The idea of sampling-based motion planning

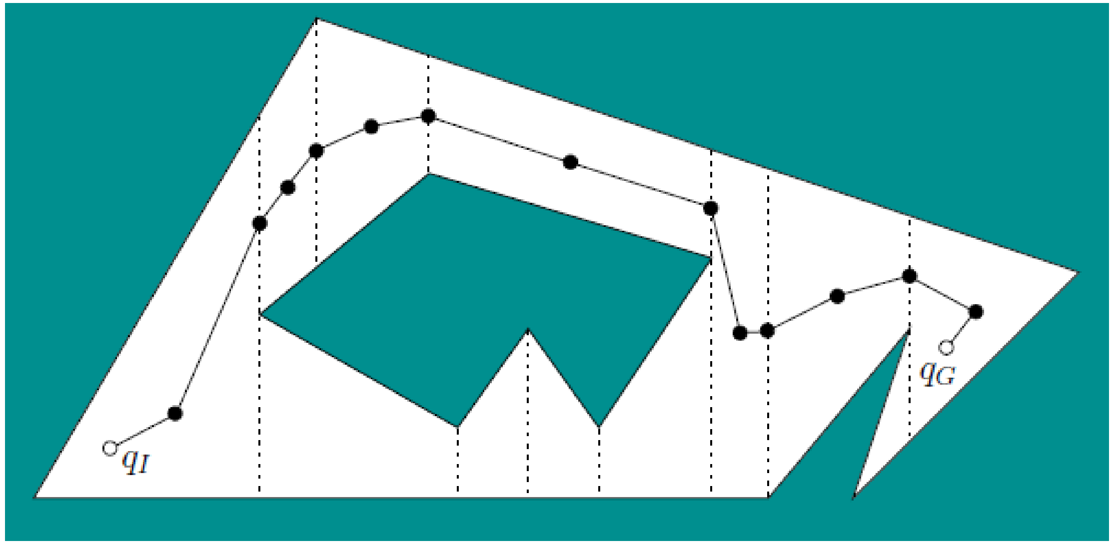
Algorithms and Data Structures 2 – Motion Planning and its applications

University of Applied Sciences Stuttgart

Dr. Daniel Schneider

We know that…

You can solve motion planning problems by computing the configuration space for low dimensions:

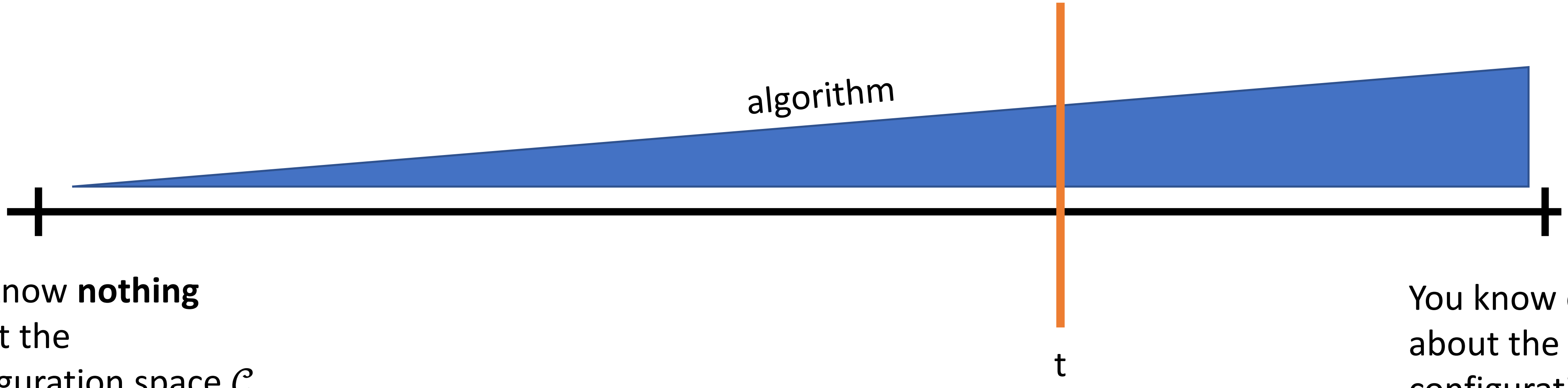


Sources:
Motion Planning: The Essentials – LaValle - http://msl.cs.illinois.edu/~lavalle/papers/Lav11b.pdf

What about higher dimensions?

On a scale

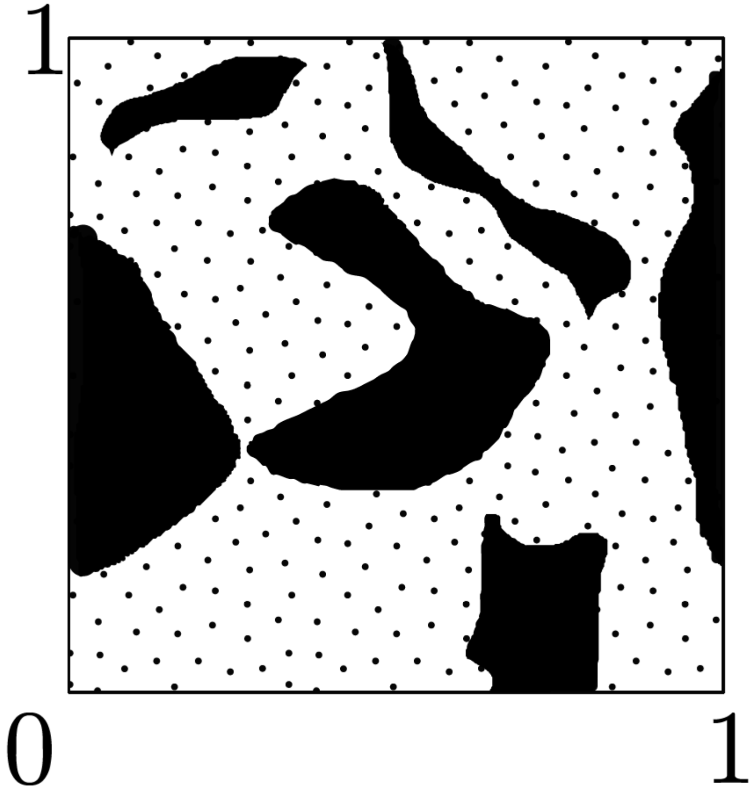
Sufficient to solve
the motion planning problem.



You know **nothing**
about the
configuration space $\mathcal{C}$

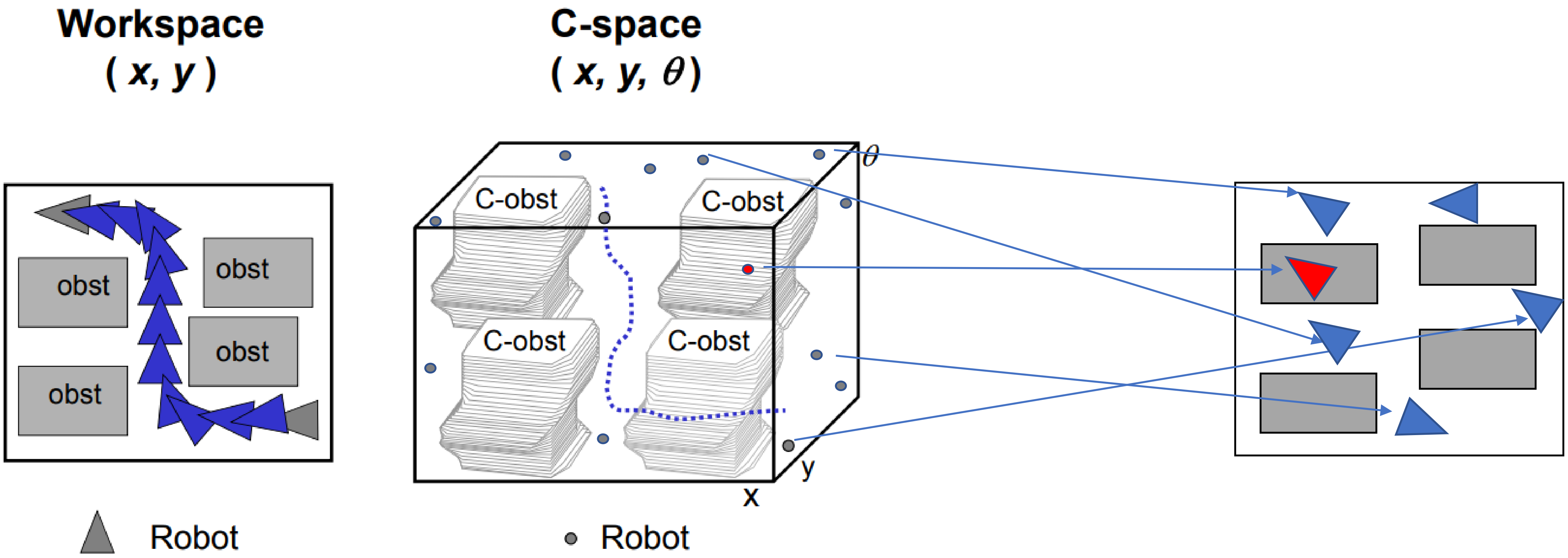
You know **everything**
about the
configuration space $\mathcal{C}$

How?



1. Use a **random** function to sample the configuration space.
2. Check if these samples are free of collision.
3. If so, store each of this sampled configurations in a data structure.
4. Apply algorithms for this set of points to find a patch from one to another configuration.

Example



Sources:
Configuration Space Configuration Space for Motion Planning– Prof. Seth Teller –
http://courses.csail.mit.edu/6.141/spring2010/pub/lectures/Lec10-ConfigurationSpace.pdf

Why does this work?

- We do not need to explicitly compute the configuration space $\mathcal{C}$.
- We approximate the free space $\mathcal{C}_{free}$ by random samples and hope that this approximation is sufficient to solve this problem.
- First we will only address the uniform sampling $U(0,1)$ to sample the configuration space. But there are multiple other approach in research for the random function. → This will be covered later in the lecture.
- As we use random function that covers the whole configuration space therefore we will cover the whole free space at some point in time. →

Why does this work?

- If you find algorithms that are:

Definition 2.5 (Probabilistic Completeness) *An algorithm ALG with x iterations is probabilistically complete if, for any feasible -Motion Planning Problem defined by* $MPP = (C_{free}, x_{init}, X_{goal})$,

$$lim_{x\to\infty} P(ALG \text{ returns a solution to } MPP) = 1 \tag{2.2}$$

You will at some finite time find a solution to the problem (needs proof):

Note:

- Not all algorithms in motion planning are probabilistic complete.
- Even if they are, the “finite” time doers not have to be practical.

What are challenges/requirements with this approach.

- To cover the whole configuration space you have to compute lots of samples. Especially for higher dimensions. →
- Each configuration must be check for validity. This means we have to call our validity function (most of the time the collision detection) in the workspace for each sample. →
- Our collision detection has to be very fast for the algorithm to be fast as well.

What are challenges/requirements with this approach.

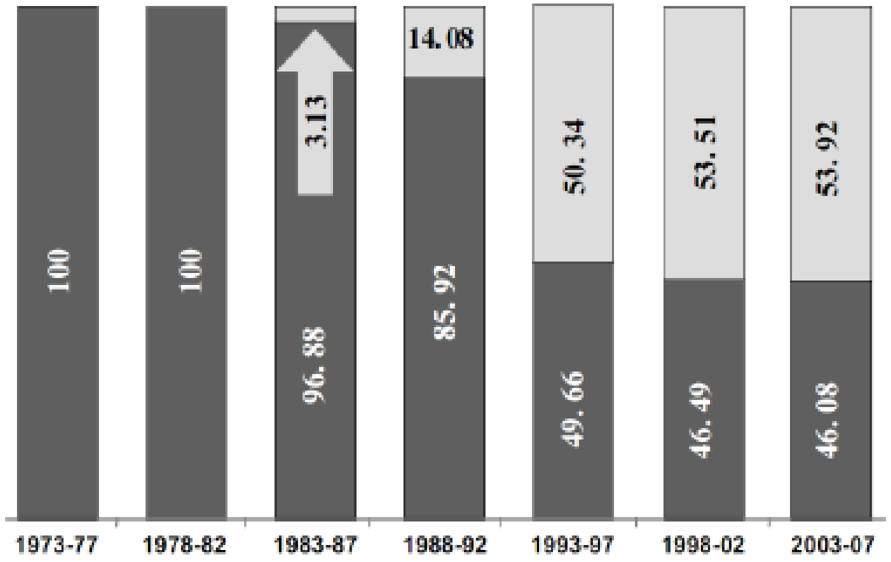
- The runtime of the algorithms is dominated by the collision detection:

Dataset	#4	#5	#6
FCL/PQP: Pre-Processing	0.03s	0.03s	0.03s
FCL/PQP: Other MP Tasks	0.50s	0.11s	0.04s
FCL/PQP: Collision Detection	118.84s	30.91s	37.43s
Sum:	**119.37s**	**31.05s**	**37.5s**

If the other MP tasks are well implemented → Up to 99%

- Collision detection in 3d with large scenes is quite challenging.
- If you have to apply constraints in the workspace → even more

In the past



Sources:
Classic and heuristic approaches in robot motion planning a chronological review
– Ellips Masehian and Davoud Sedighizadeh -
https://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=62832C3081A3BE7A59323DD111A4AAD9?doi=10.1.1.193.2015

→ Nowadays most approaches focus on sampling-based motion planning