FACULTY OF MECHANICAL ENGINEERING

institute of automation and computer science

Cybernetics and Robotics Division
NETME Centre

FACULTY OF MECHANICAL ENGINEERING
institute of automation and computer science

Programming for robots and manipulators
Lecture 6

BRNO UNIVERSITY OF TECHNOLOGY

Roman Parak@vutbr.cz

IACS
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

Content

Introduction

Introduction

Motion planning is the problem of finding a robot motion from a start state to a goal state that avoids obstacles in the environment and satisfies other constraints, such as joint limits or torque limits. Motion planning is one of the most active subfields of robotics.

Motion planning in industrial and mobile robotics
Determining where to move without hitting obstacles!

Key issues in robotic planning:

1. Where am I? **Localization**
2. Where have I been? **Mapping**
3. Where am I going? **Planning**
4. How do I get there? **Navigation**

Motion Planning

Joint-Space vs. Operational-Space

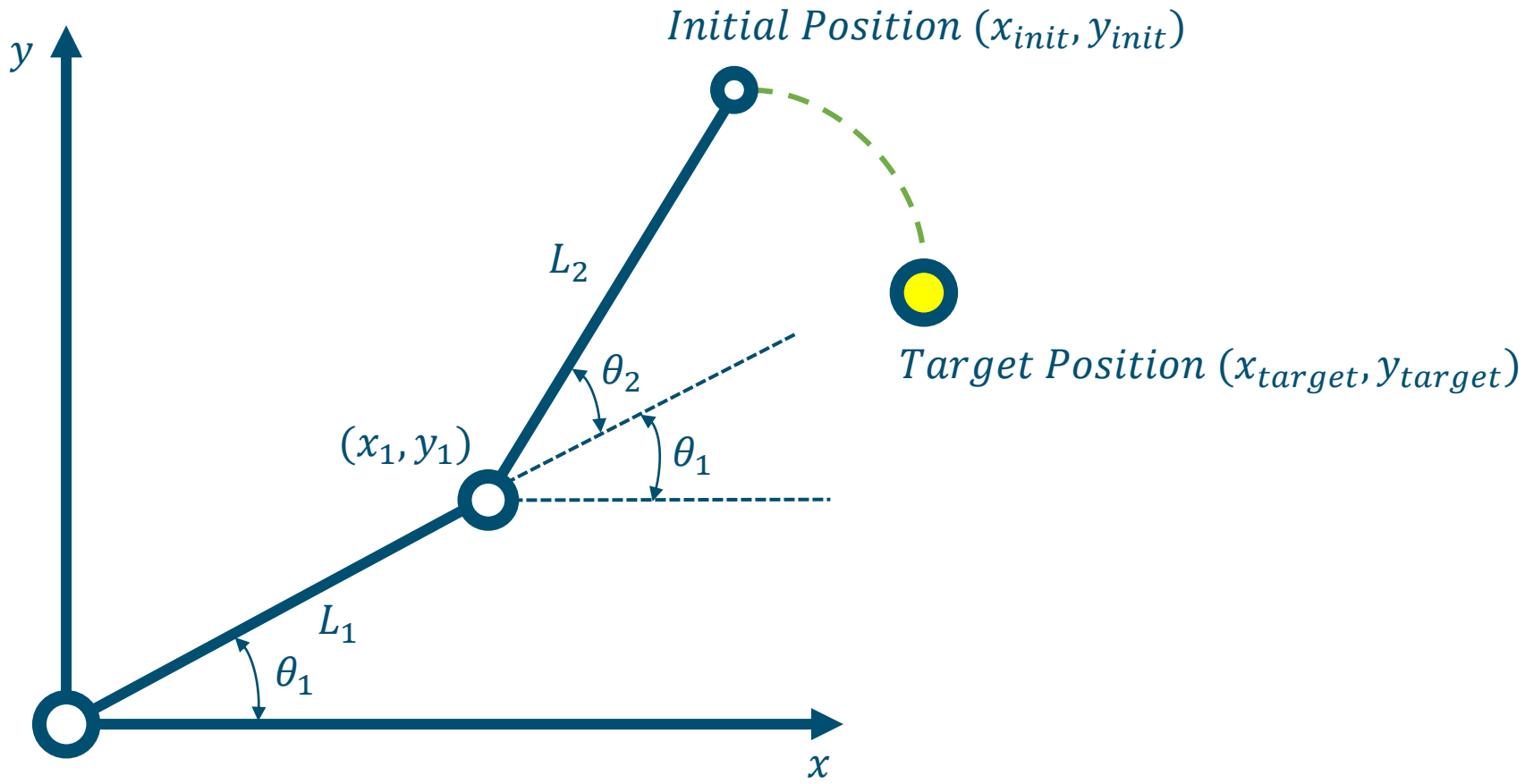
Joint-Space Description:

- The description of the motion to be made by the robot by its joint values.
- The motion between the two points is unpredictable.

Trajectory in the Joint-Space:

- Calculate the inverse kinematics solution from the initial point to the final point.
- Discretize the individual joint trajectories in time.
- Blend a continuous function between these point.

Joint-Space vs. Operational-Space



Joint-Space

Joint-Space vs. Operational-Space

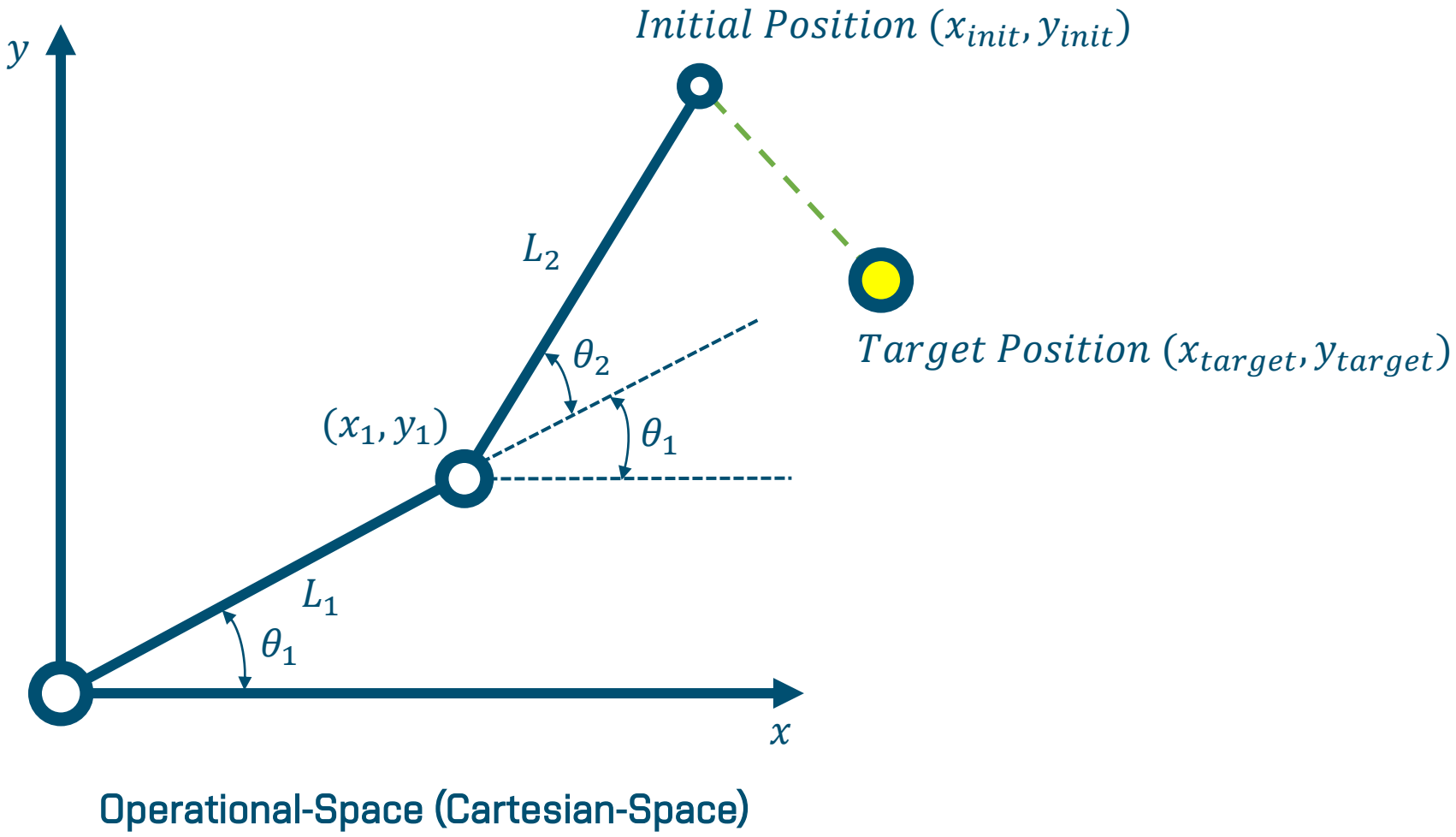
Operational-Space (Cartesian-Space) Description:

- The motion between the two points is known at all times and controllable.
- It is easy to visualize the trajectory, but it is difficult to ensure that singularity does not occur.

Trajectory in the Operational-Space (Cartesian-Space) :

- Calculate the path from the initial point to the final point.
- Discretize points in time and space.
- Blend a continuous time function between these points.
- Solve inverse kinematics at each step.

Joint-Space vs. Operational-Space



Operational-Space (Cartesian-Space)

Path vs Trajectory

The **Path** consists of an ordered number of points in space (either in the joint space or in the operating space) that the robot should follow.

- The path provides a pure geometric description of motion.
- The path is usually planned globally taking into account obstacle avoidance, traversing a complicated maze, etc.

A **Trajectory** is a path with speeds and accelerations at each of its points.

- The design of a trajectory does not need global information, which simplifies the task significantly.
- The trajectory is specified and designed locally. Parts of the path are covered by individual trajectories

Note:
Terms path, trajectory are often exchanged because they are perceived as synonyms informally.

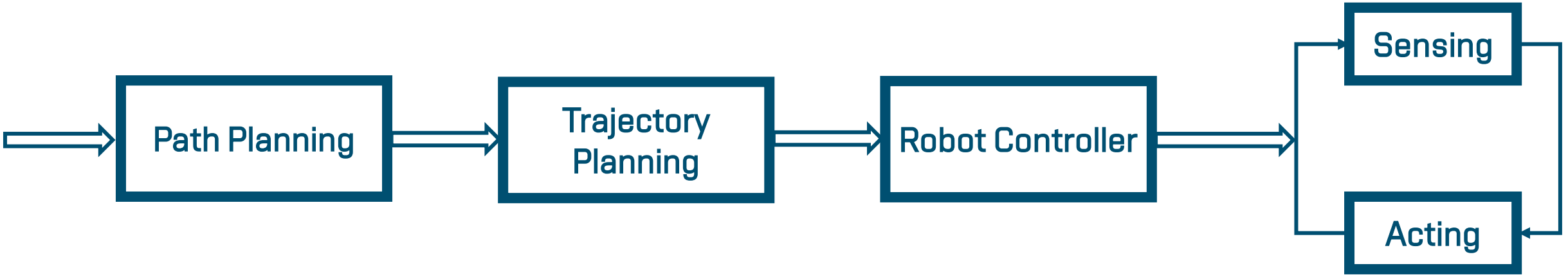
Motion Planning

Path Planning (Global):

- The (geometric) path is a sequence of waypoints defining the trajectory coarsely.
- Issues solved at this level: obstacle avoidance, shortest path.

Trajectory generating (Local)

- The path provided by path planning constitutes the input to the trajectory generator.
- Trajectory generator "approximates" the desired path waypoints by a class of polynomial functions and generates a time-based control sequence moving the manipulator/mobile platform from its initial configuration to the destination.



Problem formulation of Path Planning Task

The path planning task:
Find a collision free path for the robot from one configuration to another configuration.

Path planning is an algorithmically difficult search problem.

- The involved task has an exponential complexity with respect to the degrees of freedom (controllable joints).
- With industrial robots, the path planning is often replaced by a path/trajectory taught in by a human operator.

Problem formulation of Trajectory Generation Task

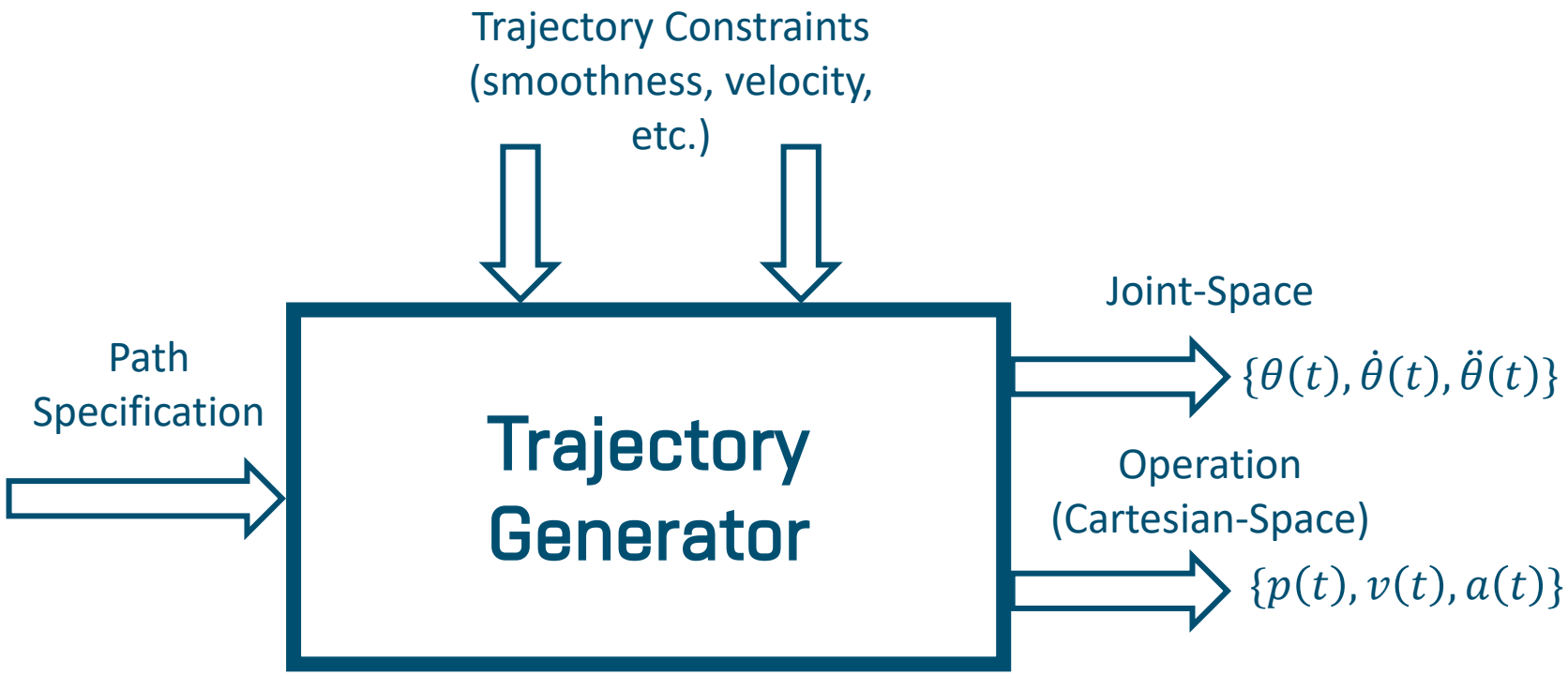
Trajectory generation aims at creating inputs to the motion control system, which ensures that the planned trajectory is smoothly executed.

The planned path is typically represented by way-points, which is the sequence of points (or end-effector poses) along the path.

Trajectory generating - creating a trajectory connecting two or more way-points.

- In the industrial settings, a trajectory is taught-in by a human expert and later played back (by teach-and-playback).
- A more recent approach utilizes several tens of trajectories performed by human experts as the input. They vary statistically. Machine learning techniques are used to create the final trajectory.
- In general, e.g. with mobile robots and more and more with industrial robots, the path points generated by the path planner are smoothly approximated using function approximation methods from mathematics.

Trajectory Generating



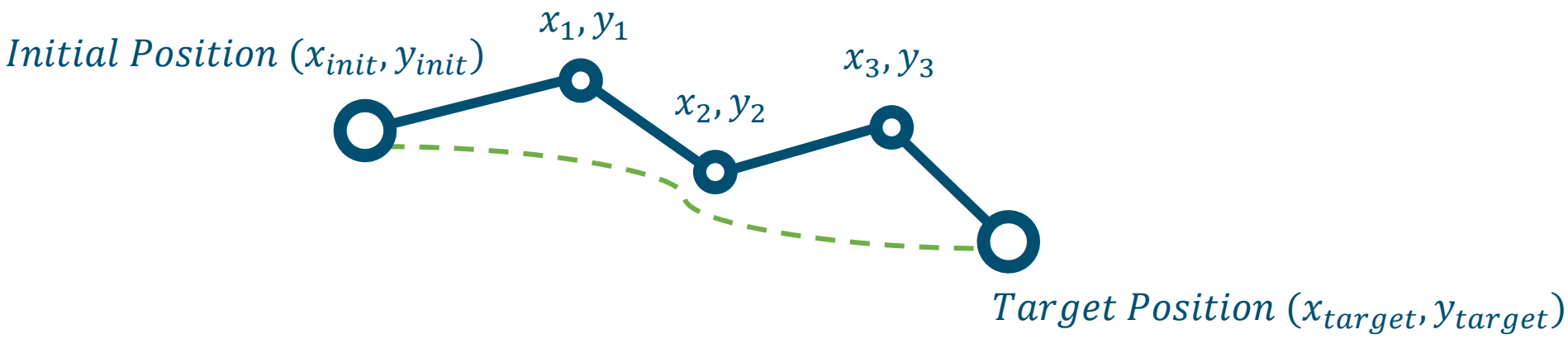
Bézier Curve

Bézier Curve

A Bézier curve is a parametric curve used in computer graphics and related fields. The curve, which is related to the Bernstein polynomial, is named after Pierre Bézier, who used it in the 1960s for designing curves for the bodywork of Renault cars.

Other uses include the design of computer fonts, animation, and motion planning. Bézier curves can be combined to form a Bézier-Spline, or generalized to higher dimensions to form Bézier surfaces.

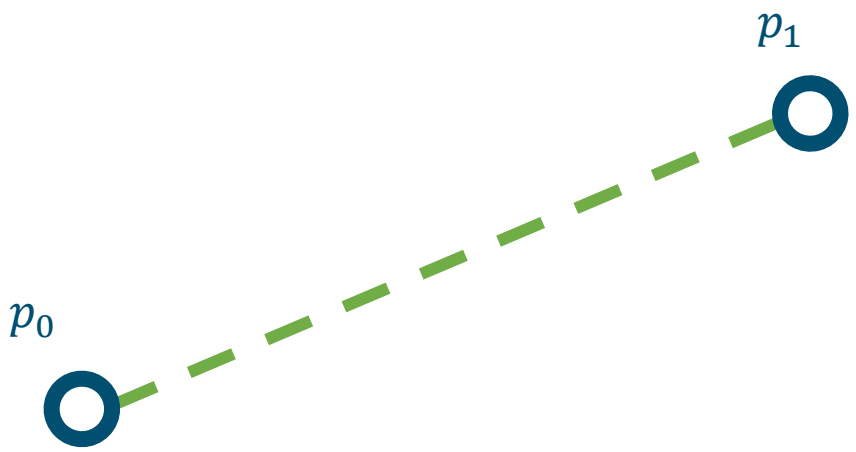
If we wish to store a curve or surface it's impractical to store every point. The solution is to somehow represent entire curves using small amounts of data.



Linear Bezier Curve

Definition.
Given two control points p_0 and p_1 we define the linear Bezier curve to be the curve parametrized by:

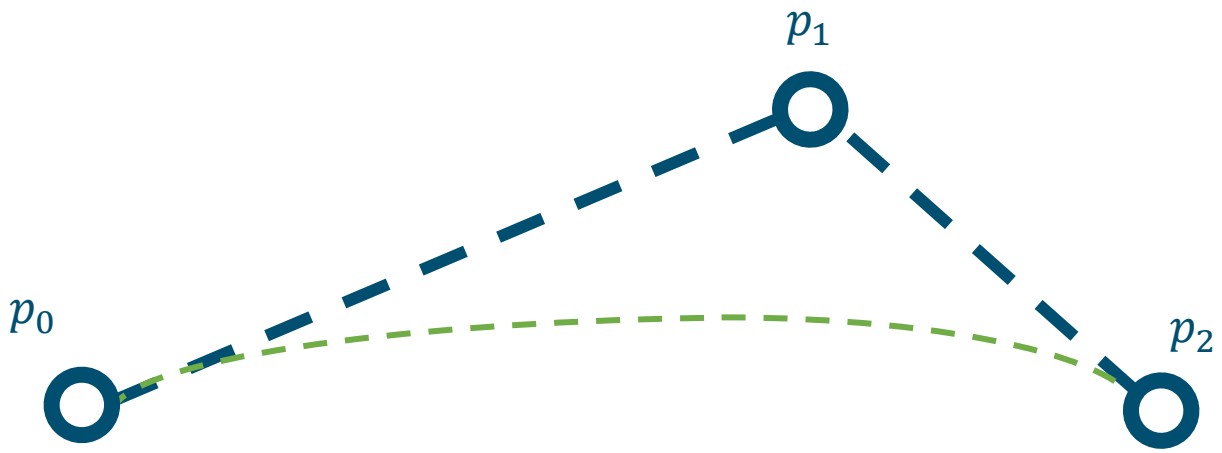
$$p(t) = (1-t)p_0 + tp_1, t \in [0,1]$$



Quadratic Bezier Curves

Definition.
Given three control points p_0, p_1 and p_2 we define the quadratic Bezier curve (degree 2 Bezier curve) to be the curve parametrized by:

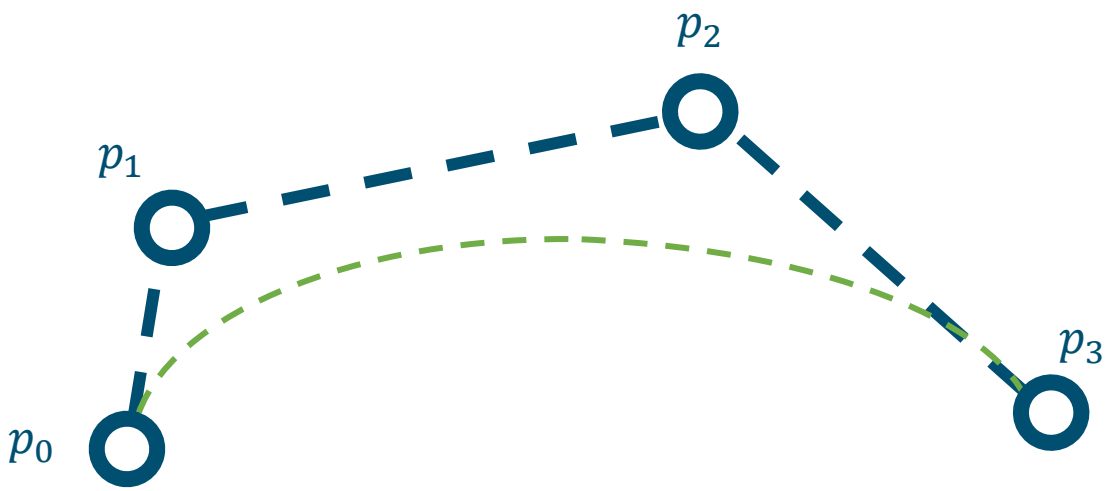
$$p(t) = (1-t)^2 p_0 + 2t(1-t)p_1 + t^2 p_2, t \in [0,1]$$



Cubic Bezier Curve

Definition.
Given four control points p_0, p_1, p_2 and p_3 we define the cubic Bezier curve (degree 3 Bezier curve) to be the curve parametrized by:

$$p(t) = (1-t)^3 p_0 + 3t(1-t)^2 p_1 + 3t^2(1-t)p_2 + t^3 p_3, t \in [0,1]$$



Planning Methods

Motion Planning – Search Approach

Motion planning is the problem of finding a robot motion from a start state to a goal state that avoids obstacles in the environment and satisfies other constraints, such as joint limits or torque limits. Motion planning is one of the most active subfields of robotics.

Search Approach
Searches algorithms use some function that estimates the cost from the current state to the goal, presuming that such a function is efficient (Heuristic approach).

Methods:
- A* Search (one of the most powerful and popular graph search algorithm)
- Dijkstra's algorithm, Breadth-first search, Suboptimal A* Search

The disadvantage of these approaches, however, is their high computational complexity, making them unsuitable for systems having more than a few degrees of freedom.

Motion Planning – Sampling Approach

Motion planning is the problem of finding a robot motion from a start state to a goal state that avoids obstacles in the environment and satisfies other constraints, such as joint limits or torque limits. Motion planning is one of the most active subfields of robotics.

Sampling Approach

Sampling methods, relies on a random or deterministic function to choose a sample from the space. Sampling methods generally give up on the resolution-optimal solutions of a grid search in exchange for the ability to find satisficing solutions quickly in high-dimensional state spaces.

Methods:

- RRT (Rapidly-Exploring Random Tree), RRT-Connect, Bidirectional RRT, etc.
- PRM (Probabilistic Roadmap)

Modern (Innovative) Planning Methods

Introduction

Planning the trajectory of the robotic arm as one of the most basic and challenging research topics in robotics has found considerable interest from research institutes in recent decades. Traditional task and motion planning methods, such as RRT, PRM can solve complex tasks but require full state observability, a lot of time for problem solving and are not adapted to dynamic scene changes.

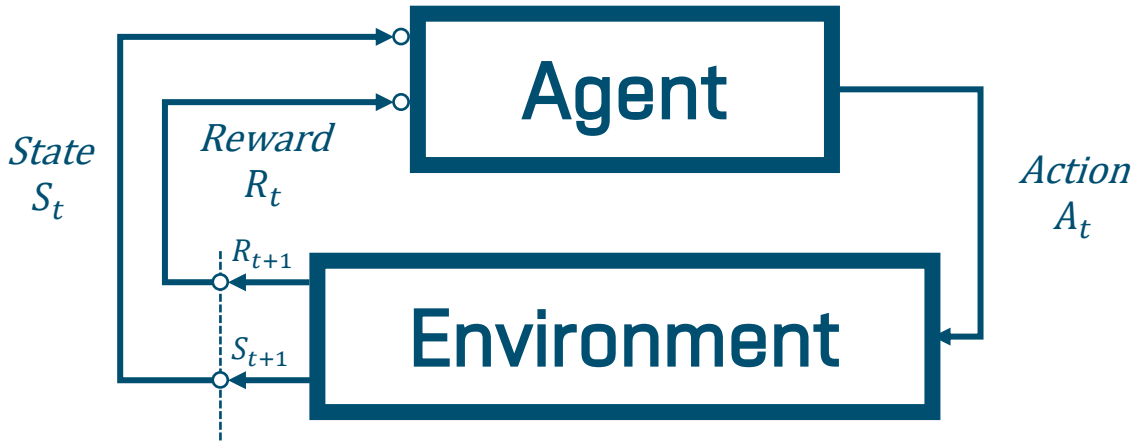
In recent years, there has been an increase in deep neural network (DNN) methods in several areas of science, technology, medicine, and more, along with significant advances in Deep Reinforcement Learning (DRL) techniques.

Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) methods are a promising approach to solving complex tasks in the real world with physical robots. RL/DRL methods are also used in real-world applications, such as improvements in the gaming industry for the Go game}, as well as in robotic applications for manipulation, goal achievement, Human-Robot Collaboration, and more.

Reinforcement Learning/Deep Reinforcement Learning

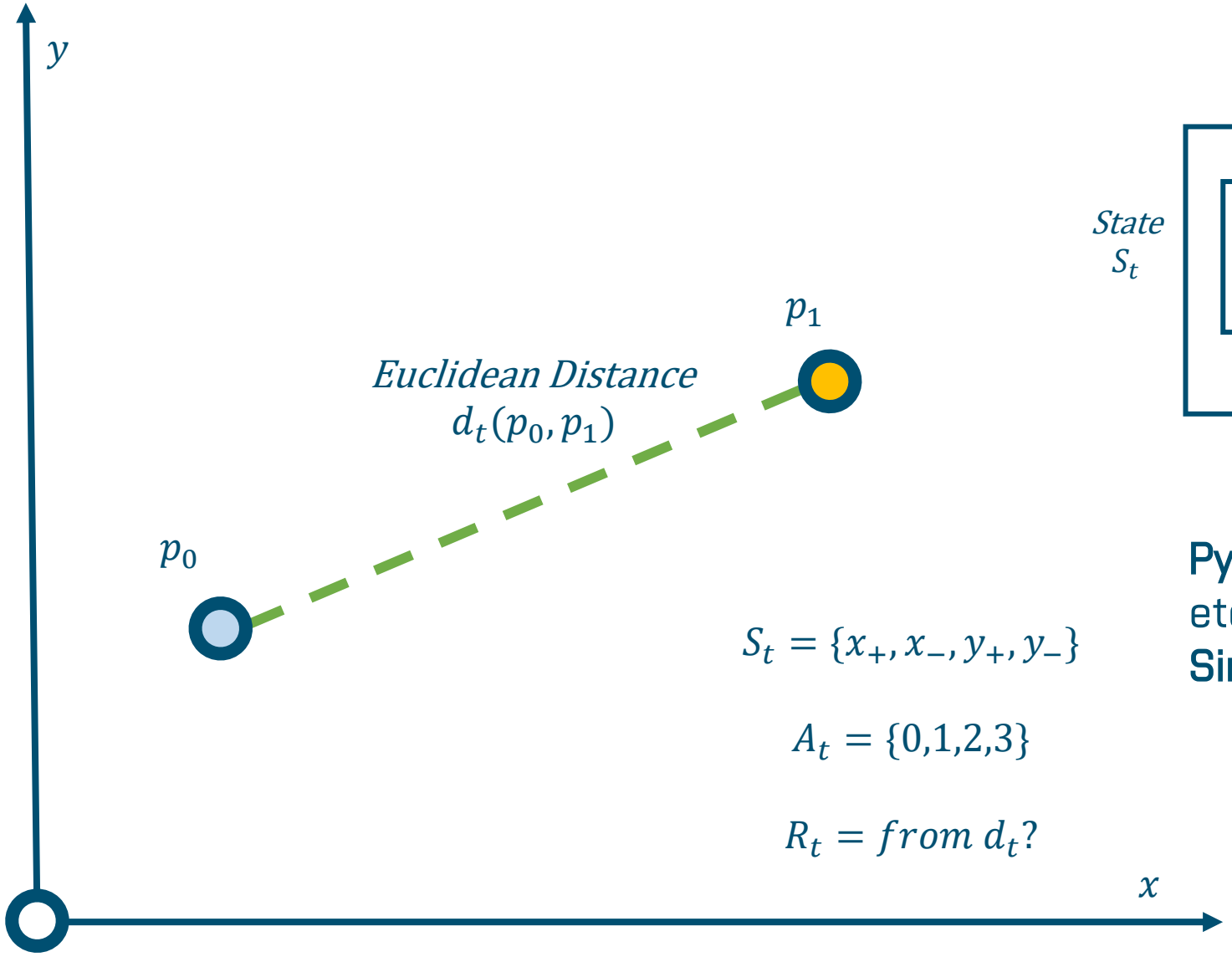
Agent (Reinforcement Learning)
- Q-Learning
- SARSA (State–action–reward–state–action)

Agent (Deep Reinforcement Learning)
- DQN (Deep Q-Network), DDQN
- Deep-SARSA
- DDPG (Deep Deterministic Policy Gradient)
- TD3 (Twin Delayed Deep Deterministi)



DRL overcomes the limitations of simple RL methods by combining parallel computation and embedded deep neural networks (DNN).

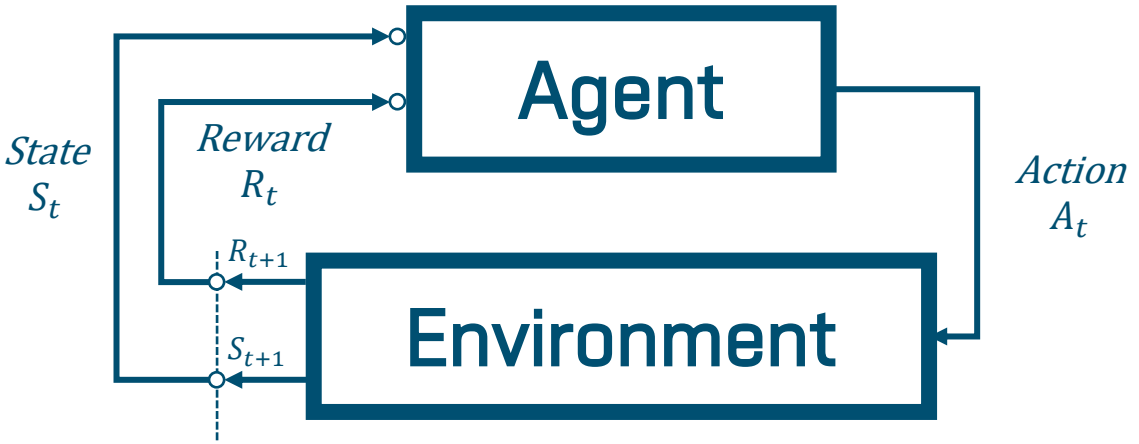
Reinforcement Learning/Deep Reinforcement Learning



$$S_t = \{x_+, x_-, y_+, y_-\}$$

$$A_t = \{0,1,2,3\}$$

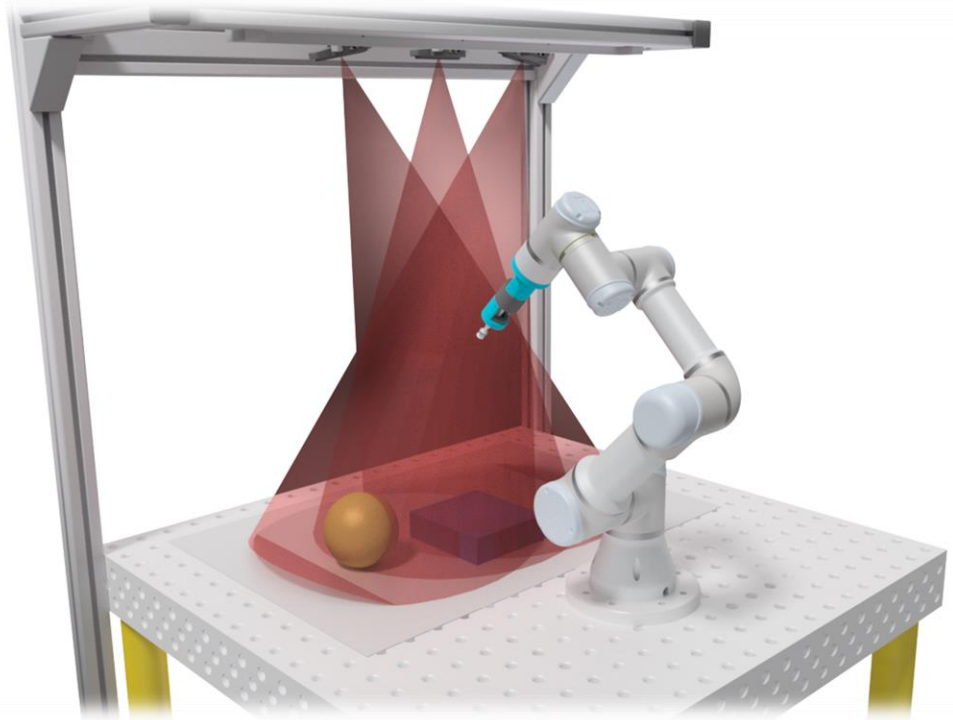
$$R_t = from\ d_t?$$

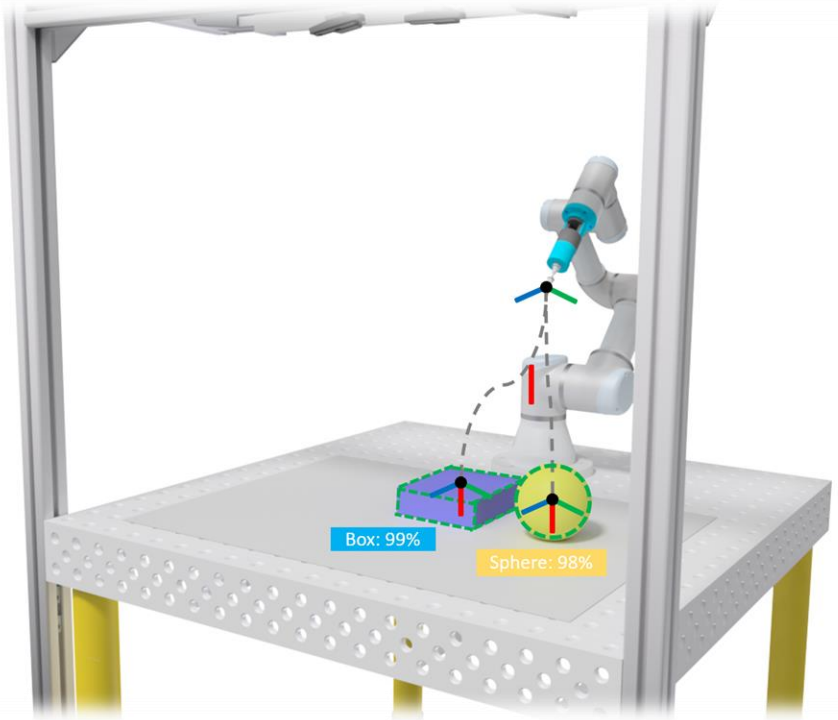


Python Lib. (OpenAI, Tensorflow, PyTorch, etc.)
Simulation Environment (PyBullet, ROS, etc.)

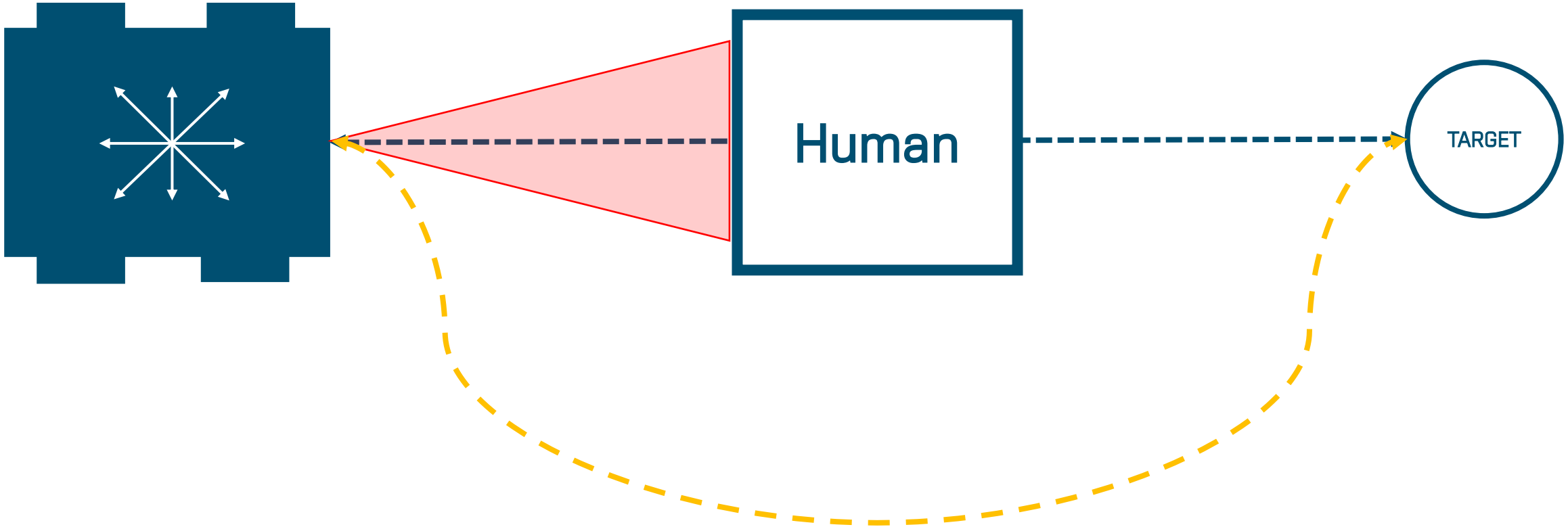
Why do we need knowledge of motion planning?

Use Case: (1)

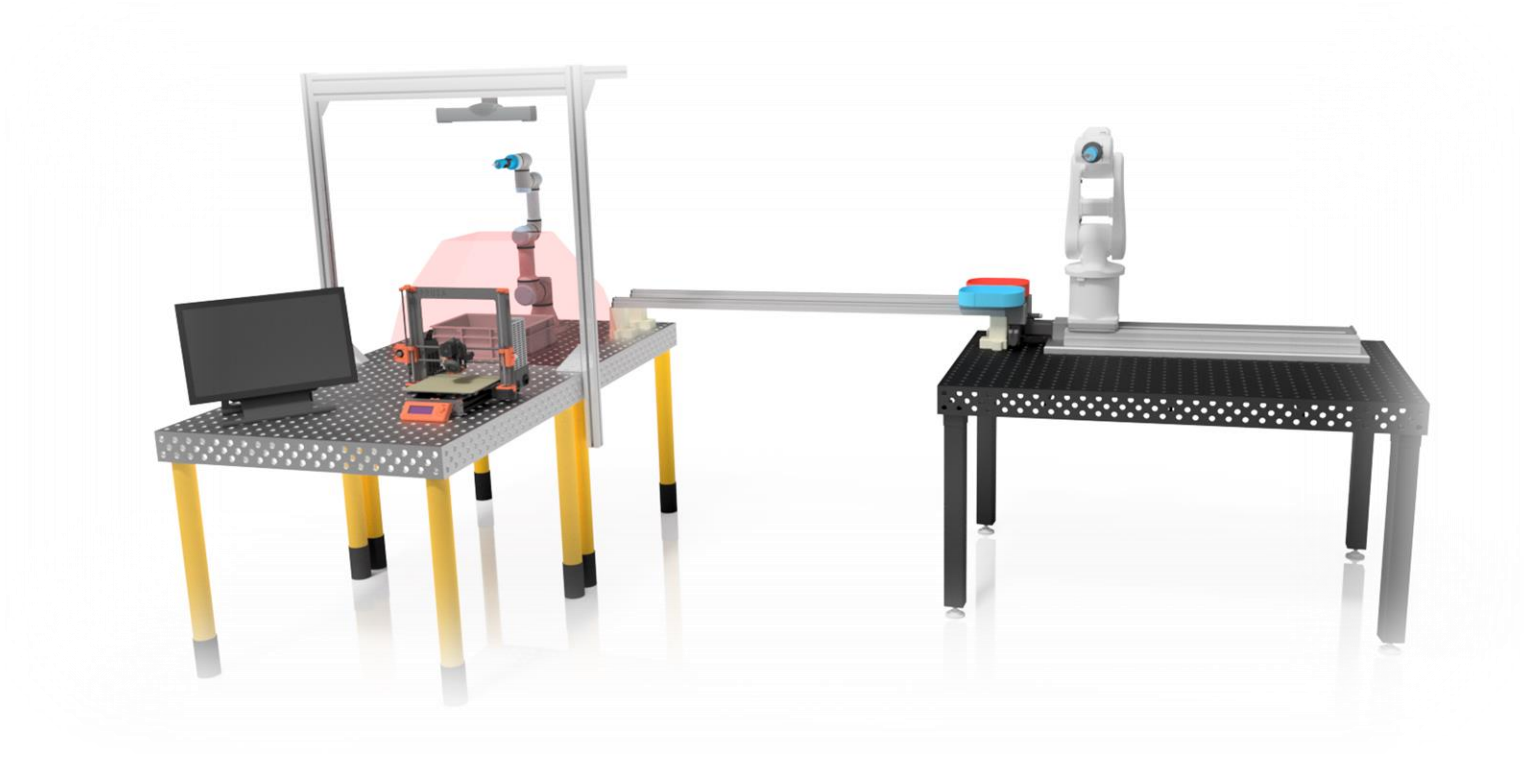




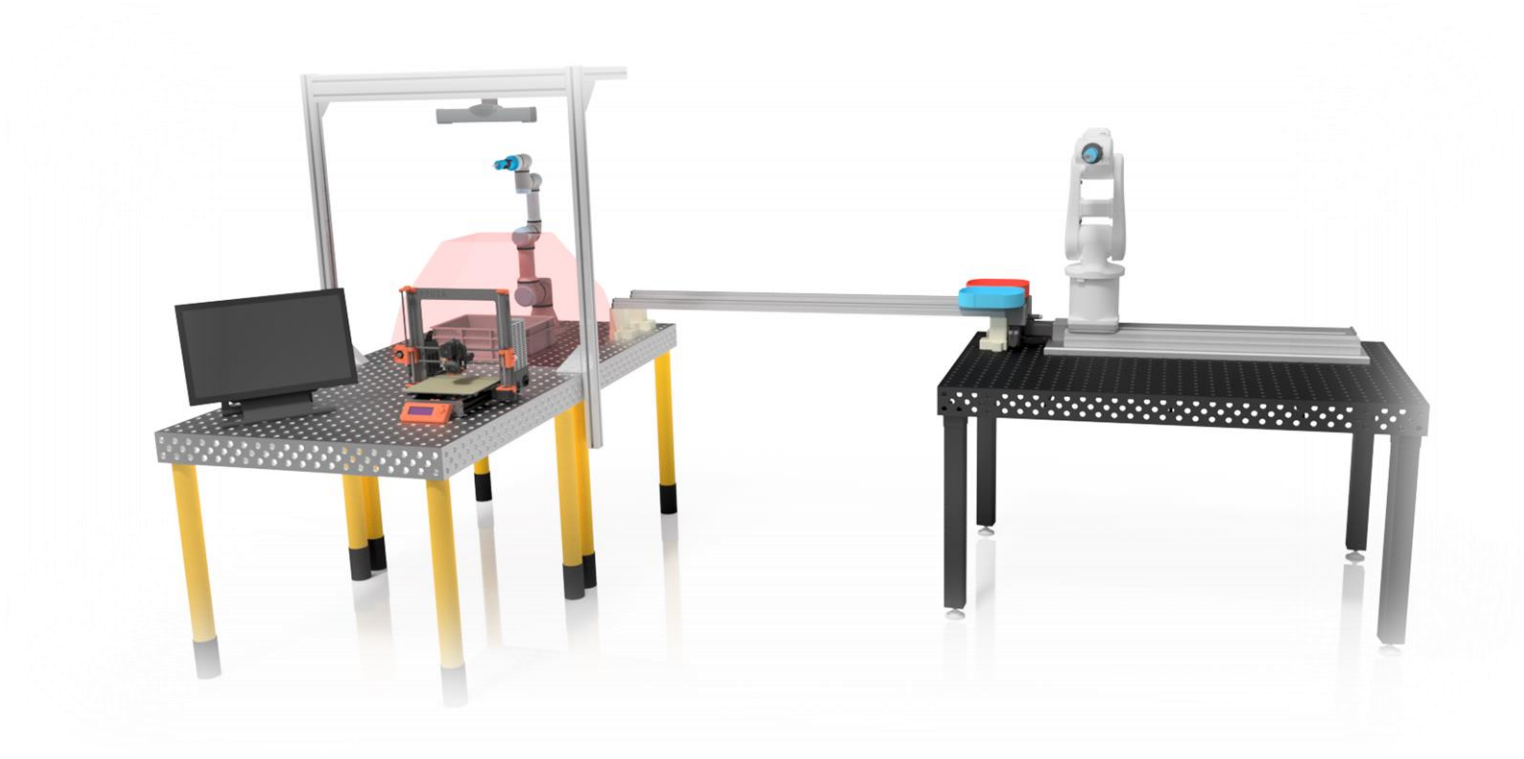
Use Case: (2)



Thank You!



Questions?



FACULTY OF MECHANICAL ENGINEERING

institute of automation and computer science