

# Principles of Robot Autonomy I

Motion planning II: sampling-based methods

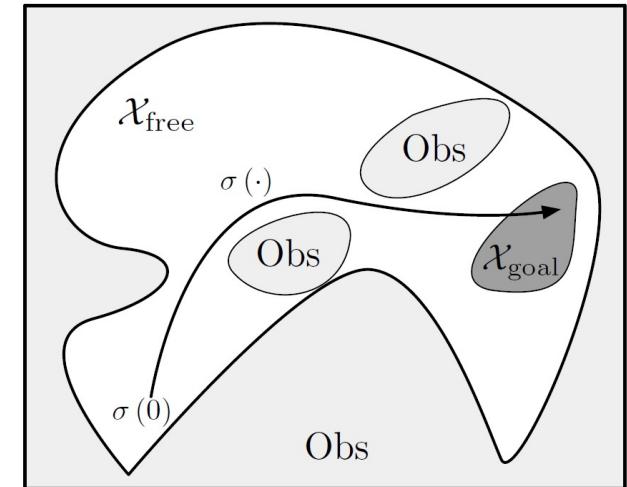


**Stanford**  
University



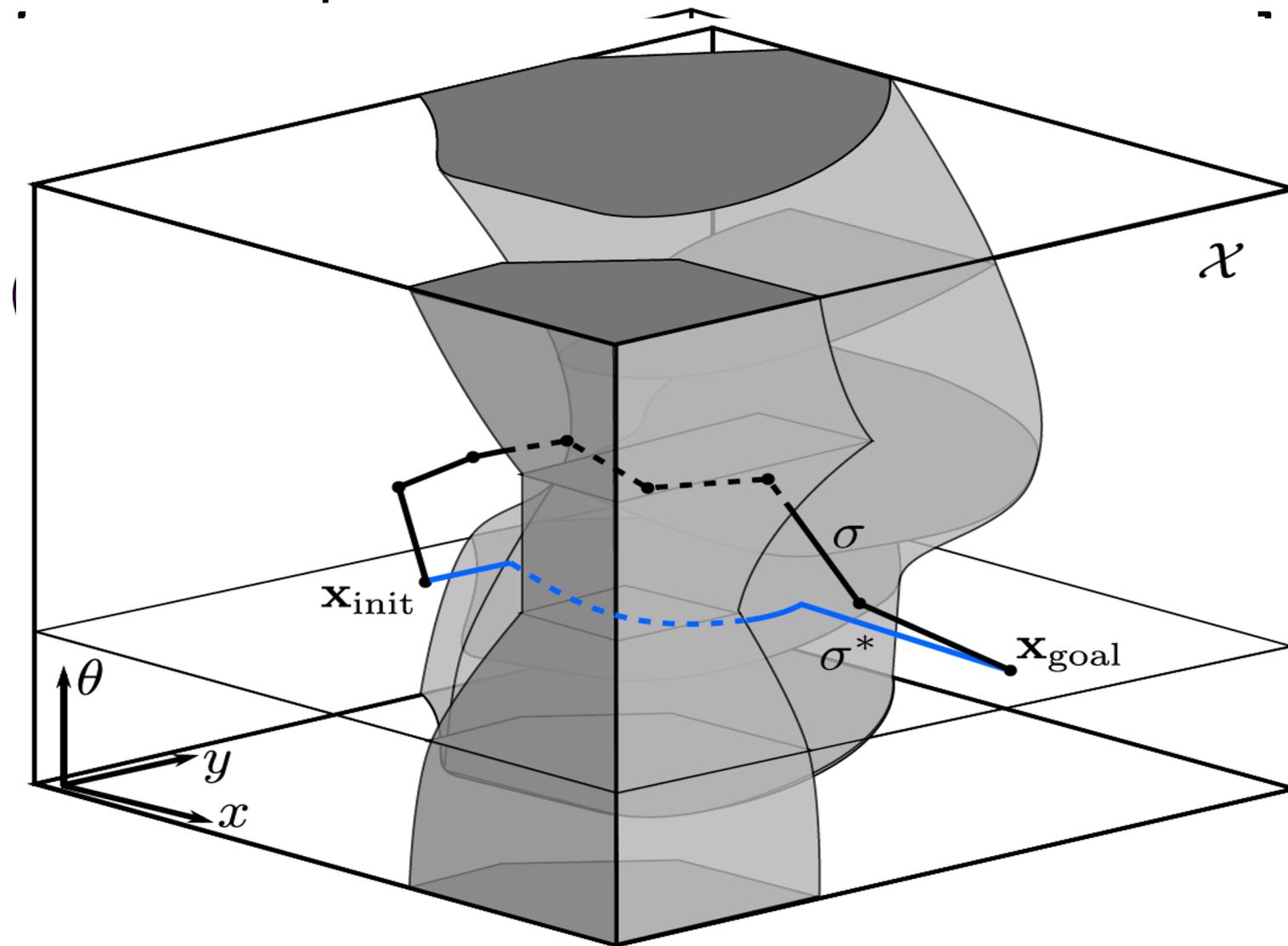
# Motion planning

Compute sequence of actions that drives a robot from an initial condition to a terminal condition while avoiding obstacles, respecting motion constraints, and possibly optimizing a cost function



- Aim
  - Learn about sampling-based motion planning algorithms
- Readings:
  - S. LaValle. Planning Algorithms. Chapter 5.

# Configuration space



# Motion planning algorithms

- **Key point:** motion planning problem described in the real-world, but it really lives in an another space - the configuration (C-)space!
- Two main approaches to *continuous* motion planning:
  - *Combinatorial planning*: constructs structures in the C-space that discretely and completely capture all information needed to perform planning
  - *Sampling-based planning*: uses collision detection algorithms to probe and incrementally search the C-space for a solution, rather than completely characterizing all of the  $C_{free}$  structure

# Sampling-based motion planning

Limitations of combinatorial approaches stimulated the development of sampling-based approaches

- Abandon the idea of explicitly characterizing  $C_{free}$  and  $C_{obs}$
- Instead, capture the structure of  $C$  by **random sampling**
- Use a black-box component (collision checker) to determine which random configurations lie in  $C_{free}$
- Use such a probing scheme to build a roadmap and then plan a path

Reference: LaValle, S. M. Motion planning. 2011.

# Sampling-based motion planning

Pros:

- Conceptually simple
- Relatively-easy to implement
- **Flexible**: one algorithm applies to a variety of robots and problems
- **Beyond the geometric case**: can cope with complex differential constraints, uncertainty, etc.

(Mild) cons:

- Unclear how many samples should be generated to retrieve a solution
- Can not determine whether a solution does not exist

# Outline

- The geometric case
- The kinodynamic case
- Alternative sampling strategies (de-randomized; biased)

# Outline

- The geometric case
- The kinodynamic case
- Alternative sampling strategies (de-randomized; biased)

# Review of sampling-based methods

Traditionally, two major approaches:

- Probabilistic Roadmap (PRM): graph-based
  - **Multi-query planner**, i.e., designed to solve multiple path queries on the same scenario
  - Original version: [Kavraki et al., '96]
  - “Lazy” version: [Bohlin & Kavraki, '00]
  - Dynamic version: [Jaillet & T. Simeon, '04]
  - Asymptotically optimal version: [Karaman & Frazzoli, '11]
- Rapidly-exploring Random Trees (RRT): tree-based
  - **Single-query** planner
  - Original version: [LaValle & Kuner, '01]
  - RDT: [LaValle, '06]
  - SRT: [Plaku et al., '05]
  - Asymptotically optimal version [Karaman & Frazzoli, '11]

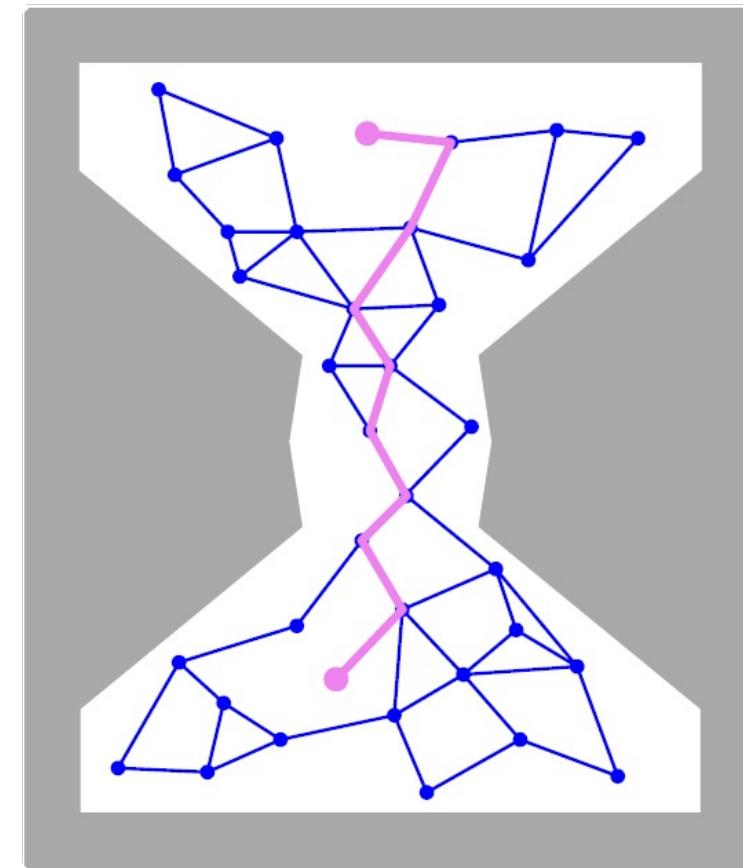
# Probabilistic roadmaps (PRM)

A **multi-query** planner, which generates a roadmap (graph)  $G$ , embedded in the free space

Preprocessing step:

1. Sample a collection of  $n$  configurations  $X_n$ ; discard configurations leading to collisions
2. Draw an edge between each pair of samples  $x, x' \in X_n$  such that  $\|x - x'\| \leq r$  and straight-line path between  $x$  and  $x'$  is collision free

Given a query  $s, t \in C_{free}$ , connect them to  $G$  and find a path on the roadmap



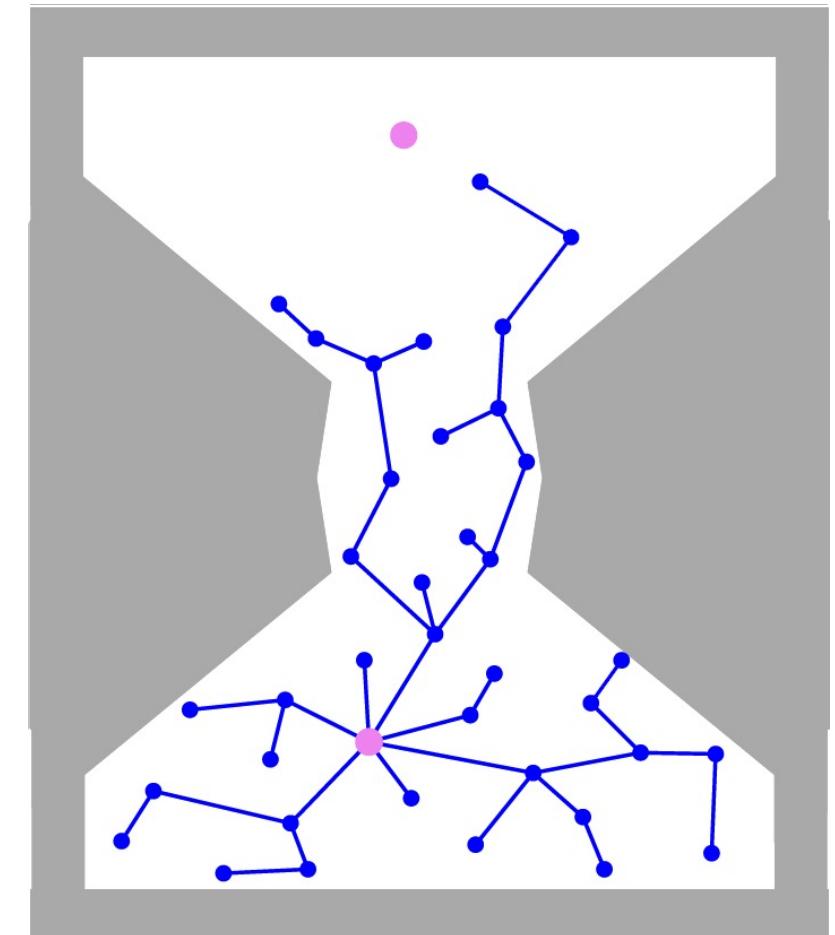
# Rapidly-exploring random trees (RRT)

A **single-query** planner, which grows a tree  $T$ , rooted at the start configuration  $s$ , embedded in  $C_{free}$

Algorithm works in  $n$  iterations:

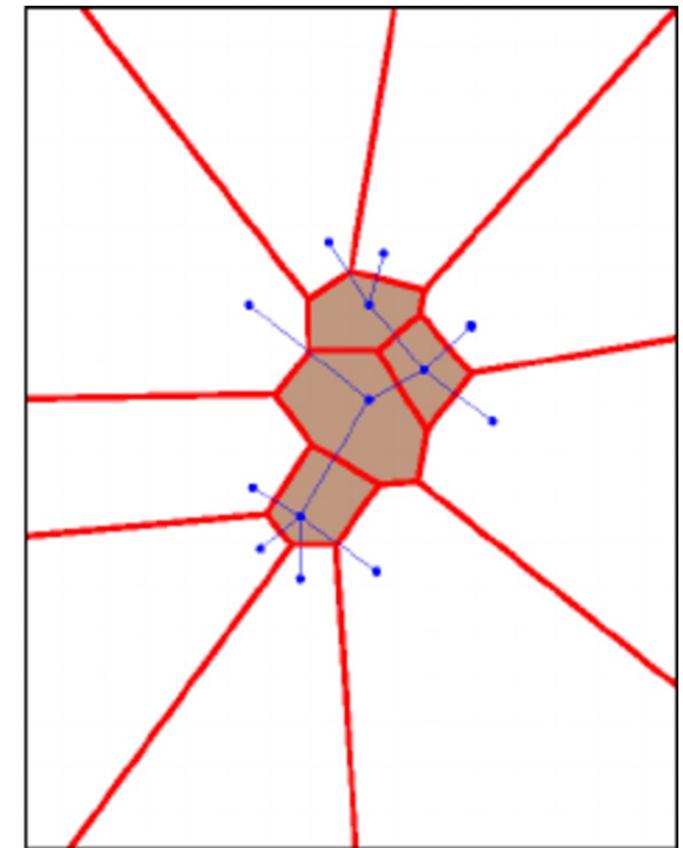
1. Sample configuration  $x_{rand}$
2. Find nearest vertex  $x_{near}$  in  $T$  to  $x_{rand}$
3. Generate configuration  $x_{new}$  in direction of  $x_{rand}$  from  $x_{near}$ , such that  $\overline{x_{near}x_{new}} \subset C_{free}$
4. Update tree:  $T = T \cup \{x_{new}, (x_{near}, x_{new})\}$

Every once in a while, set  $x_{rand}$  to be the target vertex  $t$ ; terminate when  $x_{new} = T$



# Rapidly-exploring random trees (RRT)

- RRT is known to work quite well in practice
- Its performance can be attributed to its **Voronoi bias**:
  - Consider a Voronoi diagram with respect to the vertices of the tree
  - For each vertex, its Voronoi cell consists of all points that are closer to that vertex than to any other
  - Vertices on the frontier of the tree have larger Voronoi cells – hence sampling in those regions is more likely



# Theoretical guarantees: probabilistic completeness

Question: how large should the number of samples  $n$  be? We can say something about the **asymptotic behavior**:

**Kavraki et al. 96:** PRM, with  $r = \text{const}$ , will eventually (as  $n \rightarrow \infty$ ) find a solution if one exists

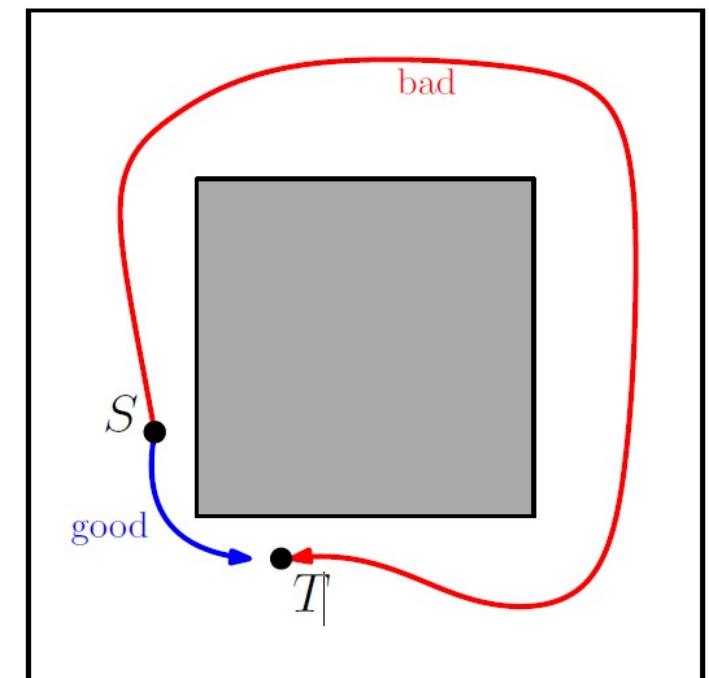
**LaValle, 98; Kleinbort et al., 18:** RRT will eventually (as  $n \rightarrow \infty$ ) find a solution if one exists

\* Unless stated otherwise, the configuration space is assumed to be the  $d$ -dimensional Euclidean unit hypercube  $[0,1]^d$ , with  $2 \leq d \leq \infty$

# Theoretical guarantees: quality

Question: what can be said about the **quality** of the returned solution for PRM and RRT, in terms of length, energy, etc.?

Nechushtan et al. (2011) and Karaman and Frazzoli (2011) proved that RRT can produce arbitrarily-bad paths with non-negligible probability: for example, RRT would prefer to take the long (red) way



# Theoretical guarantees: quality

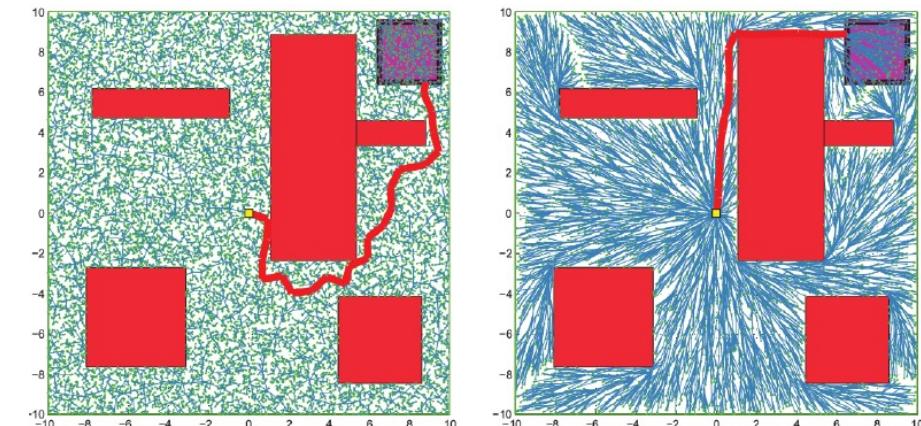
Karaman and Frazzoli in 2011 provided the first rigorous study of optimality in sampling-based planners:

**Theorem:** The **cost** of the solution returned by PRM converges, as  $n \rightarrow \infty$ , to the optimum, when  $r_n = \gamma \left( \frac{\log n}{n} \right)^{\frac{1}{d}}$ , where  $\gamma$  only depends on  $d$

- KF11 also introduced an asymptotically optimal variant of RRT called RRT\* (right)
- Result was later updated to [Solovey et al. 2019]:

$$r_n = \gamma \left( \frac{\log n}{n} \right)^{\frac{1}{d+1}}$$

- Now back to  $1/d$  [preprint, Lukyanenko et al. 2021]

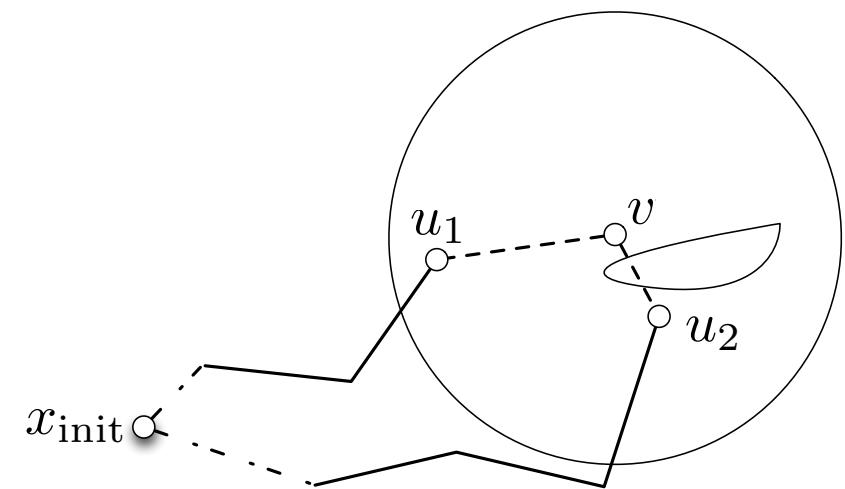


# Observations

- PRM-like motion planning algorithms
  - For a give number of nodes  $n$ , they find “good” paths
  - ...however, require many costly collision checks
- RRT-like motion planning algorithms
  - Finds a feasible path quickly
  - ...however the quality of that path is, in general, poor
  - “traps” itself by disallowing new better paths to emerge - RRT\* performs local label correction as samples are added to help remedy this

# Fast Marching Tree Algorithm (FMT\*)

- Key idea: run dynamic programming on sampled nodes, skipping any step in which the attempted connection causes a collision
  - lazy DP operator:
$$c(v) = \min_{u: \|u-v\| < r_n} \text{Cost}(u, v) + c(u)$$
- Laziness introduces “suboptimal” connections, but such connections are vanishingly rare and FMT\* is asymptotically optimal
- Ratio of # of collision-checks for FMT\* versus PRM\* goes to zero!



Reference: Janson et al. Fast Marching Tree: A Fast Marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions. 2015

# Sampling-based planning: summary

- Sampling-based planners transform the difficult global problem into a large set of **local** and **easy** problems
- A key ingredient is **collision detection**, which is conceptually easy, as it can be solved in the workspace (2D or 3D)
- **Local planning** (edge validation) is typically performed by dense sampling of path and collision detection
- Another key ingredient is nearest-neighbor search: given a query point find its nearest neighbor(s) within a set of points -- also well studied theoretically and practically

# Outline

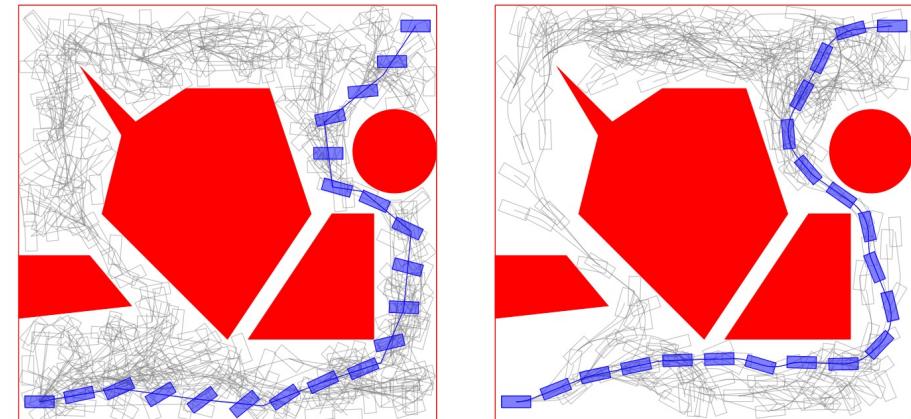
- The geometric case
- The kinodynamic case
- Alternative sampling strategies (de-randomized; biased)

# Kinodynamic planning

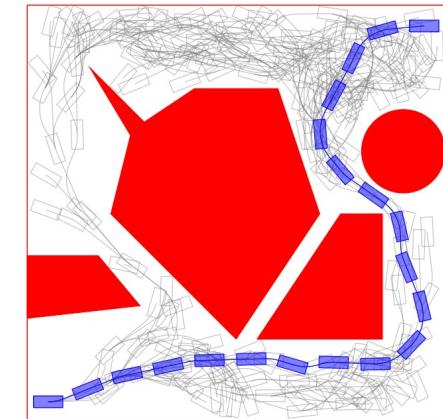
Kinodynamic motion planning problem:  
in addition to obstacle avoidance, paths  
are subject to differential constraints

- The robot operates in the state space  $X$
- To move the robot applies control  $u \in U$
- Motion needs to satisfy the system's constraints:  
$$\dot{x} = f(x, u) \text{ for } x \in X, u \in U$$

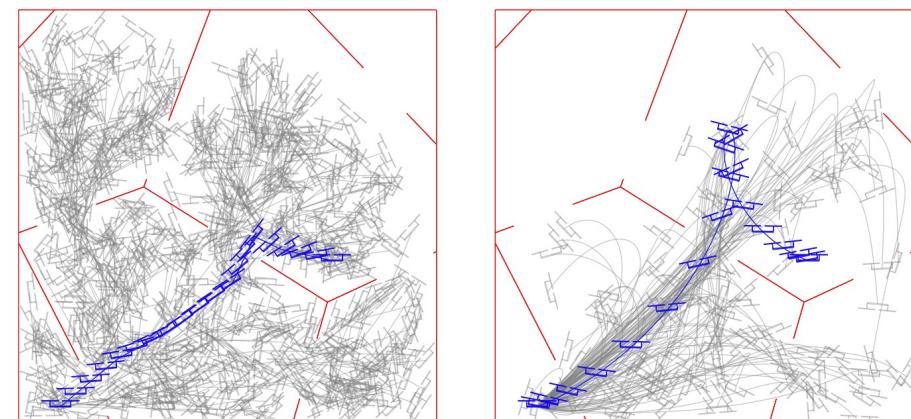
Reference: Schmerling and Pavone. Kinodynamic Planning.  
2019



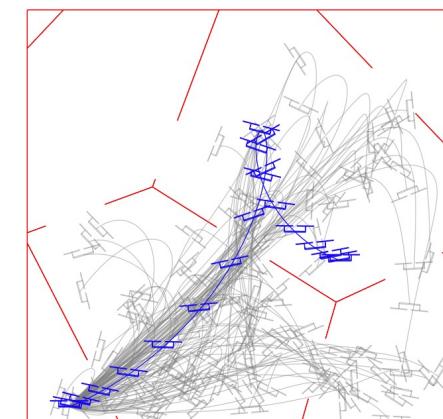
(a) Geometric Planning



(b) Planning with Dubins Car Dynamics



(c) Geometric Planning



(d) Planning with Simplified Quadrotor Dynamics

# Forward-propagation-based algorithms

RRT can be extended to kinodynamic case in a relatively easy way:

1. Draw a random state and find its nearest neighbor  $x_{near}$
2. Sample a random control  $u \in U$  and random duration  $t$
3. **Forward propagate** the control  $u$  for  $t$  time from  $x_{near}$

---

```
1:  $\mathcal{T}.\text{init}(x_{\text{init}})$ 
2: for  $i = 1$  to  $k$  do
3:    $x_{\text{rand}} \leftarrow \text{RANDOM\_STATE()}$ 
4:    $x_{\text{near}} \leftarrow \text{NEAREST\_NEIGHBOR}(x_{\text{rand}}, \mathcal{T})$ 
5:    $t \leftarrow \text{SAMPLE\_DURATION}(0, T_{\text{prop}})$ 
6:    $u \leftarrow \text{SAMPLE\_CONTROL\_INPUT}(\mathbb{U})$ 
7:    $x_{\text{new}} \leftarrow \text{PROPAGATE}(x_{\text{near}}, u, t)$ 
8:   if  $\text{COLLISION\_FREE}(x_{\text{near}}, x_{\text{new}})$  then
9:      $\mathcal{T}.\text{add\_vertex}(x_{\text{new}})$ 
10:     $\mathcal{T}.\text{add\_edge}(x_{\text{near}}, x_{\text{new}})$ 
11: return  $\mathcal{T}$ 
```

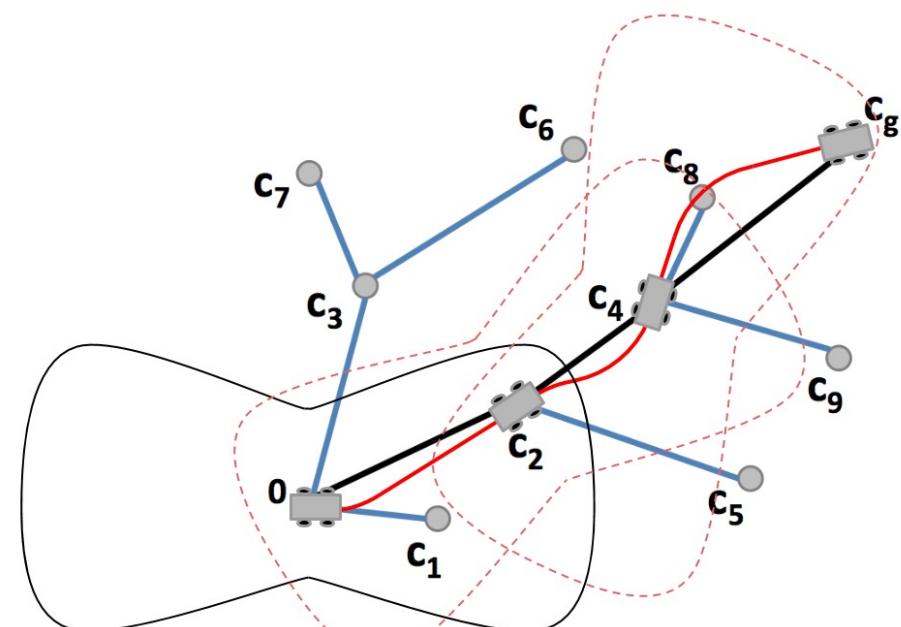
Reference: Kleinbort et al. Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation. 2018.

# Steering-based algorithms

When efficient online *steering* subroutines exist, kinodynamic planning algorithms may take advantage of this domain knowledge

1. Connect samples by using an optimal trajectory (steering problem)
2. Use reachable sets to find nearest neighbors

Reference: E. Schmerling et al. Optimal Sampling-Based Motion Planning under Differential Constraints: the Driftless Case. 2015



# Outline

- The geometric case
- The kinodynamic case
- Alternative sampling strategies (de-randomized; biased)

# Should probabilistic planners be probabilistic?

**Key question:** would theoretical guarantees and practical performance still hold if these algorithms were to be derandomized, i.e., run on deterministic samples?

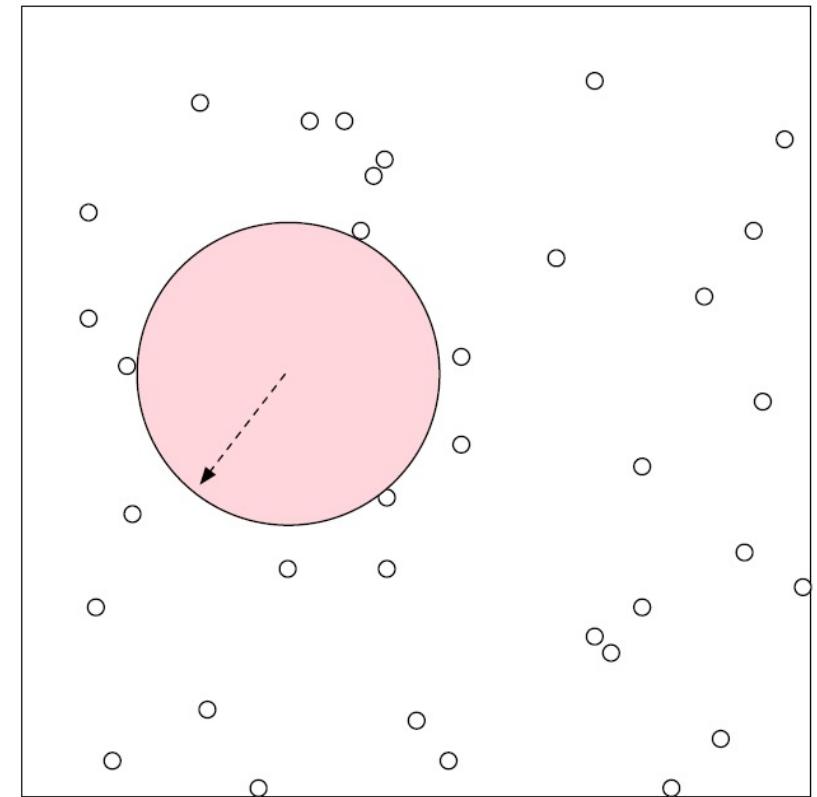
Important question as derandomization would:

- Ease certification process
- Ease use of offline computation
- Potentially simplify a number of operations (e.g., NN search)

# Designing “good” sequences

**$\ell_2$ -dispersion:** For a finite set  $S$  of points contained in  $X \subset \mathbb{R}^d$ , its  $\ell_2$ -dispersion  $D(S)$  is defined as

$$D(S) := \sup_{x \in X} \min_{s \in S} \|s - x\|_2$$



Key facts:

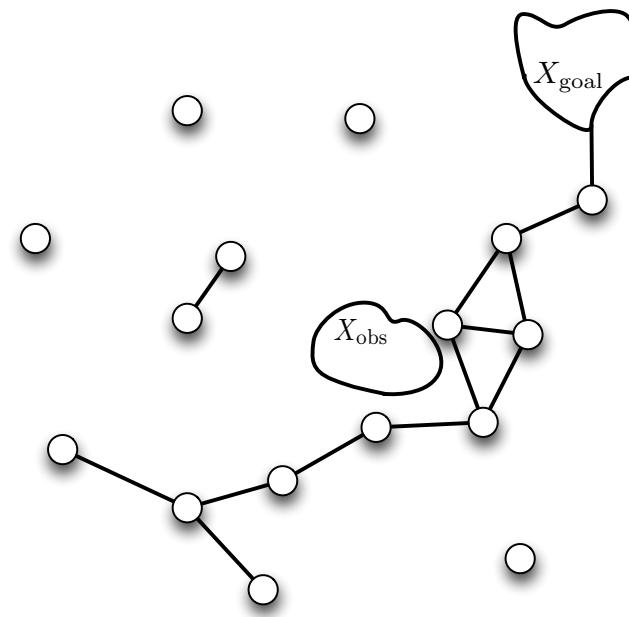
- There exist deterministic sequences with  $D(S)$  of order  $O(n^{-1/d})$ , referred to as **low-dispersion** sequences
- Sequences minimizing  $\ell_2$ -dispersion only known for  $d = 2$

# Optimality of deterministic planning

---

```
1  $V \leftarrow \{x_{\text{init}}\} \cup \text{SampleFree}(n); E \leftarrow \emptyset$ 
2 for all  $v \in V$  do
3    $X_{\text{near}} \leftarrow \text{Near}(V \setminus \{v\}, v, r_n)$ 
4   for  $x \in X_{\text{near}}$  do
5     if CollisionFree( $v, x$ ) then
6        $E \leftarrow E \cup \{(v, x)\} \cup \{(x, v)\}$ 
7     end if
8   end for
9 end for
10 return ShortestPath( $x_{\text{init}}, V, E$ )
```

---



**Optimality:** Let  $c_n$  denote the arc length of the path returned with  $n$  samples. Then if

1. Samples set  $S$  has dispersion  $D(S) \leq \gamma n^{-1/d}$  for some  $\gamma > 0$ ,
2.  $n^{1/d} r_n \rightarrow \infty$ ,

then  $\lim_{n \rightarrow \infty} c_n = c^*$ , where  $c^*$  is the cost of an optimal path

# Deterministic sampling-based motion planning

- Asymptotic optimality can be achieved with **deterministic sequences** and with a **smaller connection radius**
- Deterministic convergence rates: instrumental to the certification of sampling-based planners
- Computational and space complexity: under some assumptions, arbitrarily close to theoretical lower bound
- Deterministic sequences appear to provide superior performance

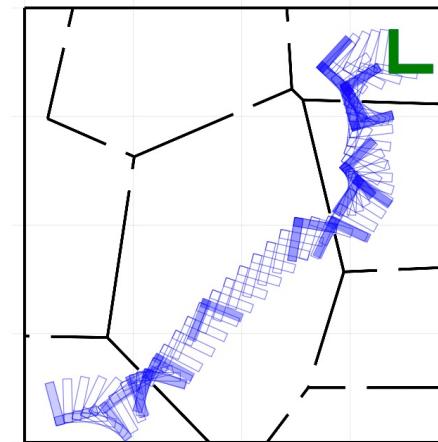
Reference: Janson et al. Deterministic Sampling-Based Motion Planning: Optimality, Complexity, and Performance. 2018

# Biased sampling for SBMP

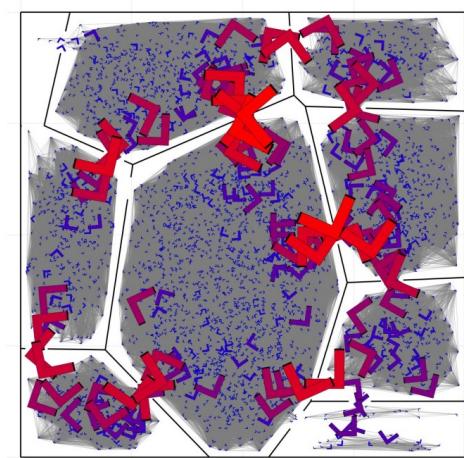
- Potential issue with uniform sampling: narrow corridors in C-space require many samples to identify/traverse
- Key idea: **bias sampling** towards suspected such challenging regions of C-space
- Biased sampling distributions can be hand-constructed and/or adapt online (e.g., Hybrid Sampling PRM), or learned from prior experience solving similar planning problems

## References:

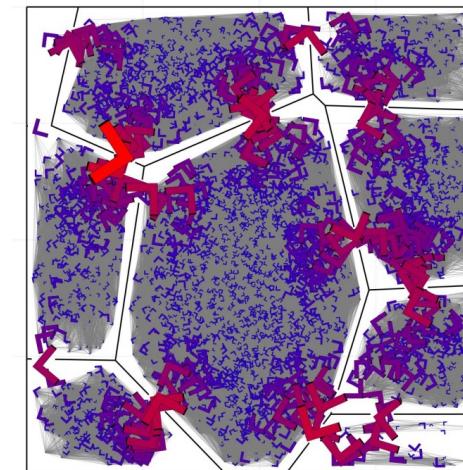
- Hsu et al. Hybrid PRM sampling with a cost-sensitive adaptive strategy. 2005.  
Ichter et al. Learned Critical Probabilistic Roadmaps for Robotic Motion Planning. 2020



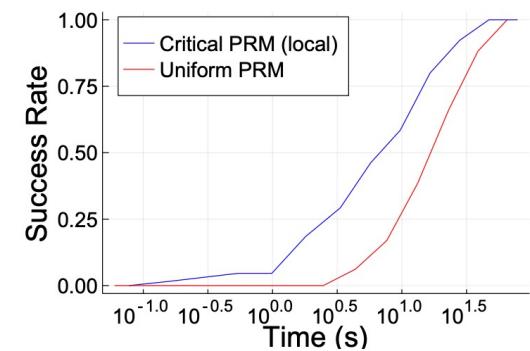
(a) SE(2) Planning



(b) Ground Truth Criticality



(c) Learned Criticality



(d) Time (s) vs. Success

# Next time: robotic sensors and introduction to computer vision

