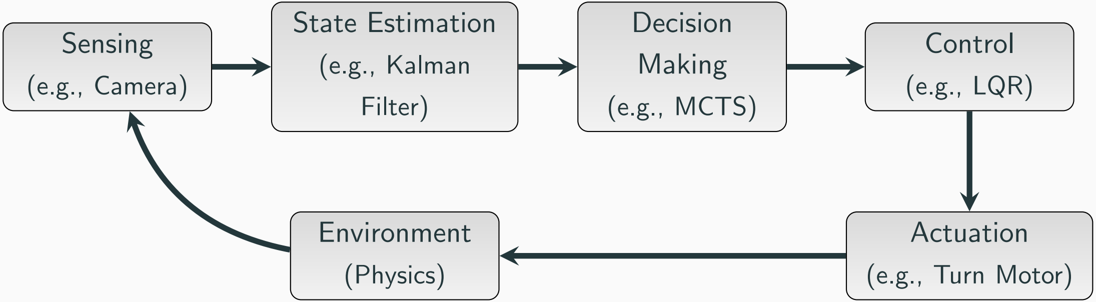
Self-driving Cars

Source: iRobot

Vacuum Robots

What is required to build these systems?

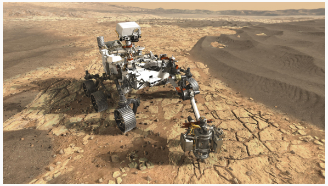
The Classic Robotics Pipeline



Motion Planning is a part of decision making: How to reach the goal, given the state of the robot and environment, without collisions.

Modern robotic systems do not strictly separate parts of the classic robotics pipeline.

Example Applications: Mobile Robots



Source: NASA

Space Exploration

Source: Bloomberg

Self-driving Cars

Source: iRobot

Vacuum Robots

Source: Robotics Business Review

Warehouse Automation

Source: NYT

UAVs

Example Applications: Manipulators

Source: RBR

Manufacturing

Source: RBR

Robotic Surgery

Source: BBC

Cooking Robots

Example Applications: Computer Games / Graphics

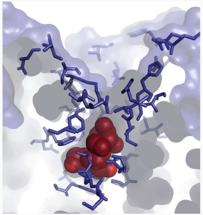
Source: [5]

Crowd Simulation

Source: Wikipedia

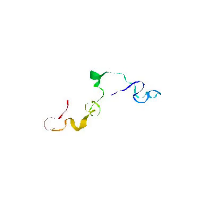
Computer Games

Example Applications: Computational Biology



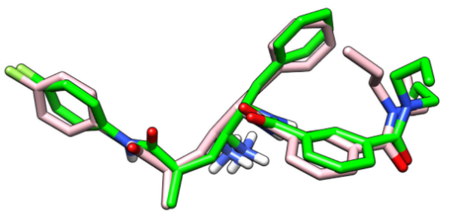
Source: [6]

Molecular Simulations



Source: Parasol Lab

Protein Folding



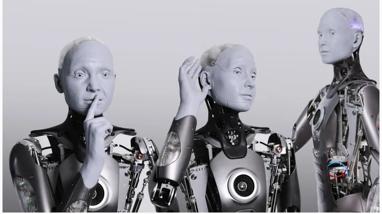
Source: [7]

Drug Design

Example Applications: Hybrid Systems

Source: Boston Dynamics

Mobile Robot With Manipulator



Source: CNET

Humanoids

Motion planning is a fundamental problem

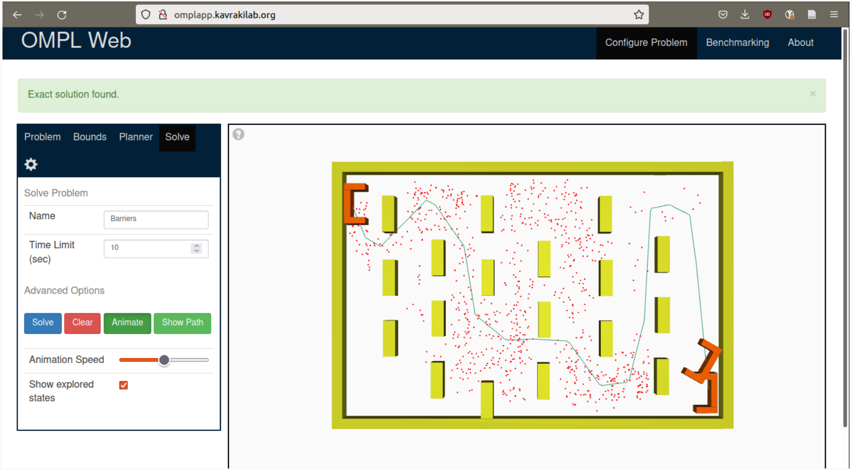
Algorithmic ideas can also be used for other decision-making tasks, where reasoning over both space and time is required.

A (Brief) History of Motion Planning

- 1968: Development of A* for Shakey (Robot at SRI)
- 1979: Complexity of motion planning (PSPACE)
- 1984: Complexity of multi-robot motion planning (PSPACE)
- 1988: Fast collision checking in 3D
- 1996–: Sampling-based Planning
- 2008: Initial version of the Search-based Planning Library (SBPL)
- 2012: Initial version of the Open Motion Planning Library (OMPL)
- 2011: Rediscovery of optimization-based motion planning
- Last decade: hybrid approaches; combination with machine learning

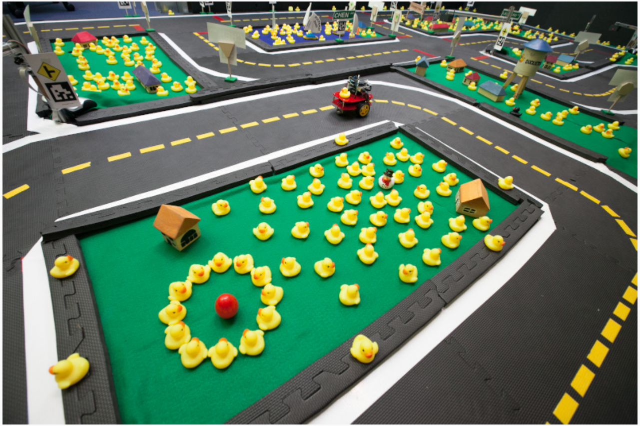
Motion Planning in the Cloud: OMPL.App Web

Demonstration at `http://omplapp.kavrakilab.org/`



Formal Problem Formulation

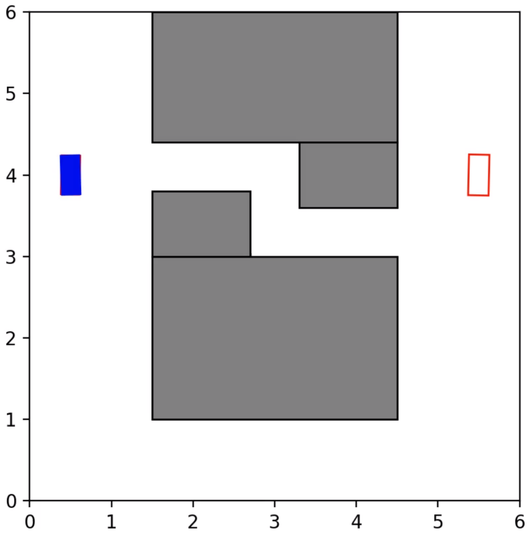
A Simple Example: A (Toy) Car



Source: Duckietown

1. How to move the car in tight spaces in your hand?
2. How to move the car with a remote control?
3. What is the “best” solution?

Some Abstractions



1. Focus on 2 dimensions
2. Bounded workspace (e.g., a room)
3. Car and obstacles are simple shapes (e.g., rectangles, circles, polygons)

Configuration Space (C-Space)

Configuration

Vector that (potentially indirectly) specifies the position of every point of the robot.

Notation: $\mathbf{q}$

Degrees of Freedom (DoF)

The minimum number of real-valued coordinates needed to represent the *configuration*.

Configuration Space (C-Space)

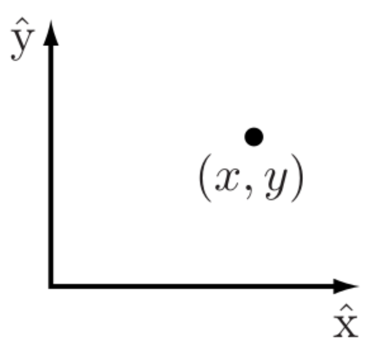
The space containing all possible configurations of the robot.

Notation: $\mathbf{q} \in \mathcal{Q}$

Configuration Space vs. State Space

Often state and configuration are used interchangeably. Notation: $\mathbf{x} \in \mathcal{X}$.

Configuration Space (C-Space): Examples



Configuration

$(x, y) \in \mathbb{R}^2$

Degrees of Freedom (DoF)

2

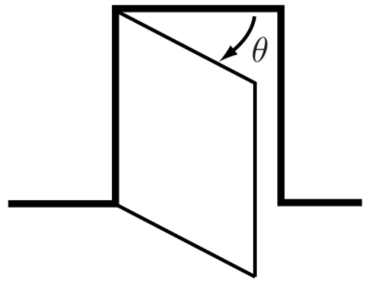
Configuration Space (C-Space)

$\mathbb{R}^2$

Workspace

$\mathbb{R}^2$

Configuration Space (C-Space): Examples



Configuration

$\{(x, y, z) : (x, y, z) \text{ is part of door}\} \subset \mathbb{R}^3$
or $\theta \in [0, \pi/2)$

Degrees of Freedom (DoF)

1

Configuration Space (C-Space)

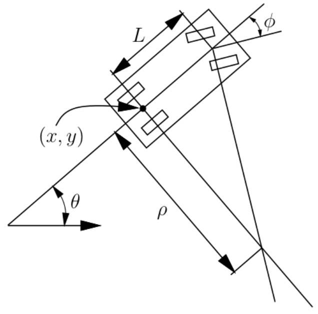
$[0, \pi/2) \subset \mathbb{R}$

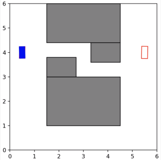
Workspace

$\mathbb{R}^3$

Configuration Space (C-Space): Examples

Top View:





Configuration

(x, y, θ)

Degrees of Freedom (DoF)

3

Configuration Space (C-Space)

$[0, 6] \times [0, 6] \times [0, 2\pi) \subset \mathbb{R}^3$

Workspace

$\mathbb{R}^2$

Configuration Map

Function that specifies the position of each point in the workspace belonging to the robot at a given configuration.

Notation: $\mathcal{B} : \mathcal{Q} \to 2^{\mathcal{W}}$

Circular Robot

at position $\mathbf{q} = (x_0, y_0)$ with radius r

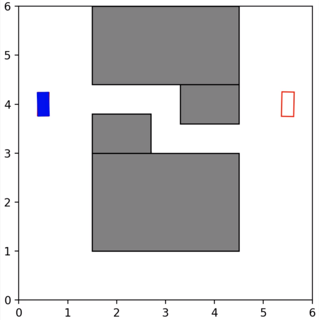
$$\mathcal{B}(\mathbf{q}) = \{(x, y) \in \mathbb{R}^2 : (x - x_0)^2 + (y - y_0)^2 \leq r^2\}$$

Free Workspace

Free Workspace

Set of points in the workspace $\mathcal{W}$ that are outside of obstacles $\mathcal{O}_i$.

$$\mathcal{W}_{free} = \mathcal{W} \setminus (\mathcal{O}_1 \cup \mathcal{O}_2 \ldots \mathcal{O}_n)$$



Geometric Motion Planning: Definition

- Input:
  - Description of the environment
  - Description of the robot shape
  - Initial configuration
  - Goal configuration
- Output: "best" collision-free sequence of configurations from the initial to the goal configuration

Geometric Motion Planning

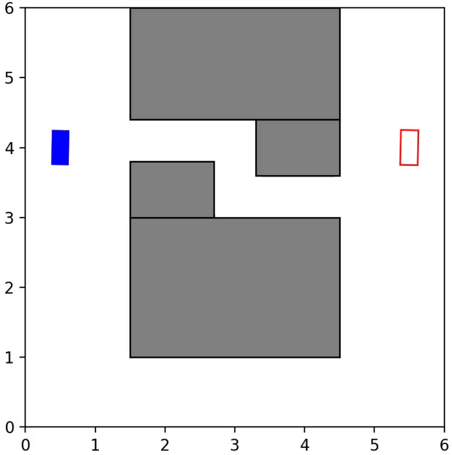
Given initial and final configurations $\mathbf{q}_{start} \in \mathcal{Q}$ and $\mathbf{q}_{goal} \in \mathcal{Q}$, the free workspace $\mathcal{W}_{free}$, the configuration map $\mathcal{B}(\cdot)$, a cost function $J(\cdot)$, we aim to find a sequence of configuration $\mathbf{q} : [0, 1] \rightarrow \mathcal{Q}$:

$$\begin{aligned} \underset{\mathbf{q}(p)}{\text{argmin}} \quad J(\mathbf{q}(p)) \quad \text{s.t.} \\ \mathbf{q}(0) = \mathbf{q}_{start} \\ \mathbf{q}(1) = \mathbf{q}_{goal} \\ \mathcal{B}(\mathbf{q}(p)) \subset \mathcal{W}_{free} \quad \forall p \in [0, 1] \end{aligned}$$

Geometric Motion Planning: Notes

- We often compute a *discrete sequence* of configurations $\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \ldots$, rather than providing an analytical function $\mathbf{q}(p)$
  - Formally, it is important to use $p \in [0, 1]$ to avoid collisions that occur when transitioning from configurations
  - p is a progress variable and is independent of time (e.g., a small jump in p could be a large or a small jump in time)
- The most frequent cost function is the shortest path
- In the literature, the kinematics constraint $\mathcal{B}(\mathbf{q}(p)) \subset \mathcal{W}_{free}$ is also sometimes written as $\mathbf{q}(p) \in \mathcal{Q}_{free}$, where $\mathcal{Q}_{free}$ is the free C-space

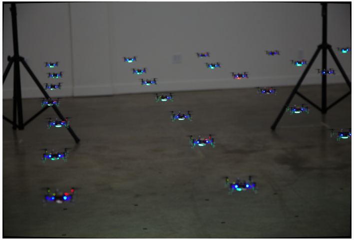
Geometric Motion Planning: Example



Geometric Planning and Robots

Source: New Venturist

Source: Örebro University



Robots are subject to Kinodynamic constraints:

Kinematic contraints Geometric constraints (e.g. joint limits, obstacles)

Dynamic constraints Temporal constraints (e.g. velocity, acceleration)

Both might be coupled (e.g. car turning)

Robot Dynamics

- We often have a model that describes how robots may move
- Often described as differential equation

Control/Action Space

The vector space that defines all possible controls/actions to operate a robot.
Notation: $\mathcal{U}$

Car Control/Action Space

Assume we can control the steering wheel angle ϕ and the speed s of the car. Then we have $\mathbf{u} = (s, \phi) \in \mathcal{U} \subset \mathbb{R}^2$.

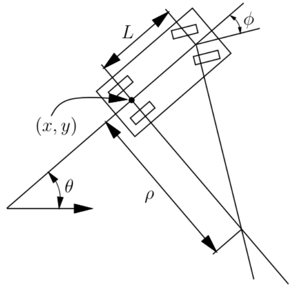
Robot Dynamics

Dynamics

A function that describes the change of the configuration space, given the current configuration and control.

Notation: $\mathbf{f} : \mathcal{Q} \times \mathcal{U} \to \mathcal{Q}$

Car Dynamics



We have $\mathbf{u} = (s, \phi) \in \mathcal{U}$ and $\mathbf{q} = (x, y, \theta) \in \mathcal{Q}$, where s is the speed, ϕ the steering wheel angle, x, y is the position, and θ is the orientation. The dynamics $\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \mathbf{u})$ are:

$$\dot{x} = s\cos\theta \qquad \dot{y} = s\sin\theta \qquad \dot{\theta} = \frac{s}{L}\tan\phi$$

Kinodynamic Motion Planning: Definition

- Input:
  - Description of the environment
  - Description of the robot shape
  - Initial configuration
  - Goal configuration
- Output: "best" collision-free sequence of configurations that obeys robot dynamics from the initial to the goal configuration

Kinodynamic Motion Planning

Given initial and final configurations $\mathbf{q}_{start} \in \mathcal{Q}$ and $\mathbf{q}_{goal} \in \mathcal{Q}$, the free workspace $\mathcal{W}_{free}$, the configuration map $\mathcal{B}(\cdot)$, the robot dynamics $\mathbf{f}$, and a cost function J; we aim to find duration T, a sequence of controls $\mathbf{u} : [0, T) \to \mathcal{U}$, and a sequence of configurations $\mathbf{q} : [0, T] \to \mathcal{Q}$:

$$\underset{T,\mathbf{u}(t),\mathbf{q}(t)}{\text{argmin}} \quad J(T, \mathbf{u}(t), \mathbf{q}(t)) \quad \text{s.t.}$$

$$\mathbf{q}(0) = \mathbf{q}_{start} \quad \mathbf{q}(T) = \mathbf{q}_{goal}$$

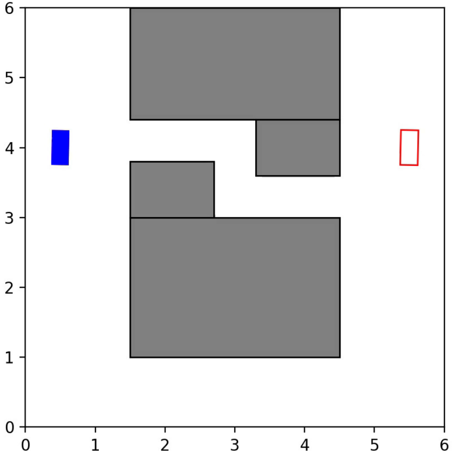
$$\mathcal{B}(\mathbf{q}(t)) \subset \mathcal{W}_{free} \quad \forall t \in [0, T]$$

$$\dot{\mathbf{q}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) \quad \forall t \in [0, T)$$

Kinodynamic Motion Planning: Notes

- Discrete sequences for $\mathbf{q}(t)$ and $\mathbf{u}(t)$ are more practical (but problematic for the definition, see the geometric case)
- Common cost functions are: time or energy $(J(T, \mathbf{u}(t), \mathbf{q}(t)) = \int_t \|\mathbf{u}(t)\|^2 dt)$
- Other frequently used synonyms: Nonholonomic motion planning, Control-space motion planning

Kinodynamic Motion Planning: Example



Geometric vs. Kinodynamic Motion Planning

- Path: Sequence of configurations (no temporal component)
- Motion (or trajectory): Sequence of configuration/time pairs

Path Planning vs. Motion Planning

Sometimes path and motion are used as synonyms. We use path (and path planning) for the geometric version and motions (and motion planning) for the kinodynamic version.

Is Kinodynamic Planning a Superset of Geometric Planning?

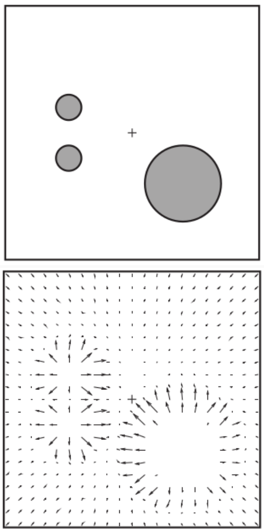
Analysis in the exercise!

Motion Planning vs. Collision Avoidance

- Virtual Potential Fields
  - Attractive "Force": towards goal
  - Repulsive "Force": avoid obstacles (static or dynamic)
  - Follow gradient of resulting force
- Often reactive (only considers the current time)

Planning Is Reasoning Over a Time Horizon

Motion planning considers the current and future states (often total time until the goal is reached). Reactive approaches are strictly speaking not planning.



Conclusion

Motivation

- Motion Planning is a part of decision making
- Important for automation, autonomous driving, health care, games/animation

New Terminology

- Configuration / state space
- Degrees of Freedom
- Configuration map
- Action space
- Geometric vs. Kinodynamic motion planning
- Path vs. Motion Planning

Next Time

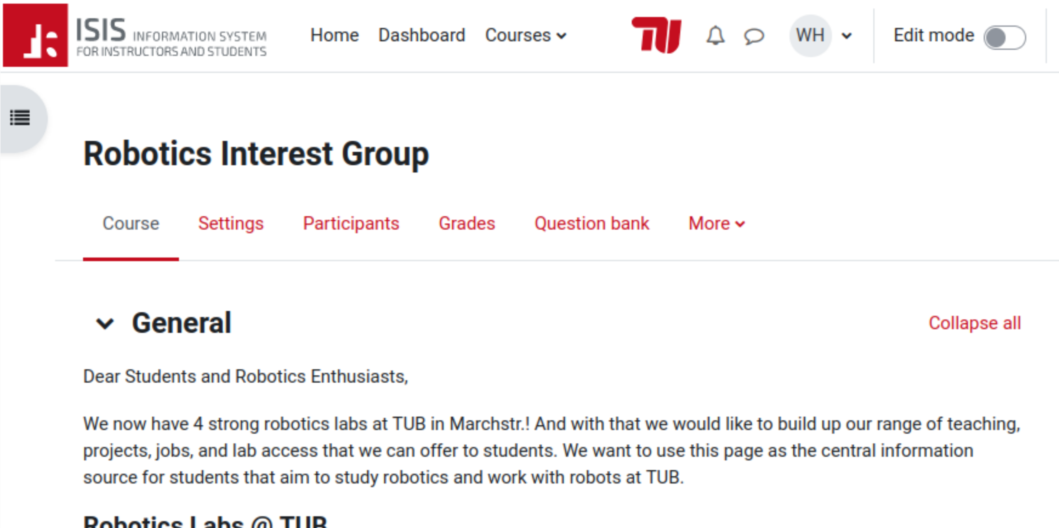
- Foundations for manipulators and operation in 3D
- Efficient Collision Checking

Suggested Reading

1. Marc Toussaint. "Maths for Intelligent Systems". In: (2019). URL: https://www.user.tu-berlin.de/mtoussai/teaching/Lecture-Maths.pdf
2. F. Bullo and S. L. Smith. *Lectures on Robotic Planning and Kinematics*. 2019. URL: http://motion.me.ucsb.edu/book-lrpk/, Section 3.2
3. Kevin M. Lynch and Frank C. Park. *Modern Robotics*. Cambridge University Press, 2017. ISBN: 978-1-107-15630-2. URL: http://hades.mech.northwestern.edu/index.php/Modern_Robotics, Section 10.1
4. Steven M. LaValle. *Planning algorithms*. Cambridge University Press, 2006. ISBN: 978-0-521-86205-9. URL: http://planning.cs.uiuc.edu, Sections 1.2, 1.3

ISIS Robotics Interest Group

https://isis.tu-berlin.de/course/view.php?id=33668



References i

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

[2] F. Bullo and S. L. Smith. *Lectures on Robotic Planning and Kinematics*. 2019. URL: http://motion.me.ucsb.edu/book-lrpk/.

[3] Kevin M. Lynch and Frank C. Park. *Modern Robotics*. Cambridge University Press, 2017. ISBN: 978-1-107-15630-2. URL: http://hades.mech.northwestern.edu/index.php/Modern_Robotics.

[4] Bruno Siciliano and Oussama Khatib, eds. *Springer Handbook of Robotics*. Springer, 2016. ISBN: 978-3-319-32550-7. DOI: 10.1007/978-3-319-32552-1.

References ii

[5] Ioannis Karamouzas, Nick Sohre, Rahul Narain, and Stephen J Guy. "Implicit crowds: Optimization integrator for robust crowd simulation". In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–13.

[6] Ibrahim Al-Bluwi, Thierry Siméon, and Juan Cortés. "Motion planning algorithms for molecular simulations: A survey". In: *Comput. Sci. Rev.* 6.4 (2012), pp. 125–143. DOI: 10.1016/j.cosrev.2012.07.002.

[7] Ankur Dhanik, John S McMurray, and Lydia E Kavraki. "DINC: a new AutoDock-based protocol for docking large ligands". In: *BMC structural biology* 13.1 (2013), pp. 1–14.

[8] Marc Toussaint. "Maths for Intelligent Systems". In: (2019). URL: https://www.user.tu-berlin.de/mtoussai/teaching/Lecture-Maths.pdf.