Introduction to Robotics

Lecture 08

Michael Görner
goerner@informatik.uni-hamburg.de

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

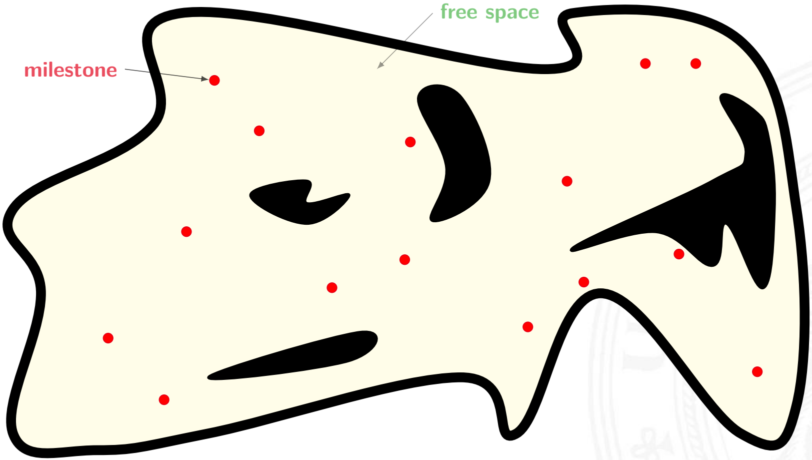
June 04, 2021

Idea

- Planning on graphs of reasonable size is simple
- Operating on grids ignores continuous spaces in $\mathcal{X}_{free}$
- Instead rely on **Probabilistic Sampling** to represent the space

Idea

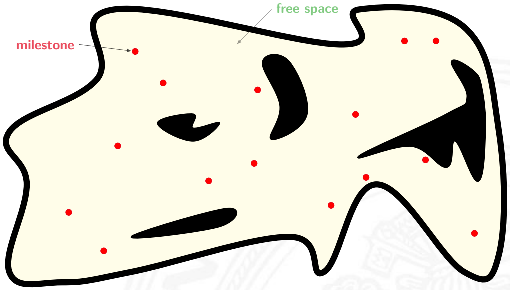
- Planning on graphs of reasonable size is simple
- Operating on grids ignores continuous spaces in $\mathcal{X}_{free}$
- Instead rely on **Probabilistic Sampling** to represent the space



Idea

Key questions:

- How to generate the samples?
- How can the samples be connected to form a planning graph?
- How many samples do you need to describe the space?



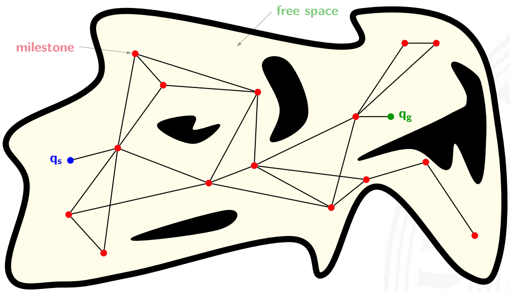
Abstract C-space with sampled valid states

Probabilistic Road Maps

Proposed by Lydia E. Kavraki et.al. 1996 [17]

Two Step algorithm:

1. Construction Phase - Build Roadmap
2. Query Phase - Connect start and goal to graph and solve graph search



Abstract C-space with sampled valid states

Probabilistic Road Maps - Algorithm

Algorithm: sPRM

$V \leftarrow \{x_{\text{init}}\} \cup \{\texttt{SampleFree}_i\}_{i=1,\ldots,n};\ E \leftarrow \emptyset;$

foreach $v \in V$ **do**

  $U \leftarrow \texttt{Near}(G = (V, E), v, r) \setminus \{v\};$

  foreach $u \in U$ **do**

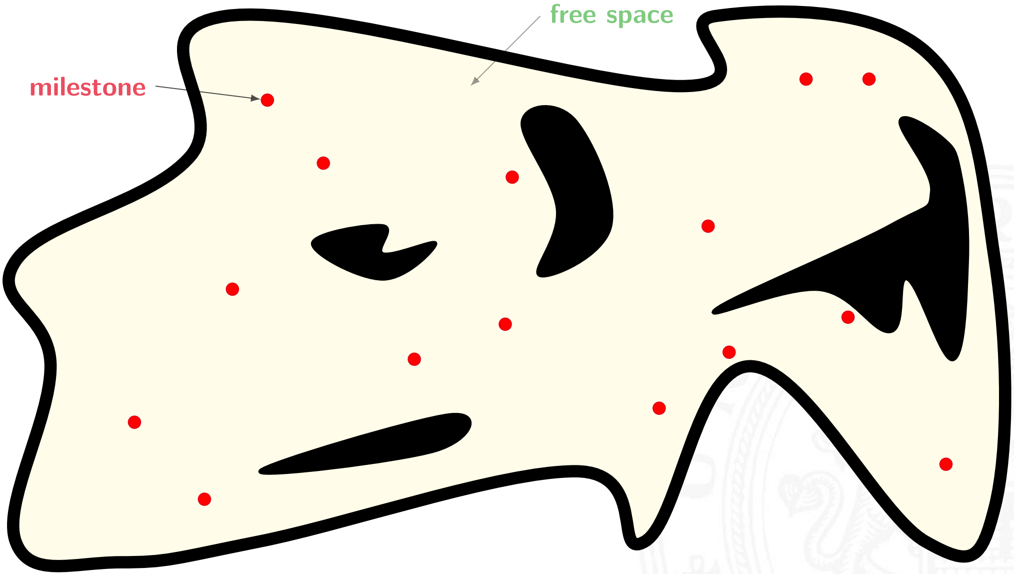
    if $\texttt{CollisionFree}(v, u)$ **then** $E \leftarrow E \cup \{(v, u), (u, v)\}$

return $G = (V, E);$

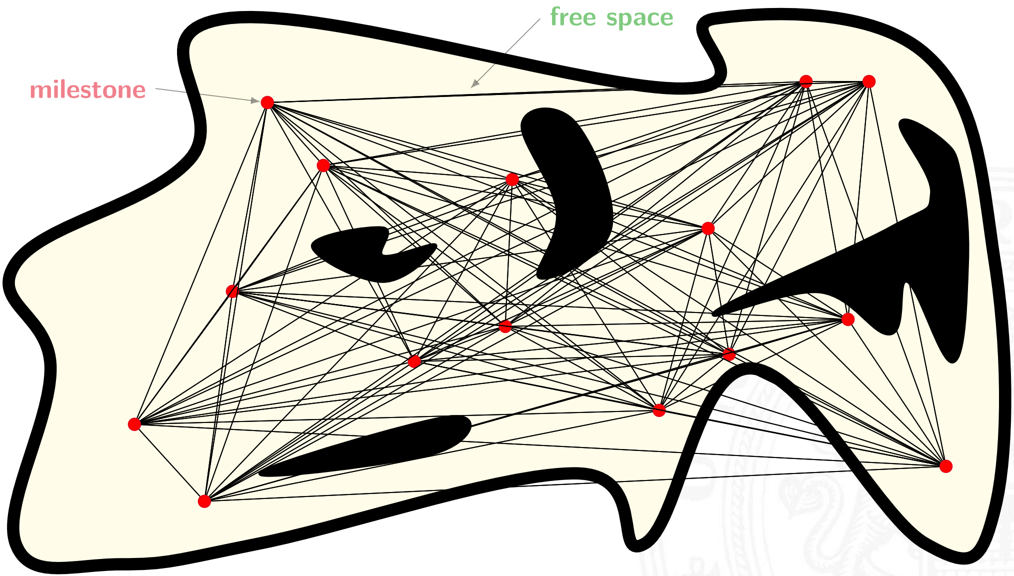
Adapted from [15]

Milestones and Roadmap - Construction

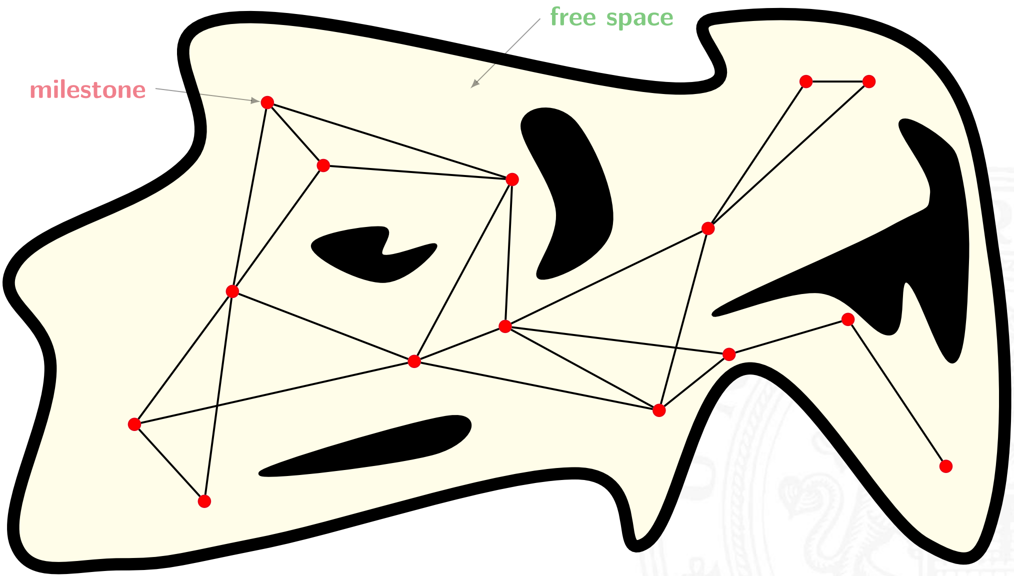
Milestones and Roadmap - Construction



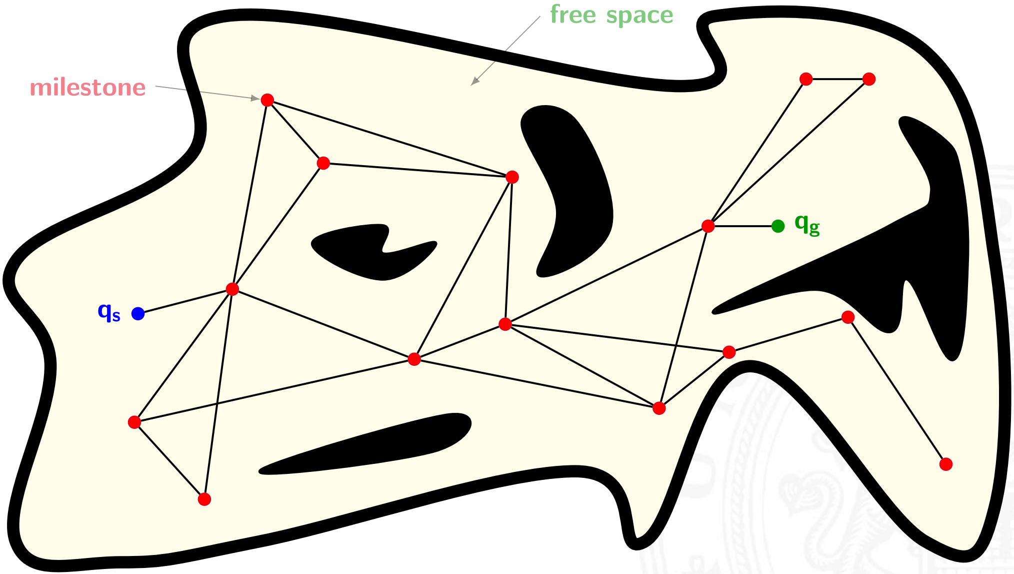
Milestones and Roadmap - Construction



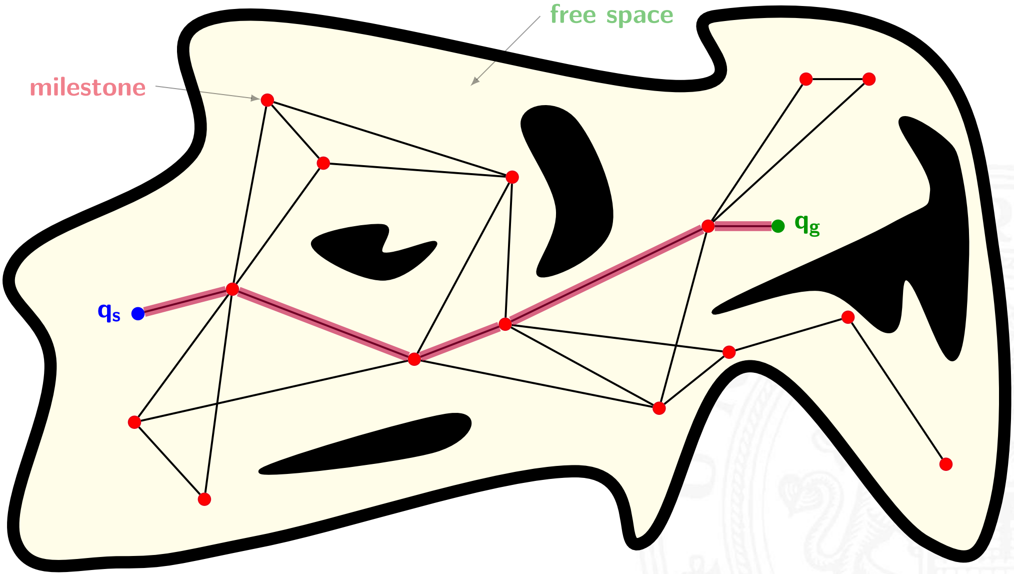
Milestones and Roadmap - Construction



Milestones and Roadmap - Query



Milestones and Roadmap - Query



Probabilistic Road Maps - Aspects

Algorithm: sPRM

```
V ← {x_init} ∪ {SampleFree_i}_{i=1,...,n}; E ← ∅;
foreach v ∈ V do
    U ← Near(G = (V, E), v, r) \ {v};
    foreach u ∈ U do
        if CollisionFree(v, u) then E ← E ∪ {(v, u), (u, v)}
return G = (V, E);
```

Probabilistic Road Maps - Aspects

Algorithm: sPRM

$V \leftarrow \{x_{\text{init}}\} \cup \{\texttt{SampleFree}_i\}_{i=1,\ldots,n}$; $E \leftarrow \emptyset$;
foreach $v \in V$ **do**
    $U \leftarrow \texttt{Near}(G = (V, E), v, r) \setminus \{v\}$;
    foreach $u \in U$ **do**
        if $\texttt{CollisionFree}(v, u)$ **then** $E \leftarrow E \cup \{(v, u), (u, v)\}$
return $G = (V, E)$;

- `SampleFree` - Sample states from $\mathcal{X}_{free}$
- `Near` - Choose Distance metric and threshold
- `CollisionFree(v,u)` - Check motion **between** states for collisions

Probabilistic Road Maps - Aspects

Algorithm: sPRM

```
1 V ← {x_init} ∪ {SampleFree_i}_{i=1,...,n}; E ← ∅;
2 foreach v ∈ V do
3     U ← Near(G = (V, E), v, r) \ {v};
4     foreach u ∈ U do
5         if CollisionFree(v, u) then E ← E ∪ {(v, u), (u, v)}
6 return G = (V, E);
```

`SampleFree` – sample states from $\mathcal{X}_{free}$

- Traditionally: Rejection Sampling
  Take samples uniformally, add sample if $x \in \mathcal{X}_{free}$
- Alternatives:
  - Projective Sampling: Replace samples $x \in \mathcal{X}_{obs}$ by closest state $x' \in \mathcal{X}_{free}$
  - Generative Sampling: For a sufficient **parameterized** space $\mathcal{X}'_{free} \subseteq \mathcal{X}_{free}$:
    Sample from $\mathcal{X}'_{free}$ via parameters

Probabilistic Road Maps - Aspects

Algorithm: sPRM

```
V ← {x_init} ∪ {SampleFree_i}_{i=1,...,n}; E ← ∅;
foreach v ∈ V do
    U ← Near(G = (V, E), v, r) \ {v};
    foreach u ∈ U do
        if CollisionFree(v, u) then E ← E ∪ {(v, u), (u, v)}
return G = (V, E);
```

Near - choose distance metric and threshold

- Traditional C-space metric: L_1 distance
- Obvious alternatives: weighted L_1 distance, L_2 distance

- Higher threshold: more negative collision checks
- Lower threshold: slower graph building

Probabilistic Road Maps - Aspects

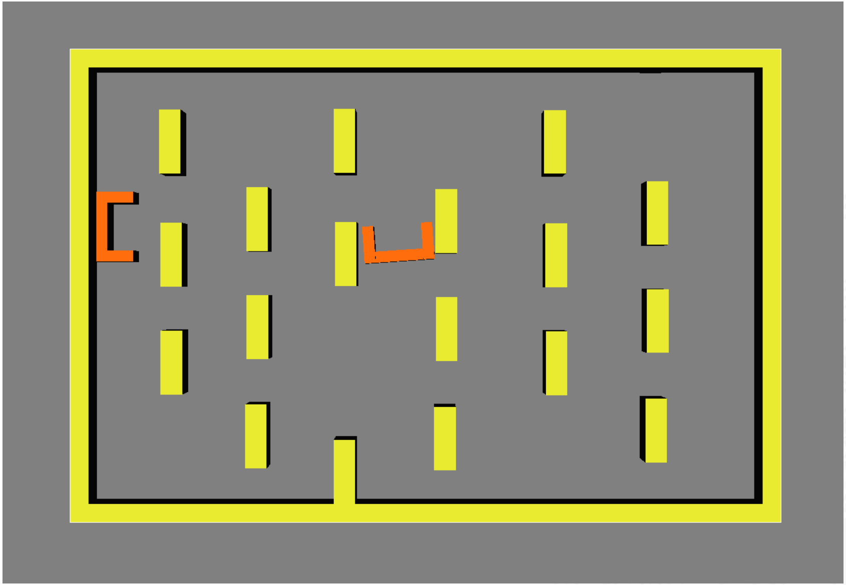
Algorithm: sPRM

1. $V \leftarrow \{x_{\text{init}}\} \cup \{\texttt{SampleFree}_i\}_{i=1,\dots,n}$; $E \leftarrow \emptyset$;
2. **foreach** $v \in V$ **do**
3. $\quad U \leftarrow \texttt{Near}(G = (V, E), v, r) \setminus \{v\}$;
4. $\quad$ **foreach** $u \in U$ **do**
5. $\quad\quad$ **if** $\texttt{CollisionFree}(v, u)$ **then** $E \leftarrow E \cup \{(v, u), (u, v)\}$
6. **return** $G = (V, E)$;

`CollisionFree(v, u)` - Local Planning

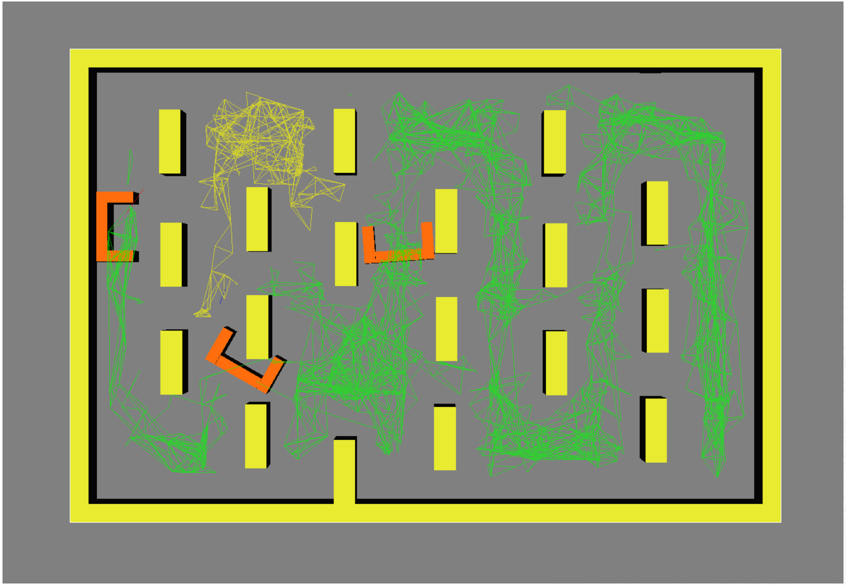
- Traditionally collision-checking tests **one** state
- Interpolate states between $\langle v, u \rangle$ and check those
  - Fixed step size in C-space can imply huge motions in workspace!
- Continuous collision checking (CCD):
  - Current systems rely on primitive motions
  - Robot links move in complex splines

Example



3dof planning problem

Example



3dof planning problem

Single and Multi-Query Planning

Definition

If only a single path is requested in a potentially changing scene, this is called **single-query** planning. If datastructures remain valid between motion requests, this is called **multi-query** planning.

PRM solves a multi-query problem by building an undirected graph.

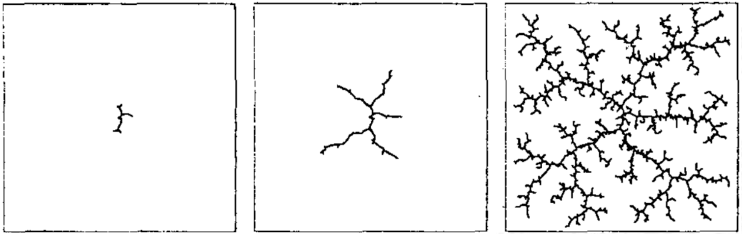
For single-shot planning, the graph search can be avoided altogether.

Rapidly-exploring Random Trees (RRT) - Basic Idea

Proposed by Kuffner and LaValle 2000 [18]

Instead of building a graph, grow a tree from the start state.

If for any leaf state $x \in \mathcal{X}_{goal}$, a solution is found.



RRT at multiple stages of extension

Rapidly-exploring Random Trees (RRT) - Algorithm

Algorithm 3: RRT

$V \leftarrow \{x_{\text{init}}\};\ E \leftarrow \emptyset;$
for $i = 1, \ldots, n$ **do**
    $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i;$
    $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}});$
    $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})\ ;$
    if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
        $V \leftarrow V \cup \{x_{\text{new}}\};\ E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}\ ;$
return $G = (V, E);$

Adapted from [15]

Rapidly-exploring Random Trees (RRT) - Algorithm

Algorithm 3: RRT

$V \leftarrow \{x_{\text{init}}\};\ E \leftarrow \emptyset;$
for $i = 1, \dots, n$ **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i;$
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}});$
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})\ ;$
  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $V \leftarrow V \cup \{x_{\text{new}}\};\ E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}\ ;$
return $G = (V, E);$

`Steer(x,y)` - Compute new state x'

- Move from x towards y: $\|y - x'\| < \|y - x\|$
- $\|x - x'\| < \eta$ to limit step size
- Alternatively compute closest $x' \in \mathcal{X}_{free}$ reachable via straight motion

Adapted from [15]

Rapidly-exploring Random Trees (RRT) - Algorithm

Algorithm 3: RRT

$V \leftarrow \{x_{\text{init}}\};\ E \leftarrow \emptyset;$

for $i = 1, \ldots, n$ **do**

  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i;$

  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}});$

  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})\ ;$

  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**

    $V \leftarrow V \cup \{x_{\text{new}}\};\ E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}\ ;$

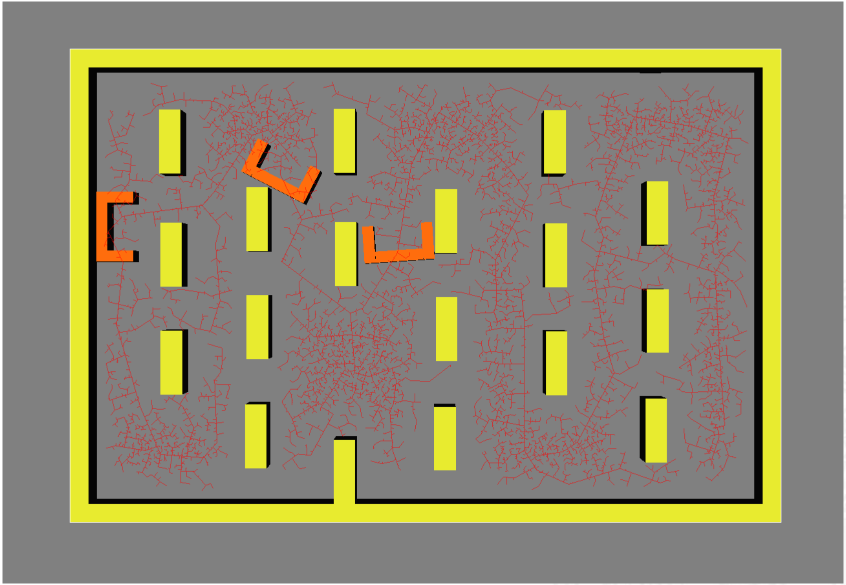
return $G = (V, E);$

`SampleFree` – sample states from $\mathcal{X}_{free}$

- Traditionally: uniform sampling
- To improve heuristically, a **Goal Bias** can be added
  - Low fraction of samples are sampled from $\mathcal{X}_{goal}$
  - Required if $\mathcal{X}_{goal}$ is small in $\mathcal{X}$

Adapted from [15]

Example



RRT graph of an example

Bi-Directional Search

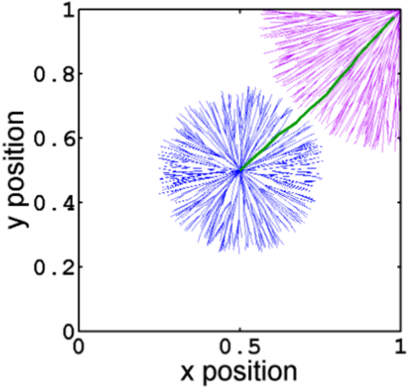
In robotics, start and goal are often in constraint areas of $\mathcal{X}_{free}$, e.g., close to obstacles.

The **transition phase** between these states is often quite flexible.

Instead of growing a single tree towards the goal

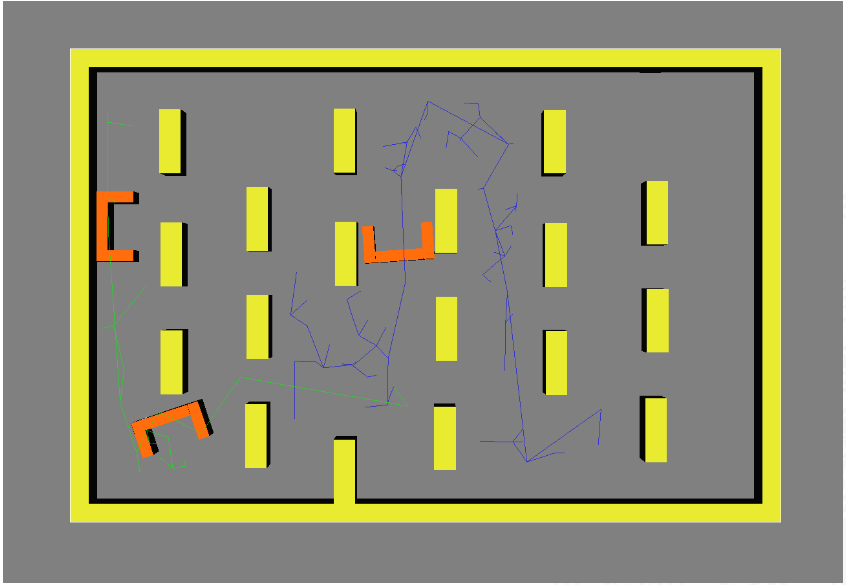
- Grow two trees from start and goal each.
- Attempt to connect them at each step.

In practice, this speeds up planning to the first solution significantly.



Bi-directional search trees [19]

RRT-Connect - Example



RRT-Connect for an example

Expansive Space Trees

PRM and RRT sample random configurations from $\mathcal{X}_{free}$.
Thus they also sample in areas which are already well-represented by milestones.

Definition

The *density* around a state x can be represented by the cardinality of its neighborhood within a distance d: $|N_d(x)|$.

Ideas

- Sample next expansion step weighted by inverse density $w(x) = \frac{1}{|N_d(x)|}$
- Stochastically reject samples in high-density areas

Hsu et.al. 1997 [20]

Expansive Space Trees

PRM and RRT sample random configurations from $\mathcal{X}_{free}$.
Thus they also sample in areas which are already well-represented by milestones.

Definition

The *density* around a state x can be represented by
the cardinality of its neighborhood within a distance d: $|N_d(x)|$.

Ideas

- Sample next expansion step weighted by inverse density $w(x) = \frac{1}{|N_d(x)|}$
- Stochastically reject samples in high-density areas

Hsu et.al. 1997 [20]

Expansive Space Trees

PRM and RRT sample random configurations from $\mathcal{X}_{free}$.
Thus they also sample in areas which are already well-represented by milestones.

Definition

The *density* around a state x can be represented by
the cardinality of its neighborhood within a distance d: $|N_d(x)|$.

Ideas

- Sample next expansion step weighted by inverse density $w(x) = \frac{1}{|N_d(x)|}$
- Stochastically reject samples in high-density areas

Hsu et.al. 1997 [20]

Expansive Space Trees - Algorithm

Algorithm *expansion*

1. Pick a node x from V with probability $1/w(x)$.
2. Sample K points from $N_d(x) = \{q \in \mathcal{C} \mid dist_c(q, x) < d\}$, where $dist_c$ is some distance metric of $\mathcal{C}$. (K and d are parameters.)
3. **for** each configuration y that has been picked **do**
4. calculate $w(y)$ and retain y with probability $1/w(y)$.
5. **if** y is retained, $clearance(y) > 0$ and $link(x, y)$ returns YES
6. **then** put y in V and place an edge between x and y.

- Expand from an existing node instead of global samples from $\mathcal{X}$
- Samples rejected in 4. are never collision checked!
- Original formulation is bidirectional

Hsu et.al. 1997 [20]

Expansive Space Trees - Algorithm

Algorithm *expansion*

1. Pick a node x from V with probability $1/w(x)$.
2. Sample K points from $N_d(x) = \{q \in \mathcal{C} \mid dist_c(q, x) < d\}$, where $dist_c$ is some distance metric of $\mathcal{C}$. (K and d are parameters.)
3. **for** each configuration y that has been picked **do**
4. calculate $w(y)$ and retain y with probability $1/w(y)$.
5. **if** y is retained, $clearance(y) > 0$ and $link(x, y)$ returns YES
6. **then** put y in V and place an edge between x and y.

- Expand from an existing node instead of global samples from $\mathcal{X}$
- Samples rejected in 4. are never collision checked!
- Original formulation is bidirectional

Hsu et.al. 1997 [20]

Expansive Space Trees - Algorithm

Algorithm *expansion*

1. Pick a node x from V with probability $1/w(x)$.
2. Sample K points from $N_d(x) = \{q \in \mathcal{C} \mid dist_c(q,x) < d\}$, where $dist_c$ is some distance metric of $\mathcal{C}$. (K and d are parameters.)
3. **for** each configuration y that has been picked **do**
4. calculate $w(y)$ and retain y with probability $1/w(y)$.
5. **if** y is retained, $clearance(y) > 0$ and $link(x,y)$ returns YES
6. **then** put y in V and place an edge between x and y.

- Expand from an existing node instead of global samples from $\mathcal{X}$
- Samples rejected in 4. are never collision checked!
- Original formulation is bidirectional

Hsu et.al. 1997 [20]

Expansive Space Trees - Algorithm

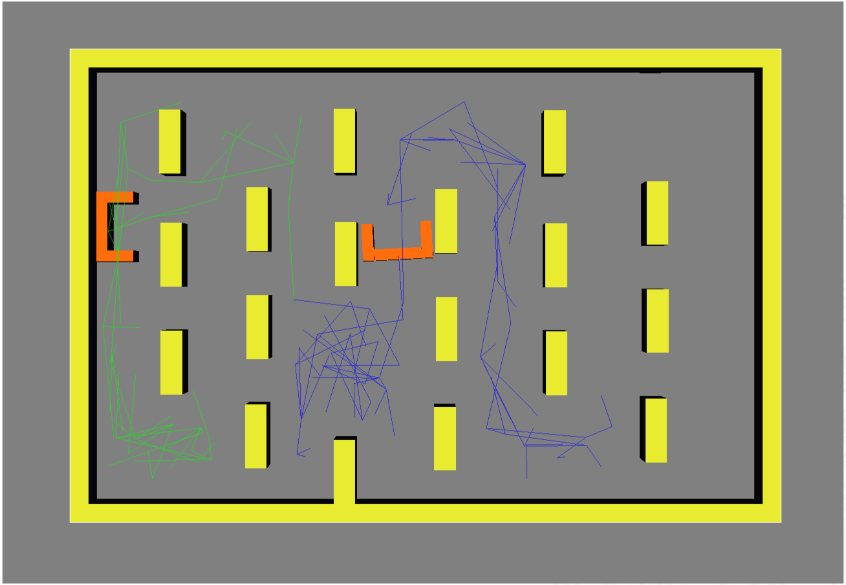
Algorithm *expansion*

1. Pick a node x from V with probability $1/w(x)$.
2. Sample K points from $N_d(x) = \{q \in \mathcal{C} \mid dist_c(q, x) < d\}$, where $dist_c$ is some distance metric of $\mathcal{C}$. (K and d are parameters.)
3. **for** each configuration y that has been picked **do**
4. calculate $w(y)$ and retain y with probability $1/w(y)$.
5. **if** y is retained, $clearance(y) > 0$ and $link(x, y)$ returns YES
6. **then** put y in V and place an edge between x and y.

- Expand from an existing node instead of global samples from $\mathcal{X}$
- Samples rejected in 4. are never collision checked!
- Original formulation is bidirectional

Hsu et.al. 1997 [20]

(Bi)EST - Example



(Bi-directional) EST for an example

Path Postprocessing

The resulting paths are not smooth and often contain unnecessary motions.

Traditional post-processing includes:

- Path Shortcutting
  - Repeatedly pick two non-consecutive waypoints and attempt to connect them
- Perturbation of individual waypoints
  - Optional
  - Can reduce solution costs
  - Computationally expensive
  - For differentiable costs: exploit gradient
- Fit smooth splines through waypoints

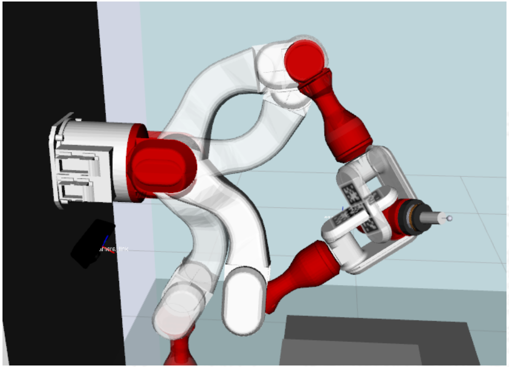
All modifications need to be collision checked.

Multiple IK Solutions

Redundant robots generate multiple joint solutions per pose.

Each Cartesian goal region adds a number of disjoint C-space goal regions.

Most tree-based planners naturally extend to **Multi-Goal Planning**, implicitly building multiple goal trees.



Multiple IK solutions for one target pose © Hendrich

Definition

An **Optimal Path Planning Problem** is defined by a path planning problem $\mathcal{P} = \langle \mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal} \rangle$ and a cost function $c(\tau) : R \geq 0$. It requires to find a feasible path τ^* such that $\tau^* = \text{argmin}_\tau \{ c(\tau) \mid \tau \text{ is feasible for } \mathcal{P} \}$

In practice:

- Two-step process:
  - Find feasible path(s)
  - Optimize path(s)
- Planners are **asymptotically** optimal
  - Convergence might take long
  - Non-trivial to detect ε-optimal solution
- What cost function should be used?
  - C-space path length
  - Accumulated clearance (distance to obstacles)
  - Cartesian end-effector path length
  - Physical work

Definition

An **Optimal Path Planning Problem** is defined by a path planning problem $\mathcal{P} = \langle \mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal} \rangle$ and a cost function $c(\tau) : R \geq 0$. It requires to find a feasible path τ^* such that $\tau^* = \text{argmin}_\tau \{ c(\tau) \mid \tau \text{ is feasible for } \mathcal{P} \}$

In practice:

- Two-step process:
  - Find feasible path(s)
  - Optimize path(s)
- Planners are **asymptotically** optimal
  - Convergence might take long
  - Non-trivial to detect ε-optimal solution
- What cost function should be used?
  - C-space path length
  - Accumulated clearance (distance to obstacles)
  - Cartesian end-effector path length
  - Physical work

Optimal Planning

Definition

An **Optimal Path Planning Problem** is defined by a path planning problem $\mathcal{P} = \langle \mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal} \rangle$ and a cost function $c(\tau) : R \geq 0$. It requires to find a feasible path τ^* such that $\tau^* = \mathrm{argmin}_\tau \{ c(\tau) \mid \tau \text{ is feasible for } \mathcal{P} \}$

In practice:

- Two-step process:
  - Find feasible path(s)
  - Optimize path(s)
- Planners are **asymptotically** optimal
  - Convergence might take long
  - Non-trivial to detect ε-optimal solution
- What cost function should be used?
  - C-space path length
  - Accumulated clearance (distance to obstacles)
  - Cartesian end-effector path length
  - Physical work

Optimal Planning

Definition

An **Optimal Path Planning Problem** is defined by a path planning problem $\mathcal{P} = \langle \mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal} \rangle$ and a cost function $c(\tau) : R \geq 0$. It requires to find a feasible path τ^* such that $\tau^* = \text{argmin}_\tau \{c(\tau) \mid \tau \text{ is feasible for } \mathcal{P}\}$

In practice:

- Two-step process:
  - Find feasible path(s)
  - Optimize path(s)
- Planners are **asymptotically** optimal
  - Convergence might take long
  - Non-trivial to detect ε-optimal solution
- What cost function should be used?
  - C-space path length
  - Accumulated clearance (distance to obstacles)
  - Cartesian end-effector path length
  - Physical work

Planner*

Method

Instead of stopping at the first trajectory, continue sampling to improve solution.

Karaman and Frazzoli 2011 [15] introduced **PRM*** and **RRT***.
Both are efficient, asymptotically optimal versions of the basic algorithms.

PRM is asymptotically optimal as-is.

- Eventually all points on the optimal path will be added to the roadmap.

Ensure minimal required graph connectivity of $O(n \cdot \log(n))$.

- Reduce the neighborhood radius r with sample size n:

$$r(n) = \gamma_{PRM} \cdot (\frac{log(n)}{n})^{\frac{1}{d}}$$

where γ_{PRM} depends on the planning space, d is the dimensionality of $\mathcal{X}$

Method

Update tree whenever new samples yield cheaper paths to root.

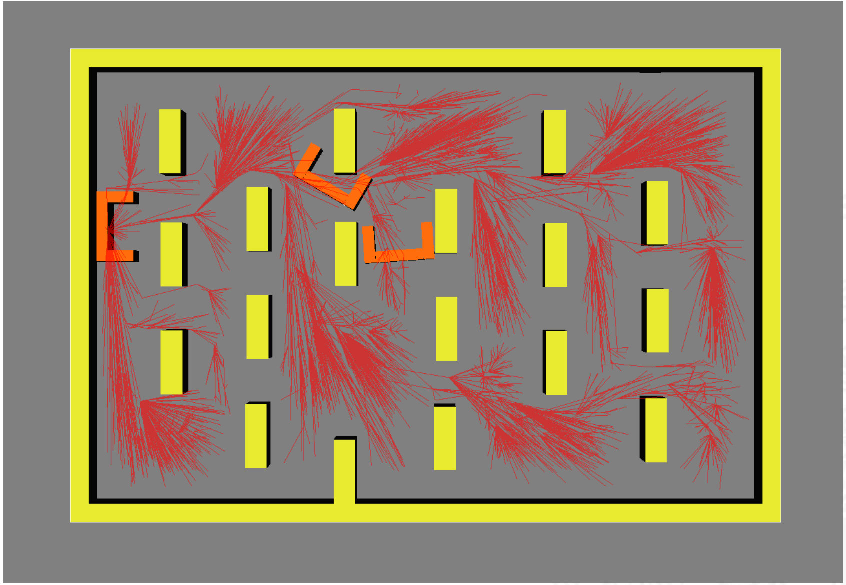
- Instead of connecting the new states to *closest node*, connect to the *cheapest node* in neighborhood
- Change parent of neighboring states to new state if new path is cheaper

RRT* - Algorithm

Algorithm 6: RRT*

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
    $V \leftarrow V \cup \{x_{\text{new}}\}$;
    $x_{\text{min}} \leftarrow x_{\text{nearest}}$; $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{nearest}}) + c(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}))$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Connect along a minimum-cost path
      if $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\text{min}}$ **then**
        $x_{\text{min}} \leftarrow x_{\text{near}}$; $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$
    $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Rewire the tree
      if $\texttt{CollisionFree}(x_{\text{new}}, x_{\text{near}}) \wedge \texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}})) < \texttt{Cost}(x_{\text{near}})$
      then $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

RRT* - Example



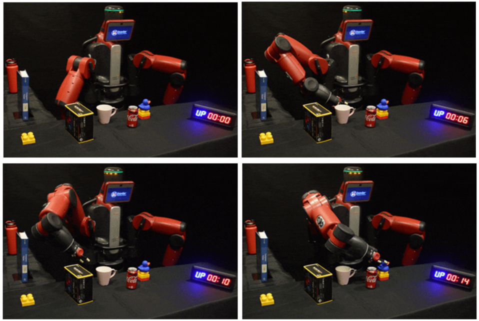
RRT* for an example

Summary - Sampling Based Planning

- Represent $\mathcal{X}_{free}$ probabilistically through samples
- Relies heavily on binary collision checking
- Post-processing solutions is essential
- Various (dozens) of algorithms with varying performance
- Straight-forward extensions for asymptotically optimal planning

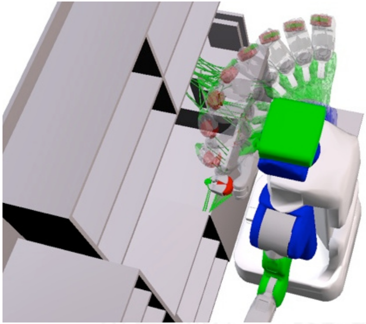
Beyond Sampling-Based Planning

MPNet



Fast deep-learning system learning from planners [21]

TrajOpt



Sequential convex optimizer solving trajectories [22]

Bibliography

[1] G.-Z. Yang, R. J. Full, N. Jacobstein, P. Fischer, J. Bellingham, H. Choset, H. Christensen, P. Dario, B. J. Nelson, and R. Taylor, “Ten robotics technologies of the year,” 2019.

[2] J. K. Yim, E. K. Wang, and R. S. Fearing, “Drift-free roll and pitch estimation for high-acceleration hopping,” in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8986–8992, IEEE, 2019.

[3] J. F. Engelberger, *Robotics in service*.
MIT Press, 1989.

[4] K. Fu, R. González, and C. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*.
McGraw-Hill series in CAD/CAM robotics and computer vision, McGraw-Hill, 1987.

[5] R. Paul, *Robot Manipulators: Mathematics, Programming, and Control: the Computer Control of Robot Manipulators*.
Artificial Intelligence Series, MIT Press, 1981.

[6] J. Craig, *Introduction to Robotics: Pearson New International Edition: Mechanics and Control*.
Always learning, Pearson Education, Limited, 2013.

Bibliography (cont.)

[7] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[8] T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, 2009.

[9] W. Böhm, G. Farin, and J. Kahmann, "A Survey of Curve and Surface Methods in CAGD," *Comput. Aided Geom. Des.*, vol. 1, pp. 1–60, July 1984.

[10] J. Zhang and A. Knoll, "Constructing Fuzzy Controllers with B-spline Models - Principles and Applications," *International Journal of Intelligent Systems*, vol. 13, no. 2-3, pp. 257–285, 1998.

[11] M. Eck and H. Hoppe, "Automatic Reconstruction of B-spline Surfaces of Arbitrary Topological Type," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), pp. 325–334, ACM, 1996.

Bibliography (cont.)

[12] A. Cowley, W. Marshall, B. Cohen, and C. J. Taylor, "Depth space collision detection for motion planning," 2013.

[13] Hornung, Armin and Wurm, Kai M. and Bennewitz, Maren and Stachniss, Cyrill and Burgard, Wolfram, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, pp. 189–206, 2013.

[14] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE International Conference on Robotics and Automation*, pp. 625–632, 2009.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[16] O. Khatib, "The Potential Field Approach and Operational Space Formulation in Robot Control," in *Adaptive and Learning Systems*, pp. 367–377, Springer, 1986.

[17] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[18] J. Kuffner and S. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning.," vol. 2, pp. 995–1001, 01 2000.

[19] J. Starek, J. Gómez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," *Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2015, 07 2015.

[20] D. Hsu, J. . Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of International Conference on Robotics and Automation*, vol. 3, pp. 2719–2726 vol.3, 1997.

[21] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2118–2124, IEEE, 2019.

[22] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *in Proc. Robotics: Science and Systems*, 2013.

Bibliography (cont.)

[23] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[24] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[25] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1470–1477, 2011.

[26] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach.," in *Robotics: Science and Systems*, pp. 1–6, 2016.

[27] J. Ferrer-Mestres, G. Francès, and H. Geffner, "Combined task and motion planning as classical ai planning," *arXiv preprint arXiv:1706.06927*, 2017.

[28] M. Görner, R. Haschke, H. Ritter, and J. Zhang, "MoveIt! Task Constructor for Task-Level Motion Planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Bibliography (cont.)

[29] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010.

[30] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[31] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141, IEEE, 2018.

[32] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[33] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, pp. 14–23, Mar 1986.

[34] M. J. Mataric, "Interaction and intelligent behavior.," tech. rep., DTIC Document, 1994.

[35] M. P. Georgeff and A. L. Lansky, "Reactive reasoning and planning.," in *AAAI*, vol. 87, pp. 677–682, 1987.

[36] J. S. Albus, "The nist real-time control system (rcs): an approach to intelligent systems research," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 157–174, 1997.

[37] T. Fukuda and T. Shibata, "Hierarchical intelligent control for robotic motion by using fuzzy, artificial intelligence, and neural network," in *Neural Networks, 1992. IJCNN., International Joint Conference on*, vol. 1, pp. 269–274 vol.1, Jun 1992.

[38] L. Einig, *Hierarchical Plan Generation and Selection for Shortest Plans based on Experienced Execution Duration*.
Master thesis, Universität Hamburg, 2015.

[39] J. Craig, *Introduction to Robotics: Mechanics & Control. Solutions Manual*.
Addison-Wesley Pub. Co., 1986.

Bibliography (cont.)

[40] H. Siegert and S. Bocionek, *Robotik: Programmierung intelligenter Roboter: Programmierung intelligenter Roboter.* Springer-Lehrbuch, Springer Berlin Heidelberg, 2013.

[41] R. Schilling, *Fundamentals of robotics: analysis and control.* Prentice Hall, 1990.

[42] T. Yoshikawa, *Foundations of Robotics: Analysis and Control.* Cambridge, MA, USA: MIT Press, 1990.

[43] M. Spong, *Robot Dynamics And Control.* Wiley India Pvt. Limited, 2008.