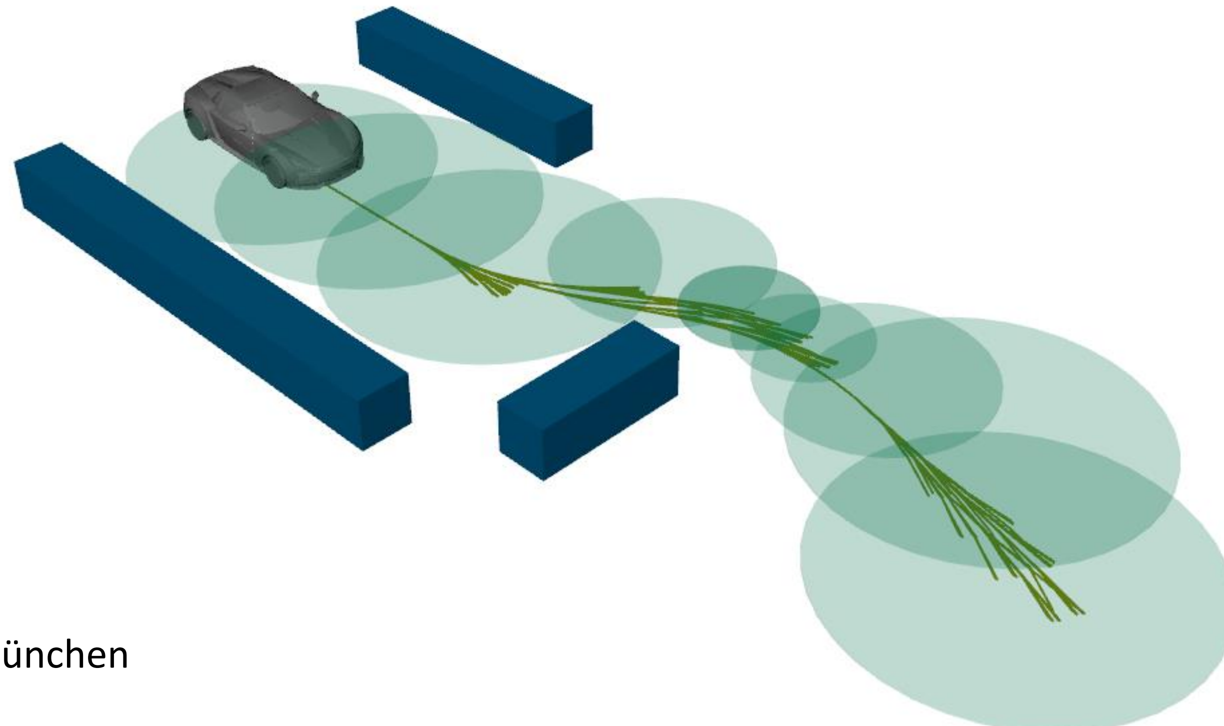
Robotics & Embedded Systems

# Autonomes Fahren
SS 2019

# Path Planning and Motion Planning

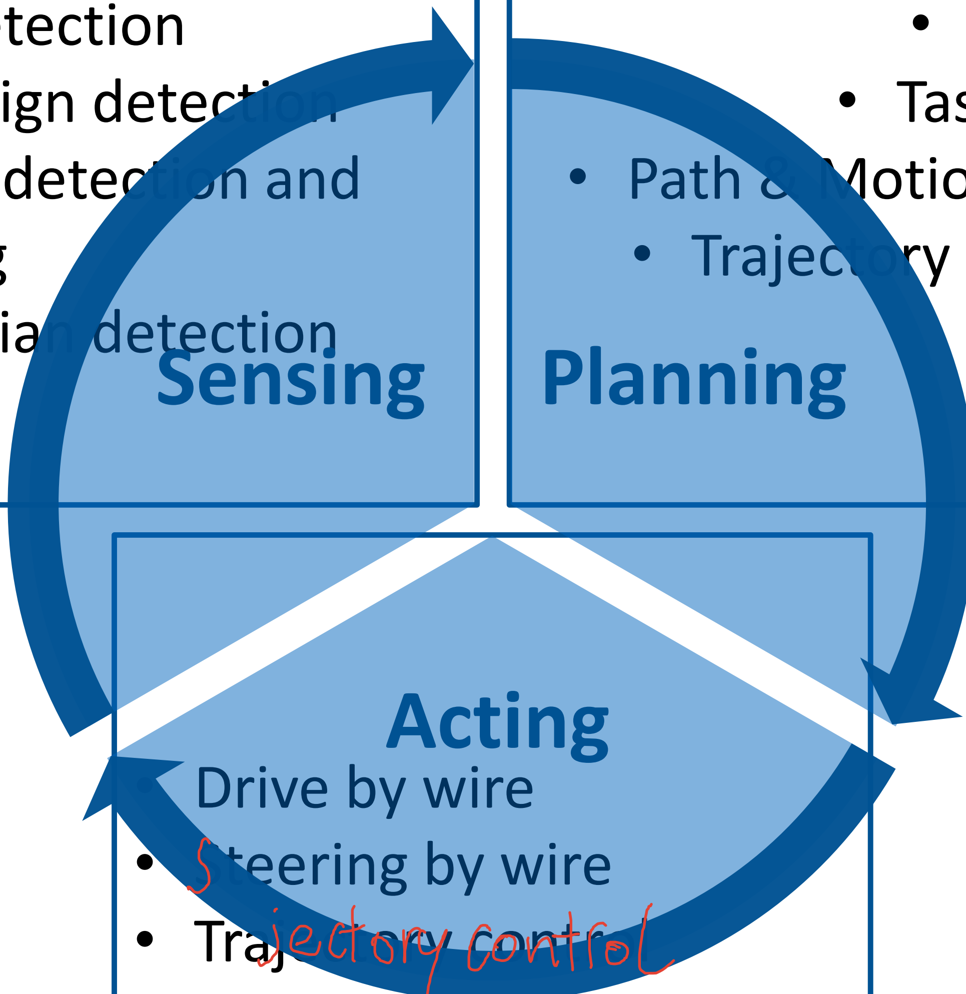


Technische Universität München

# Motion planning for autonomous driving

- Lane detection
- Traffic sign detection
- Vehicle detection and tracking
- Pedestrian detection

- Navigation
- Task planning
- Path & Motion planning
- Trajectory generation




- Drive by wire
- Steering by wire
- Trajectory control
- Emergency brake

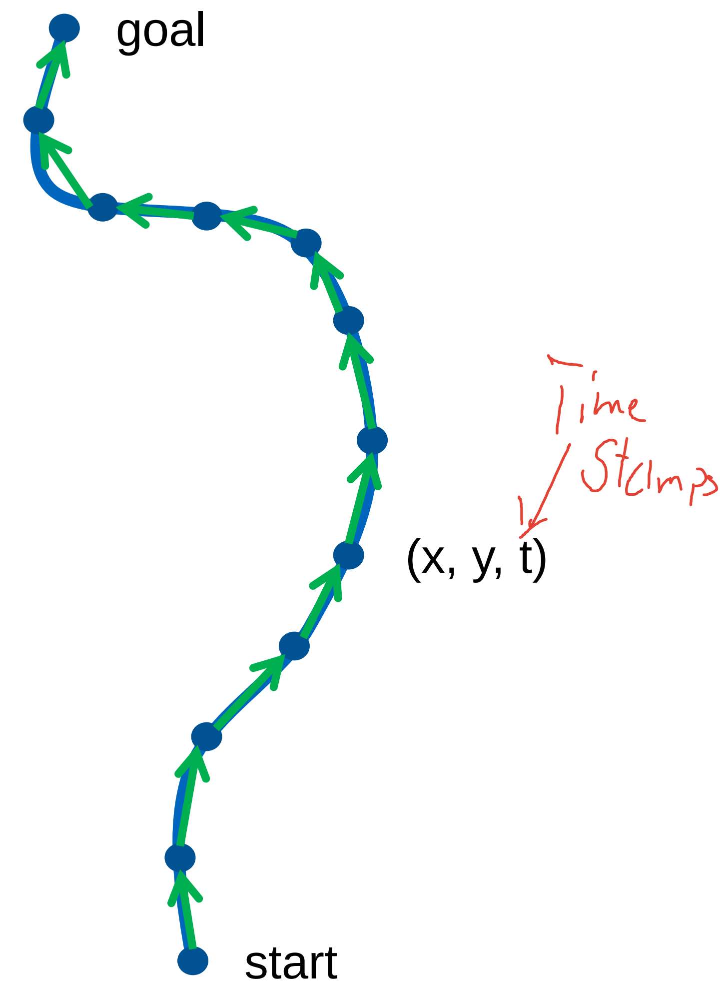
trajectory control

# Outlines

- Problem Definitions
  - Path, trajectory and motion
  - Configuration space and workspace
  - Constraints
  - Motion planning problem
  - Correct, optimal and complete
- Planning Methods
  - Combinatorial methods
  - Behavior-based methods
  - Random sampling methods
  - Search methods

# Path, Trajectory and Motion

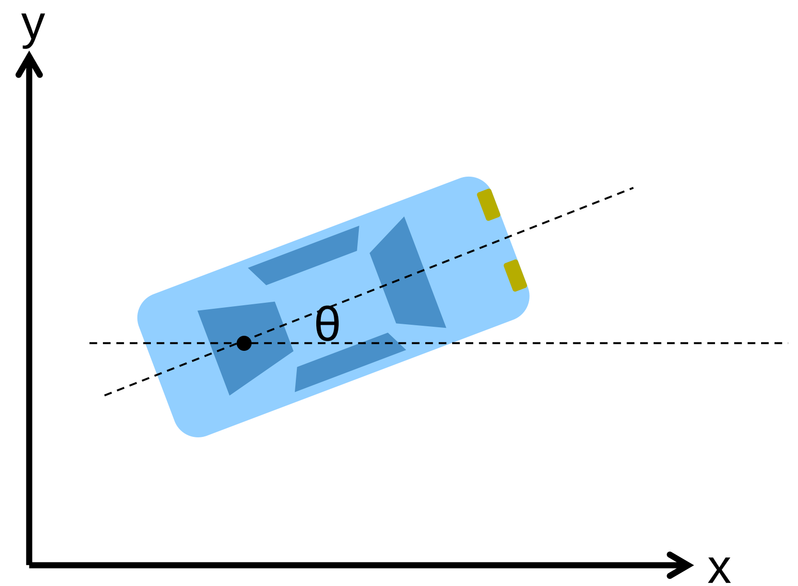
- Path
  - The geometric form of a motion from start to goal
  - A list of poses
- Motion
  - The movement along the path regarding the physical laws
  - A list of control inputs
- Trajectory
  - The result of a motion
  - A list of poses with timestamps



# Configuration Space and Workspace

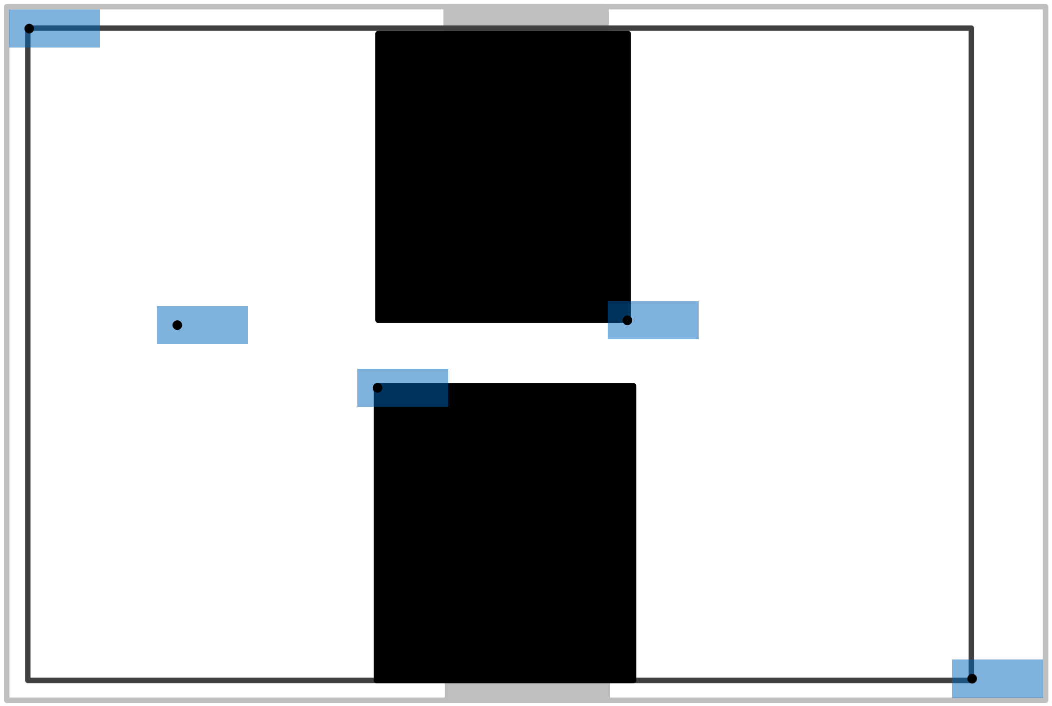
non honomomic
⇒ we are not free to move
from to everywhere

- Configuration space
  - General coordinate system: (x, y, θ)
- Workspace
  - Physical Cartesian space: (x, y)



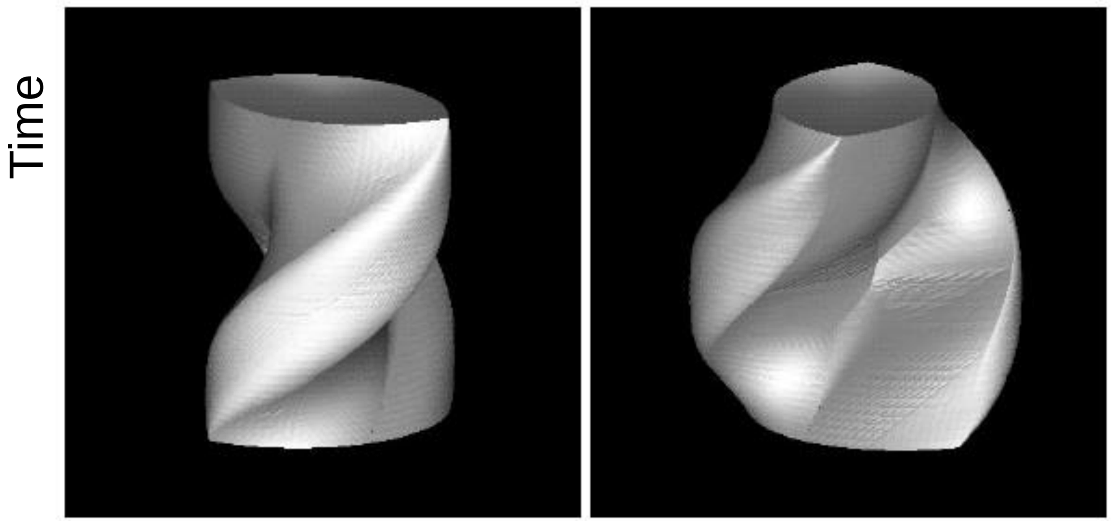
# Constraints
## Holonomic Constraints

- Only depends on the general coordinates
  - $f(x_0, x_1, \cdots, x_n, t) = 0$
- Example: obstacles, no rotation



# Constraints
## Nonholonomic Constraints

- Depends on the general coordinates and the time derivatives
  - $f(x_0, x_1, \cdots, x_n, \dot{x_0}, \dot{x_1}, \cdots, \dot{x_n}, t) = 0$
- Example: vehicle kinematics with limited steering radius




Reachable configuration

Accessible domain of shortest path with fixed length for Reeds-Shepp car [Laumond1998]

# Motion Planning Problem

- Given:
  - Configuration space
  - Constraints: obstacles, kinematics
  - Start and goal
- To find:
  - Path
  - Motion
  - Trajectory

# Correct, Optimal and Complete

- Correct
  - All the path internal configurations are valid (collision-free)
  - The motion between the configurations are executable
- Optimal
  - Time or distance
  - Safety: low risk
  - Comfort: smooth
  - Eco: energy consumption
- Complete
  - Decide whether a solution exists in a finite amount of time.
  - If a solution exists, return one solution in finite time.

# Planning Methods

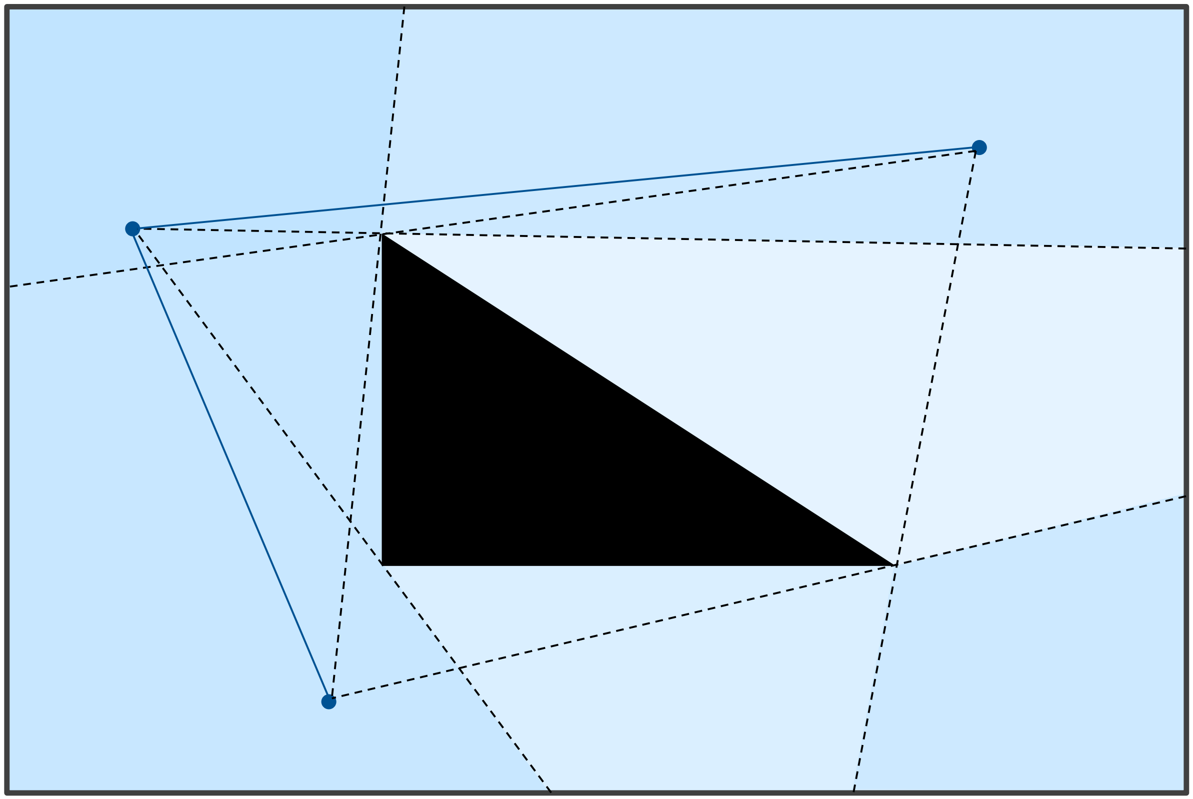
- Combinatorial Methods
- Behavior-based Methods
- Random Sampling Methods
- Search Methods

# Combinatorial Methods
## General Ideas

- Directly solve the path planning problem based on the geometry of the configuration space
- The motion is calculated regarding an explicit motion metric

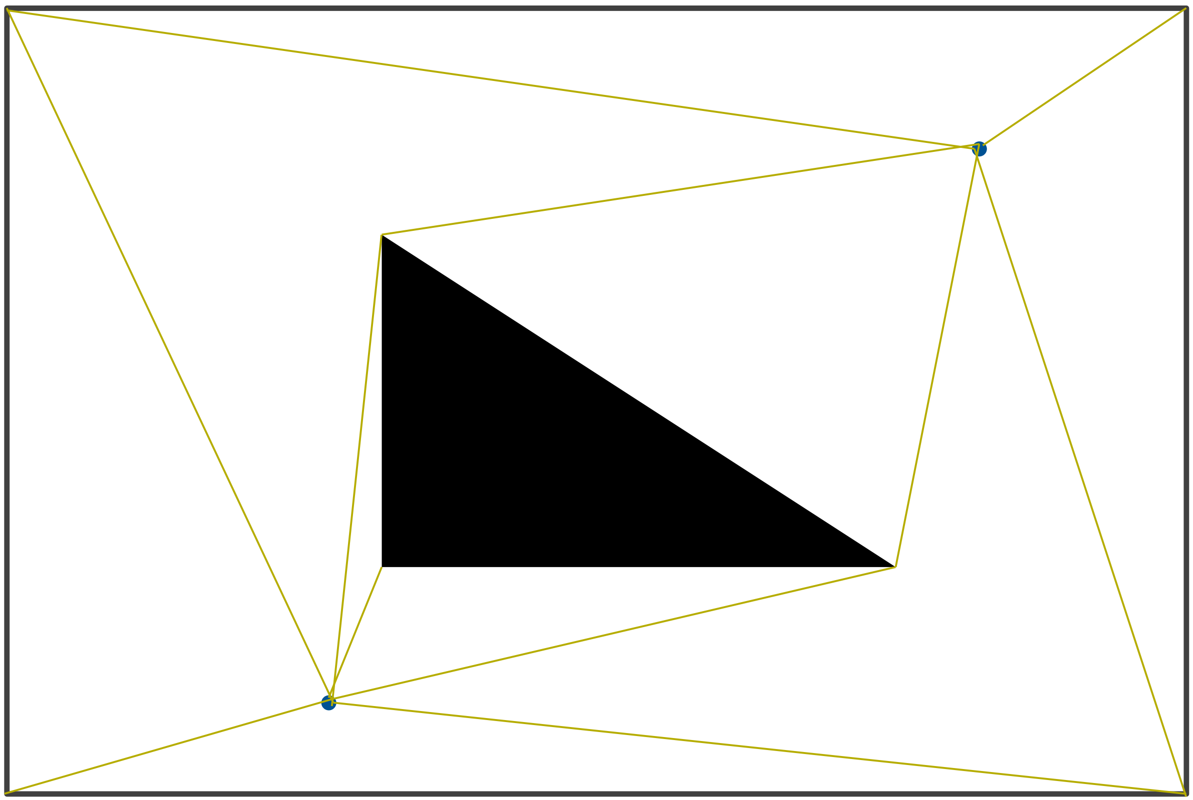
# Combinatorial Methods
## Visibility Map

- Build a roadmap based on a set of points, whose fields of view cover the whole configuration space.



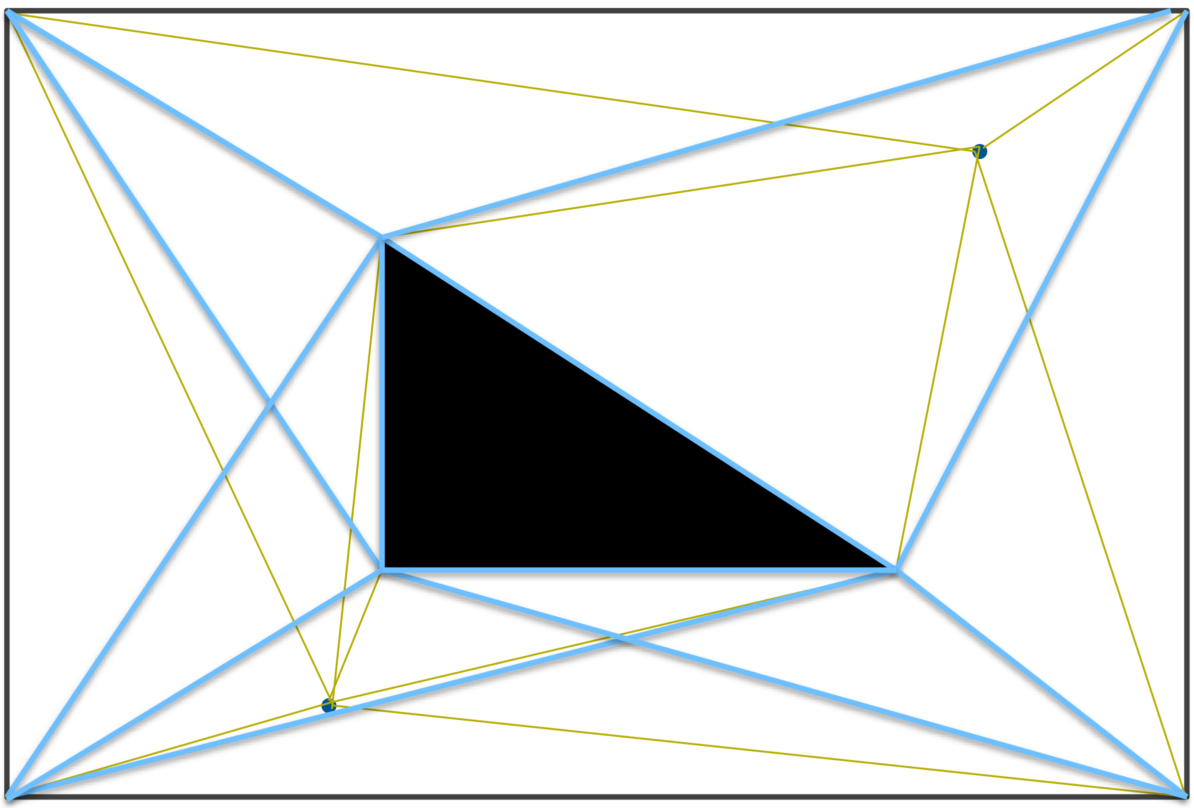
# Combinatorial Methods
## Visibility Map

- Build a roadmap based on a set of points, whose fields of view cover the whole configuration space.



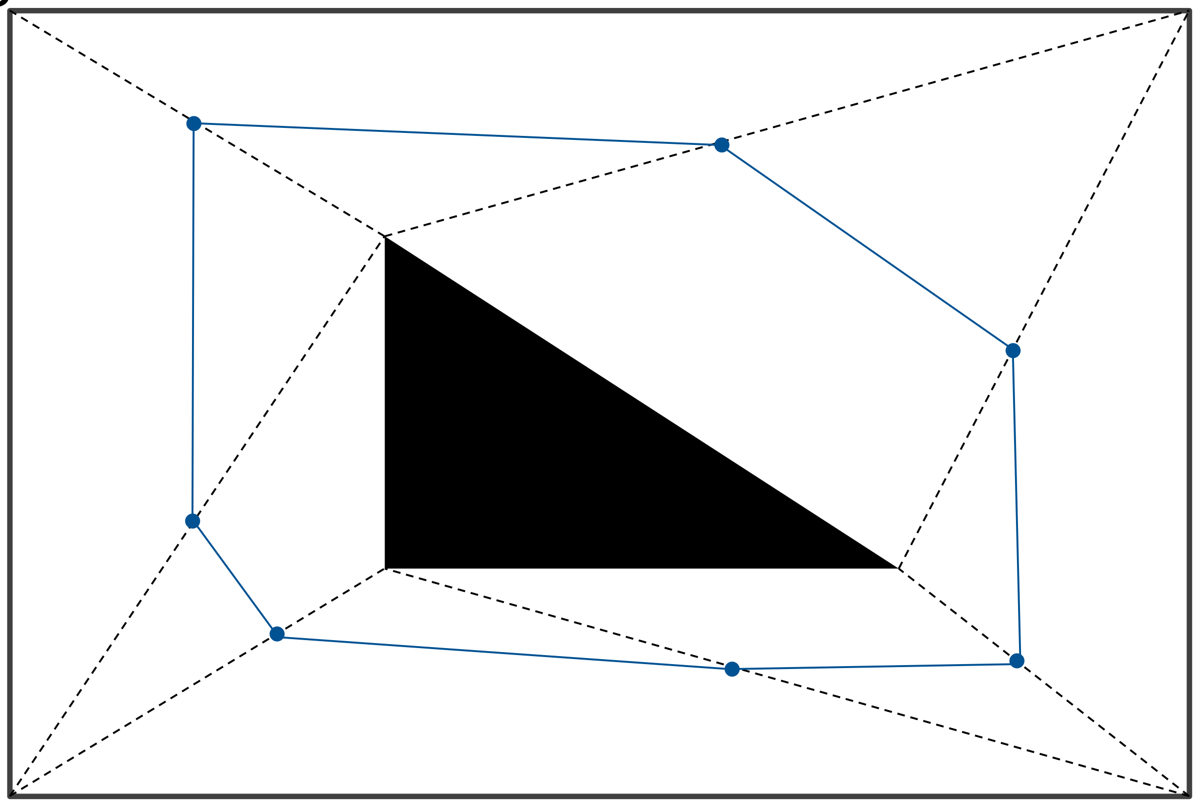
# Combinatorial Methods
## Visibility Map

- Build a roadmap based on a set of points, whose fields of view cover the whole configuration space.



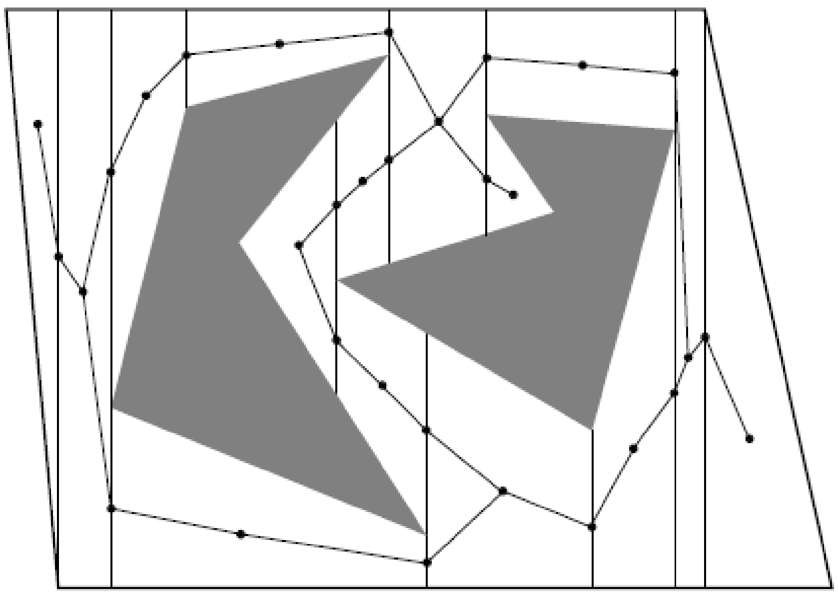
# Combinatorial Methods
## Space Decomposition

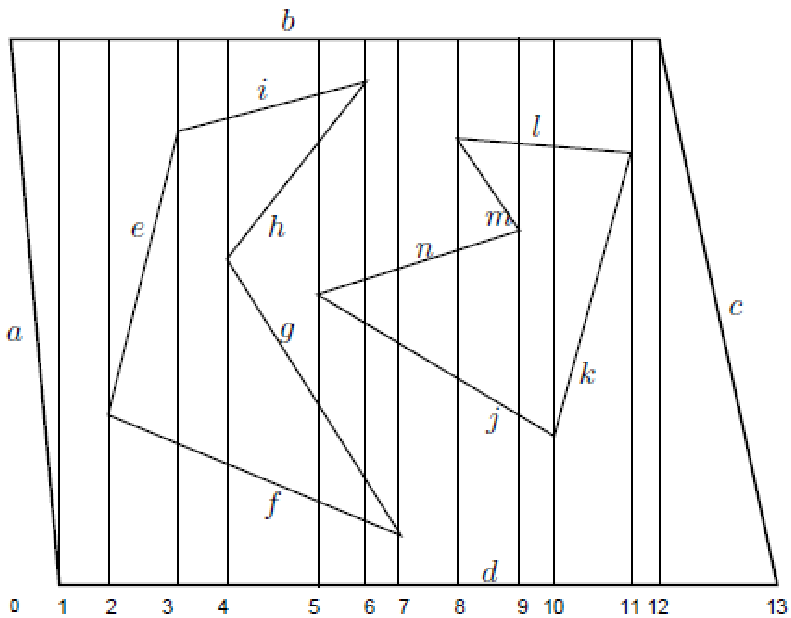
- Build a roadmap based on the space decomposition
- Triangulation



# Combinatorial Methods
## Space Decomposition

**Trapezoidal decomposition** **Cylindrical decomposition**

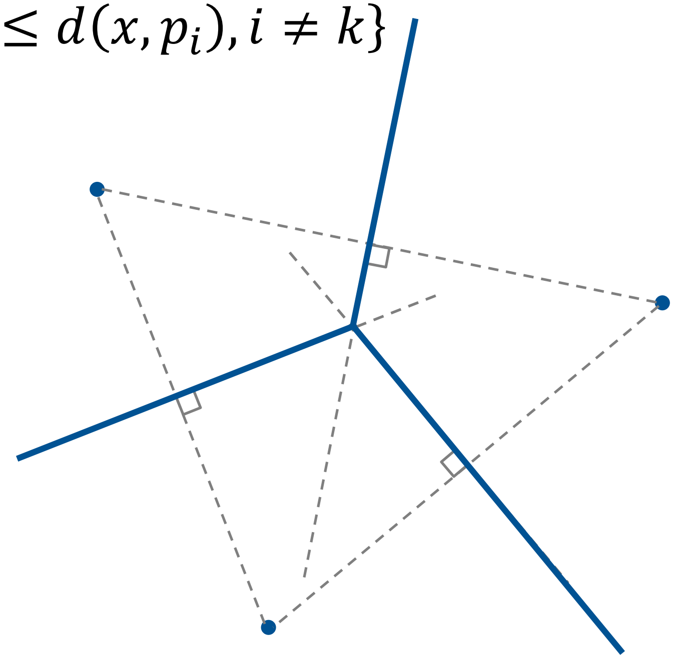




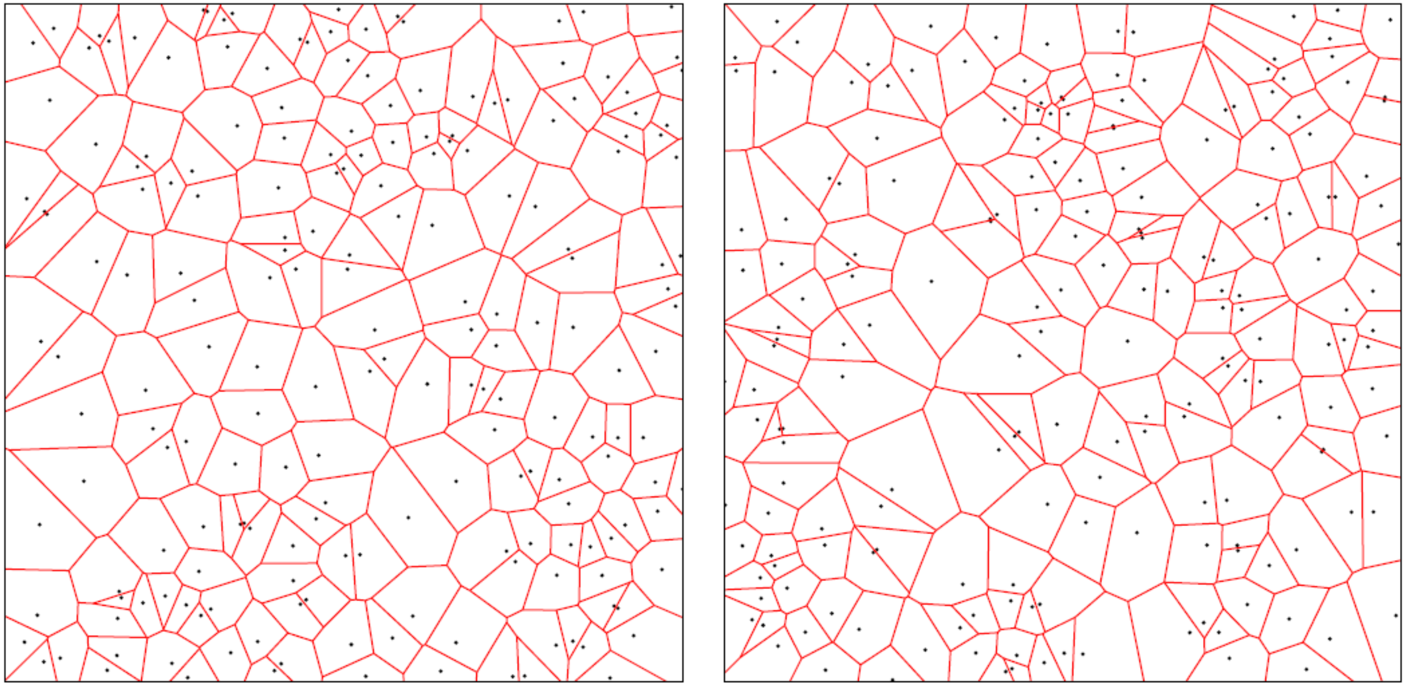

[LaValle2004]

# Combinatorial Methods
## Voronoi Decomposition

- Decompose the configuration space with a set of objects $\{p_n\}$
- Each object $p_k$ has a region $R_k$, where all points are closer to this object than the others.
  - $R_k = \{x | d(x, p_k) \leq d(x, p_i), i \neq k\}$



# Combinatorial Methods
## Voronoi Diagram



Voronoi diagram of 196 pseudorandom samples
[LaVallle2004]

# Combinatorial Methods
## Summary

- Complete
- Geometric representation of the configuration space is important
- Easily applicable for motion planning with holonomic constraints

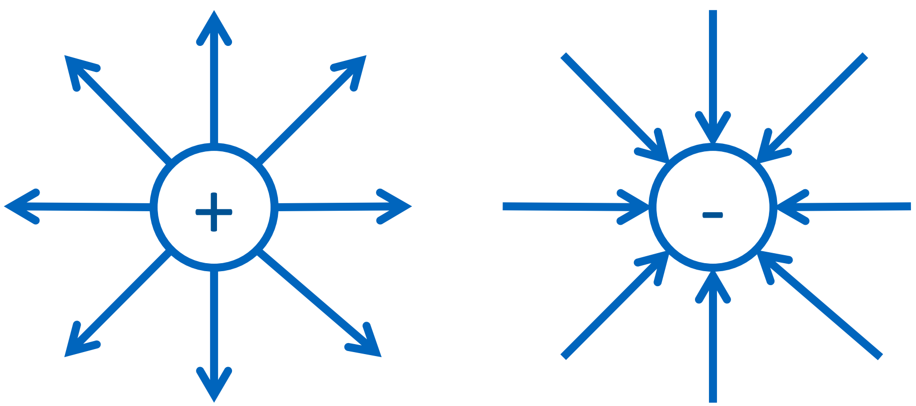
# Behavior-based Method

## General Ideas

- A reactive motion strategy for the whole configuration space
- The complex behavior is a reflex of the sophisticated environment with simple sensing-acting schema and a few internal states

# Behavior-based Method
## Artificial Potential Fields

- Conservative force and potential field
  - The work is independent of the path
  - Superposition of potential fields by sum




$$U = k_e \frac{Q}{r}, \qquad F = k_e \frac{q_1 q_2}{r^2}$$

$$U = -G \frac{M}{r}, \qquad F = G \frac{m_1 m_2}{r^2}$$

$$U = k \frac{d^2}{2}, \qquad F = kd$$

# Behavior-based Method

## Artificial Potential Fields

- Artificial potential field

$$U(x) = U_a(x) + U_r(x)$$

- Attractive potential field

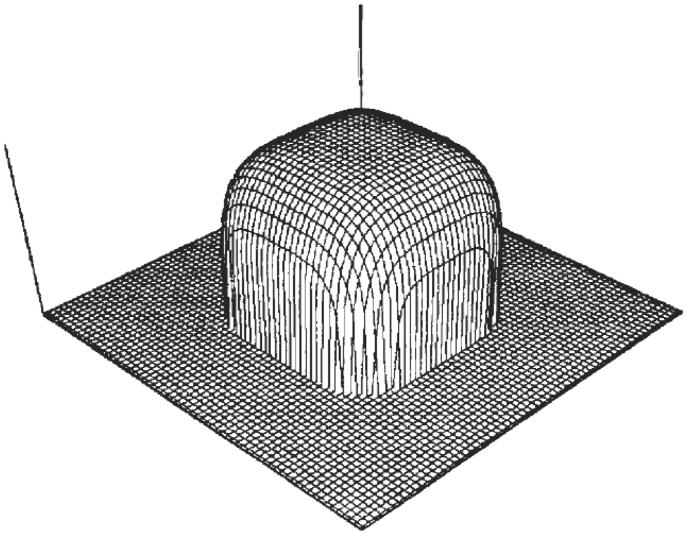
$$U_a(x) = \frac{1}{2} \mathrm{k} d_{goal}(x)^2$$

$$F_a(x) = \nabla U_a(x) = \mathrm{k} d_{goal}(x) \nabla d_{goal}(x)$$

- Repulsive potential field

$$U_r(x) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{d(x)} - \frac{1}{D}\right)^2 & d(x) \leq D \\ 0 & d(x) > D \end{cases}$$

$$F_r(x) = \nabla U_r(x) = \begin{cases} -\eta\left(\frac{1}{d(x)} - \frac{1}{D}\right)\frac{1}{d^2(x)} \nabla d(x) & d(x) \leq D \\ 0 & d(x) > D \end{cases}$$



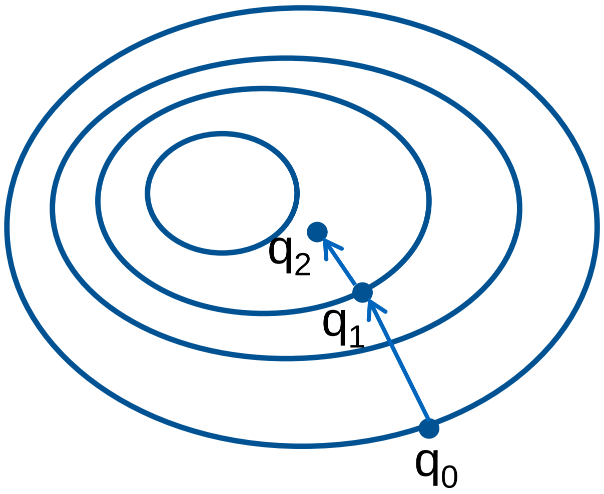
[Khatib1986]

# Behavior-based Method

## Artificial Potential Fields

- Gradient descent

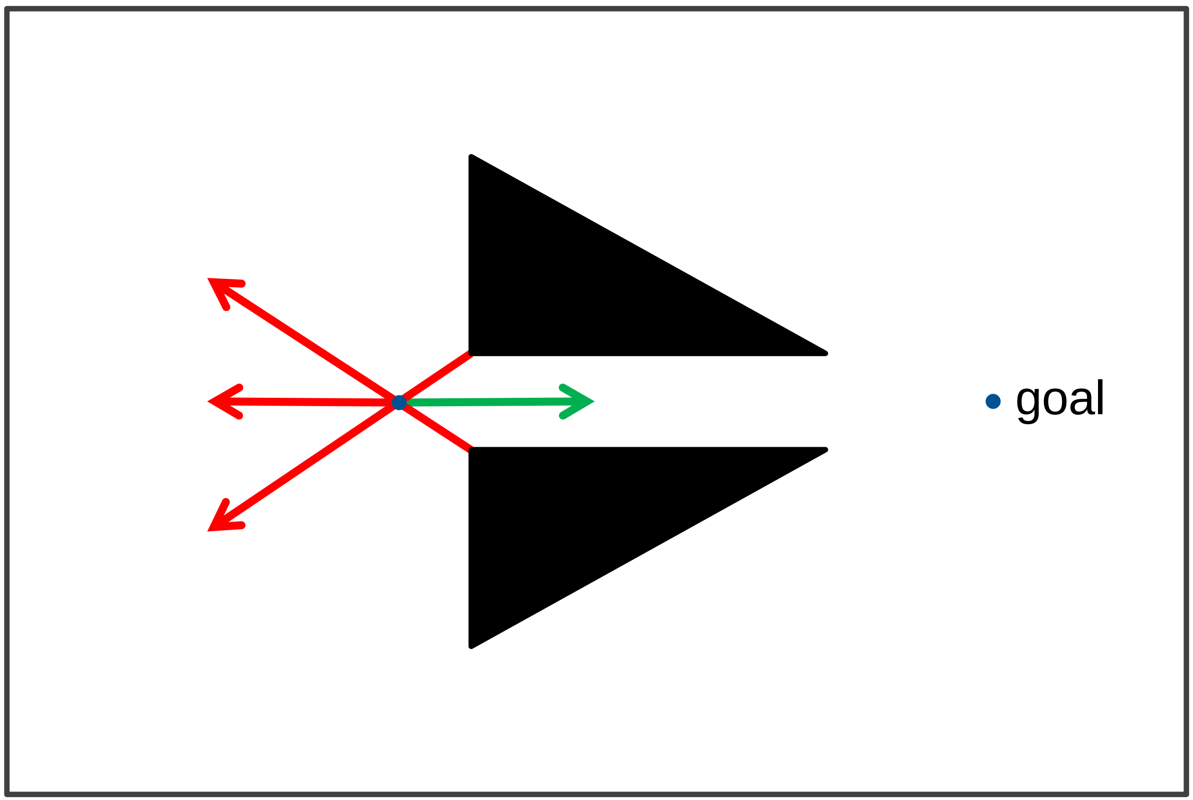
$q_0 = q_{start}$;
For ($i = 0$; $F(q_i) \neq 0$; $++i$)
    $q_{i+1} = q_i + s_i * F(q_i)$;




  - $s_i$ is the step size of the iteration i

# Behavior-based Method
## Artificial Potential Fields

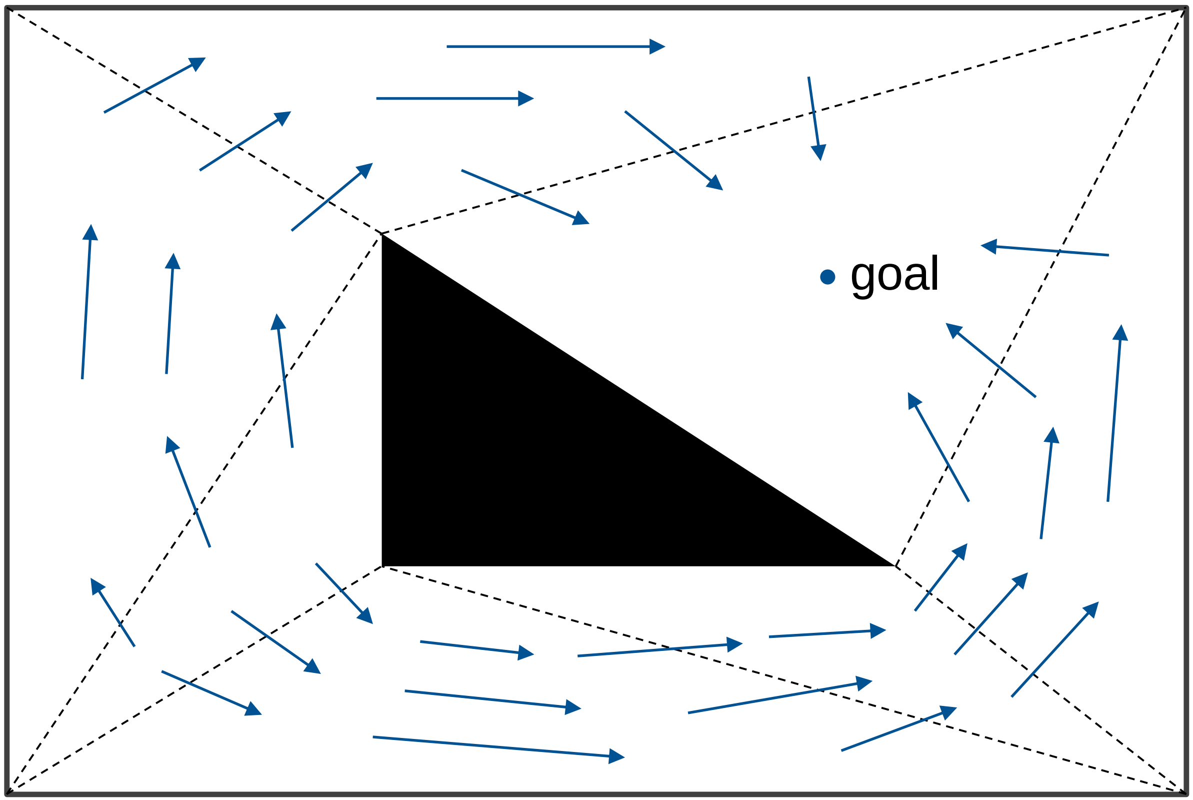
- Local minima



# Behavior-based Method

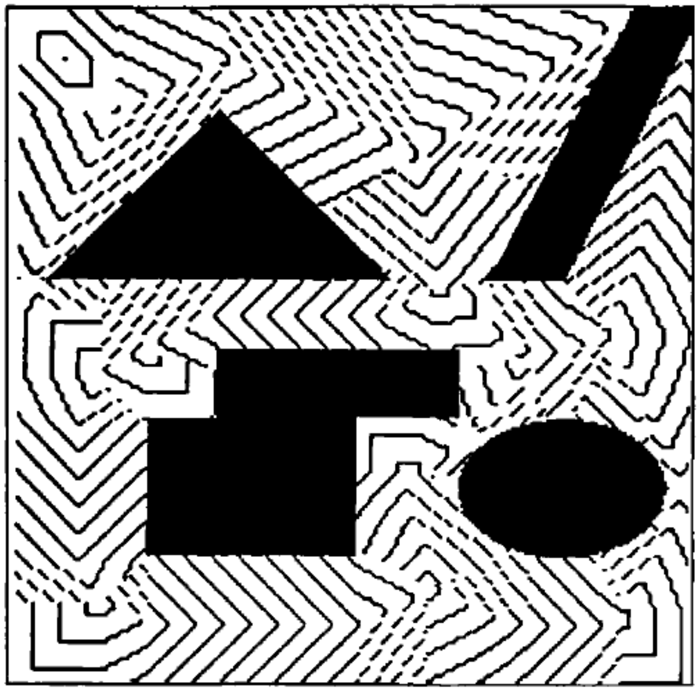
## Artificial Potential Fields

- Combine with space decomposition

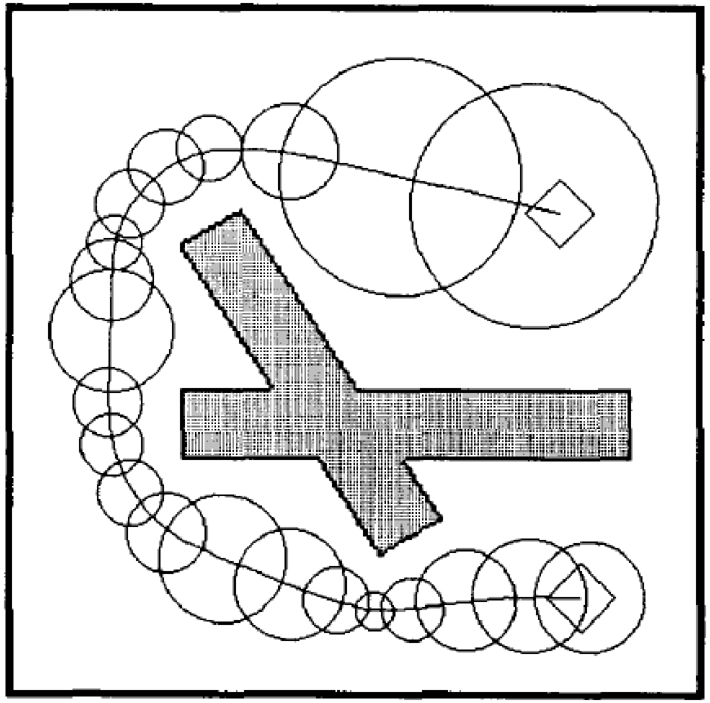


# Behavior-based Method
## Artificial Potential Fields

- Global navigating function



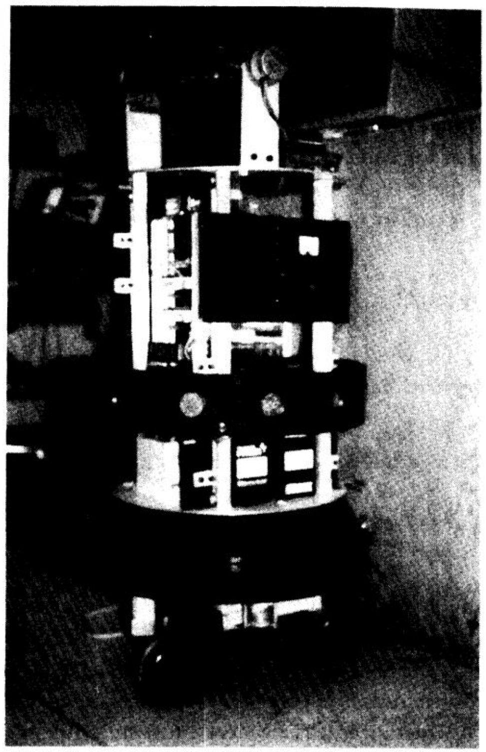
[Barraquand1991]

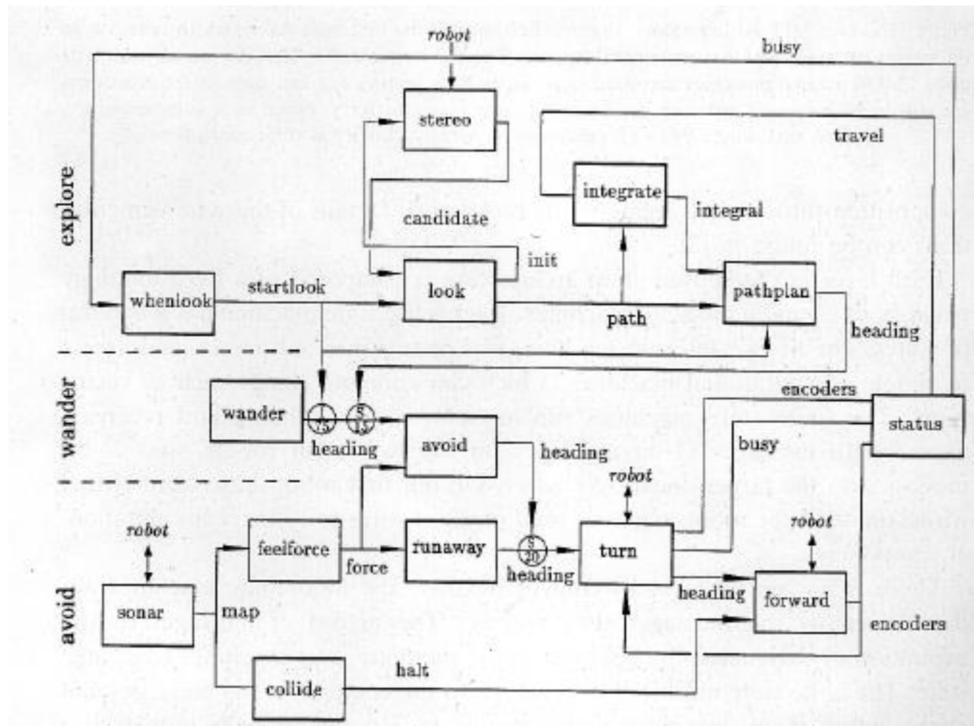


[Quinlan1993]

# Behavior-based Method
## Combined Behaviors

- Combination with simple behaviors






[Brooks1991]

# Behavior-based Method
## Autonomous Parking

# Behavior-based Method

## Summary

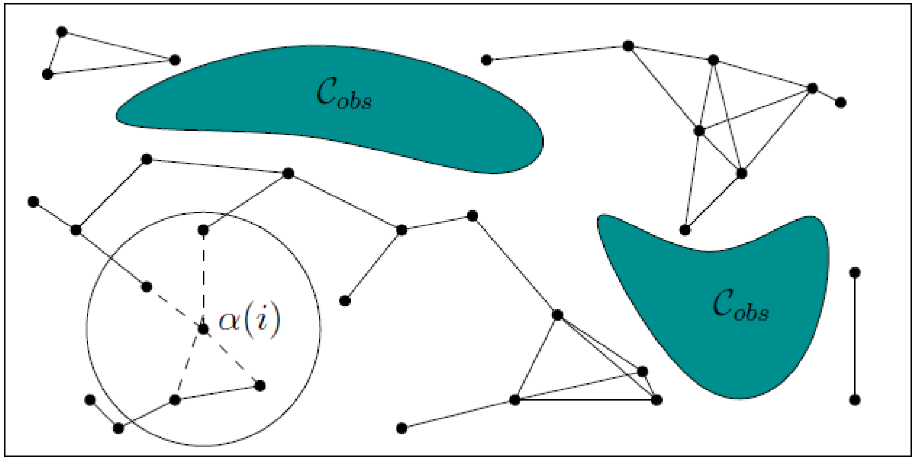
- Simple motion strategies for the whole configuration space
- A global guidance is required to escape the local minima
- Efficient for local motions

# Random Sampling Methods
## General Ideas

- Explore the configuration space with random samples
- Create a graph in the configuration space which connects the start and goal

# Random Sampling Methods
## Probabilistic Roadmaps (PRM)

- Idea:
  - Select a random valid configuration q
  - Find the nearest neighbors $N_q$
  - Connect q with $N_q$
- Algorithm




[Kavraki1996]

N ← Ø; N: nodes
E ← Ø; E: edges
While(1)
    q ← random tree configuration;
    $N_q$ ← nearest neighbors of q from N;
    N ← N + {q};
    for all n ϵ $N_q$
        if unconnected(q, n) and exists_path(q, n)
            E ← E + {(q, n)};

# Random Sampling Methods

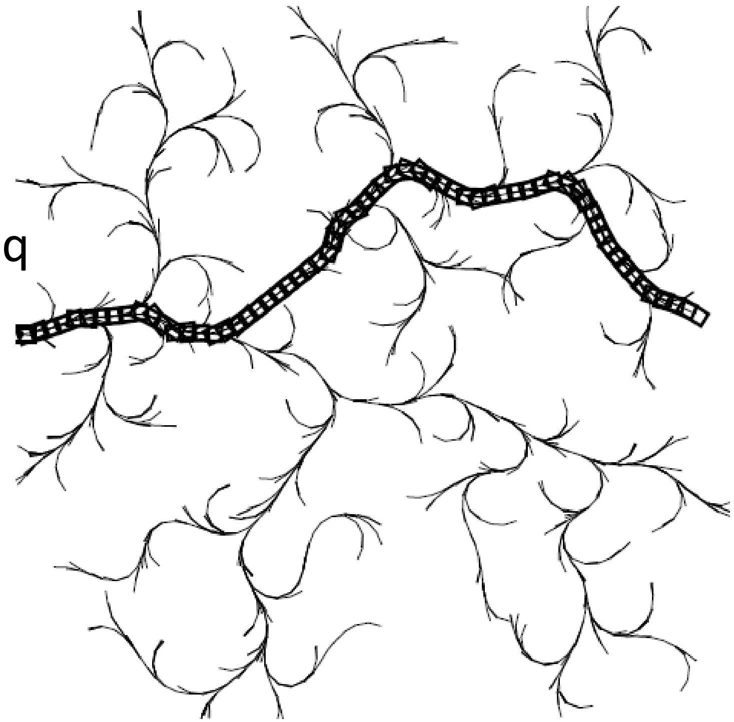
## Probabilistic Roadmaps (PRM)

- Explicit motion metric is required to connect the configurations
- Can be used as a global strategy, and employ other planning methods for the local connection
- The map can be reused for multiple queries

# Random Sampling Methods
## Rapid-exploring Random Trees (RRT)

- Idea:
  - Select a random configuration q
  - Find the nearest neighbor $q_n$
  - Create a new configuration from $q_n$ to q
- Algorithm:

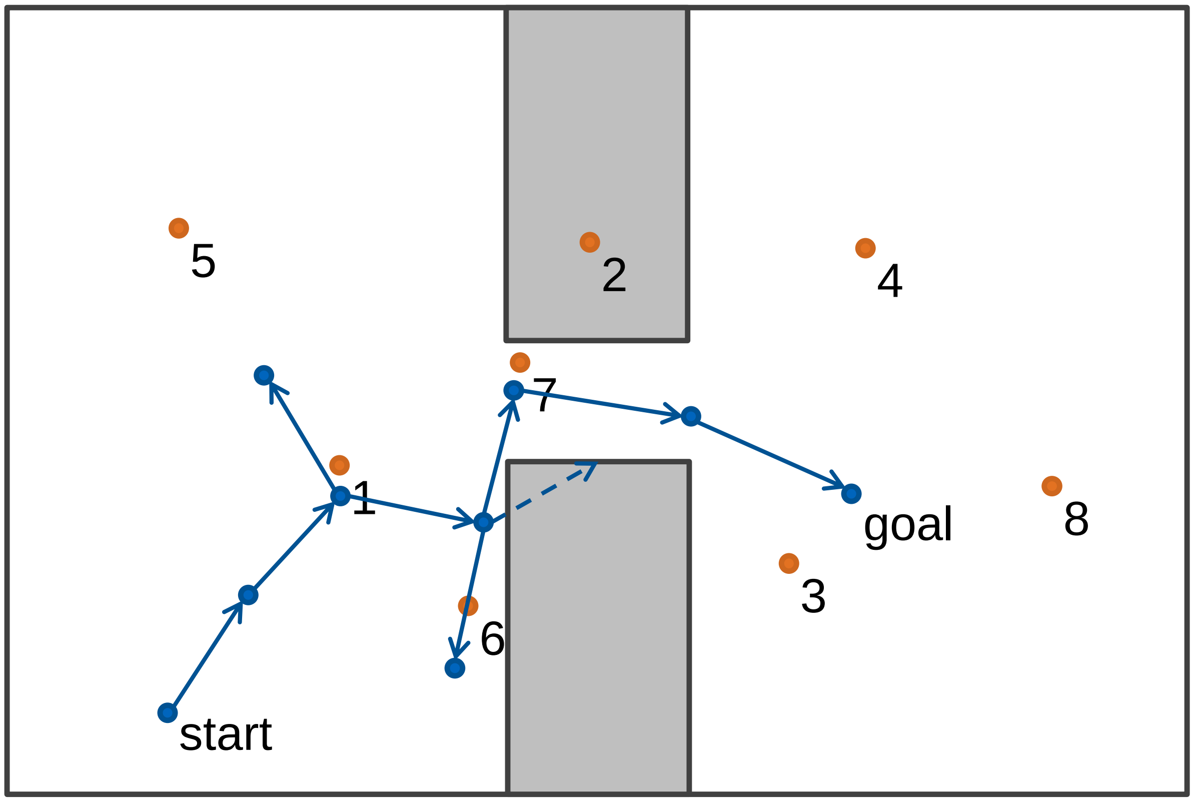
```
N ← q_start;
E ← Ø;
While(1)
      q ← random configuration;
      q_n ← nearest neighbor of q in N;
      q_i ← create_state(q, q_n, q_goal);
      N ← N + {q_i};
      E ← E + {(q_n, q_i)};
```



[LaValle1998]

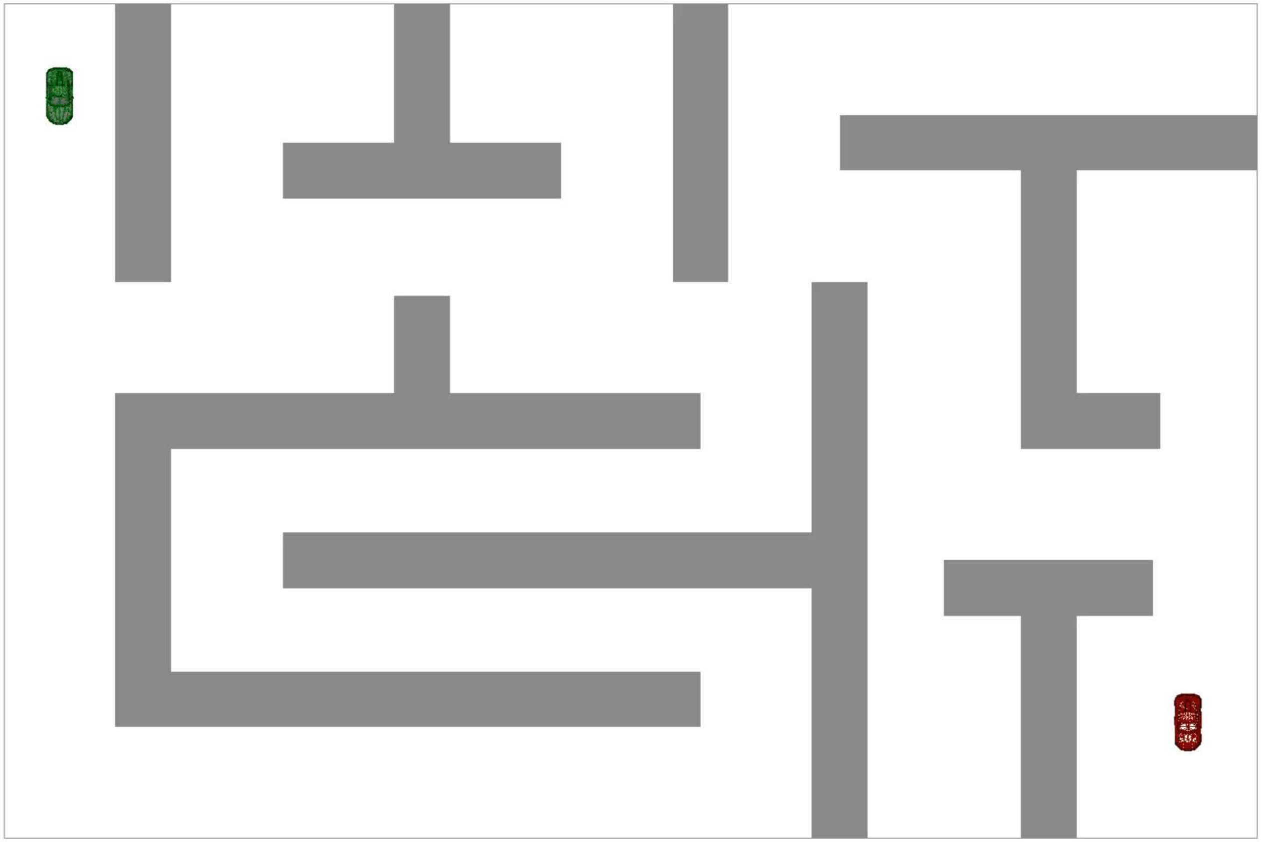
# Random Sampling Methods
## Rapid-exploring Random Trees (RRT)

- RRT Example



# Random Sampling Methods
## Rapid-exploring Random Trees (RRT)

- RRT with forward dynamics



# Random Sampling Methods
## Rapid-exploring Random Trees (RRT)

- The tree starts from the initial configuration and grows with bias to the goal configuration
- Does not require explicit motion metric of the configuration space
- Can apply the forwards kinematics/dynamics of nonholonomic vehicles

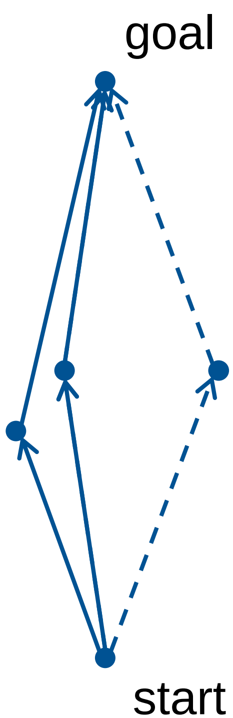
# Random Sampling Methods

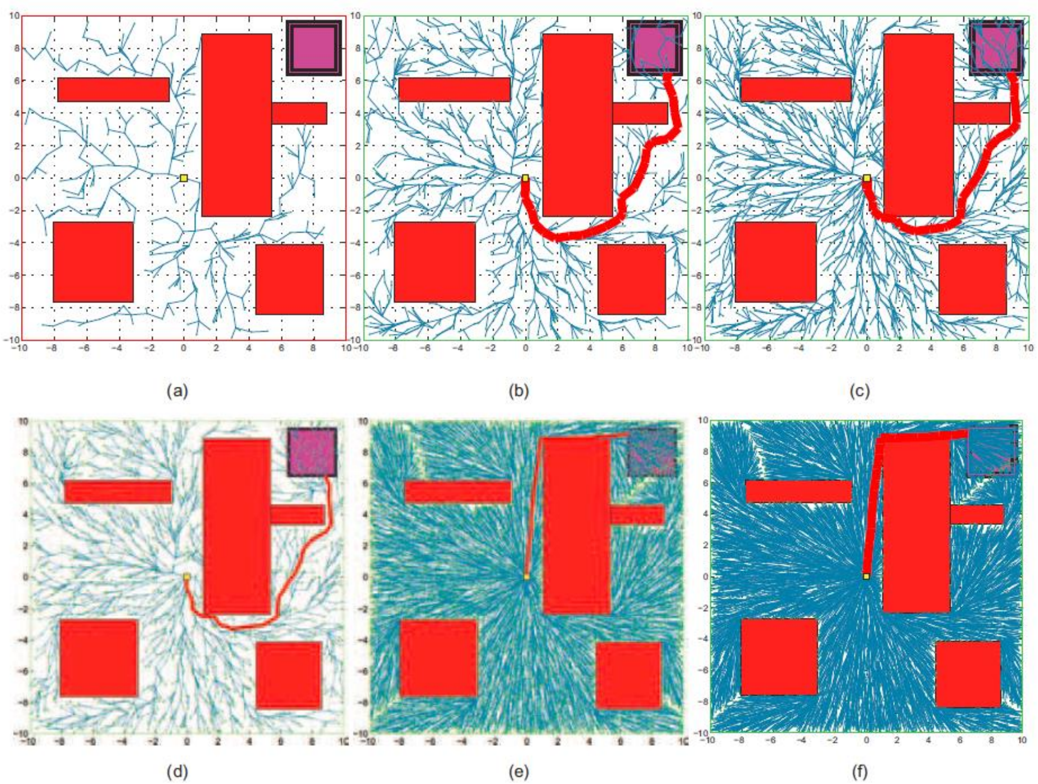
## Weak completeness

- **Resolution complete**: if no solution exists, the algorithm will run forever.
- **Probabilistically complete**: with infinite samples, the probability of finding an existing solution converges to one.

# Random Sampling Methods
## RRT*

- Incremental planning for an optimal solution




[Karaman2011]

# Random Sampling Methods
## Summary

- Generic motion planning methods
- Resolution and probabilistically complete
- Randomness of the results
- Inefficient with certain constraints: narrow passage
- Nearest neighbor search is time consuming

# Search Methods

## General Graph Search Algorithm

$N \leftarrow q_{initial}$;

$E \leftarrow \emptyset$;

While(1)

    $q \leftarrow$ select a node from N;

    $q_{new} \leftarrow$ create a new node from q to $q_{goal}$;

    $N \leftarrow N + \{q_{new}\}$:

    $E \leftarrow E + \{(q, q_{new})\}$;

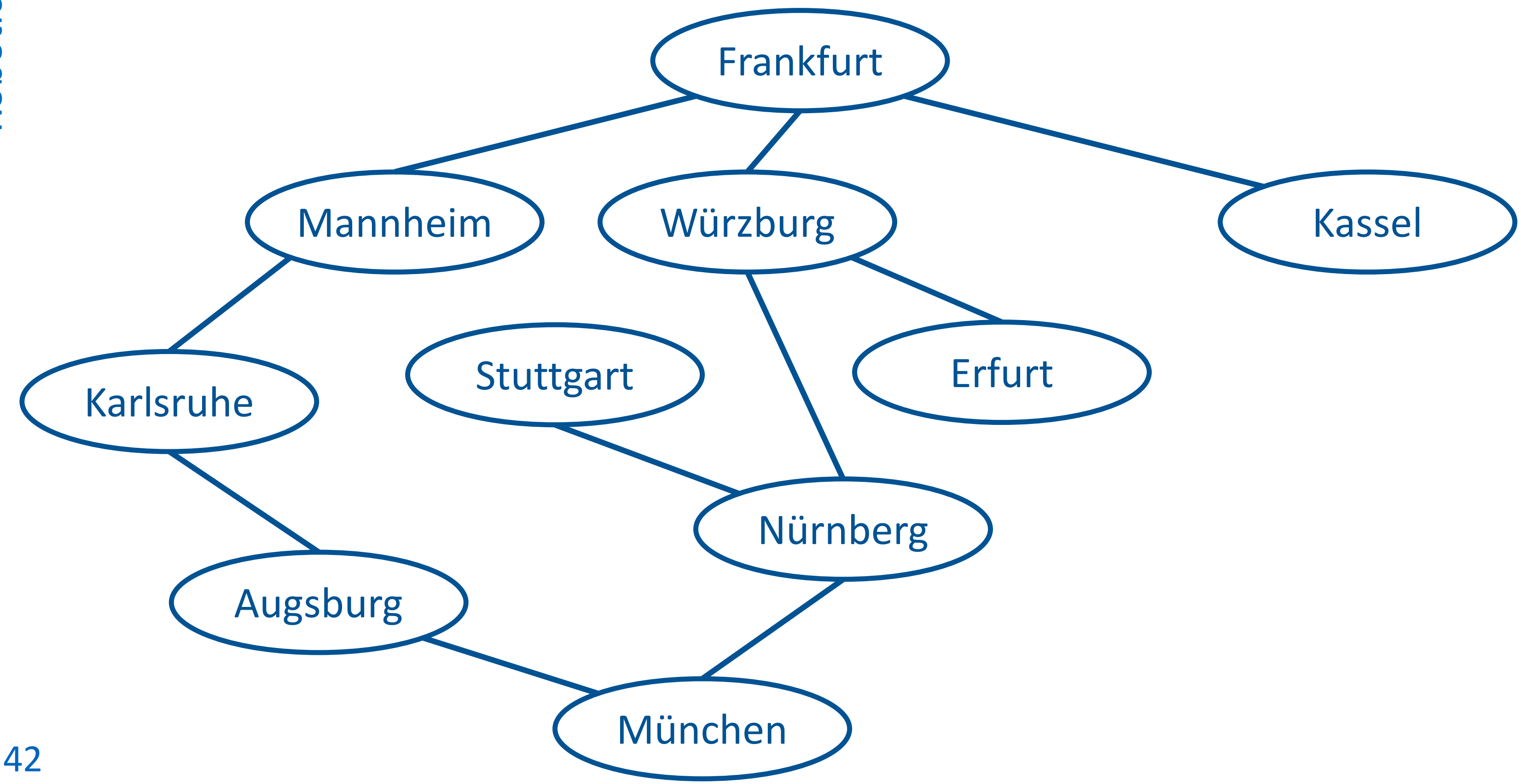
    if ($q_{goal} \in N$)

        return success;

# Search Methods

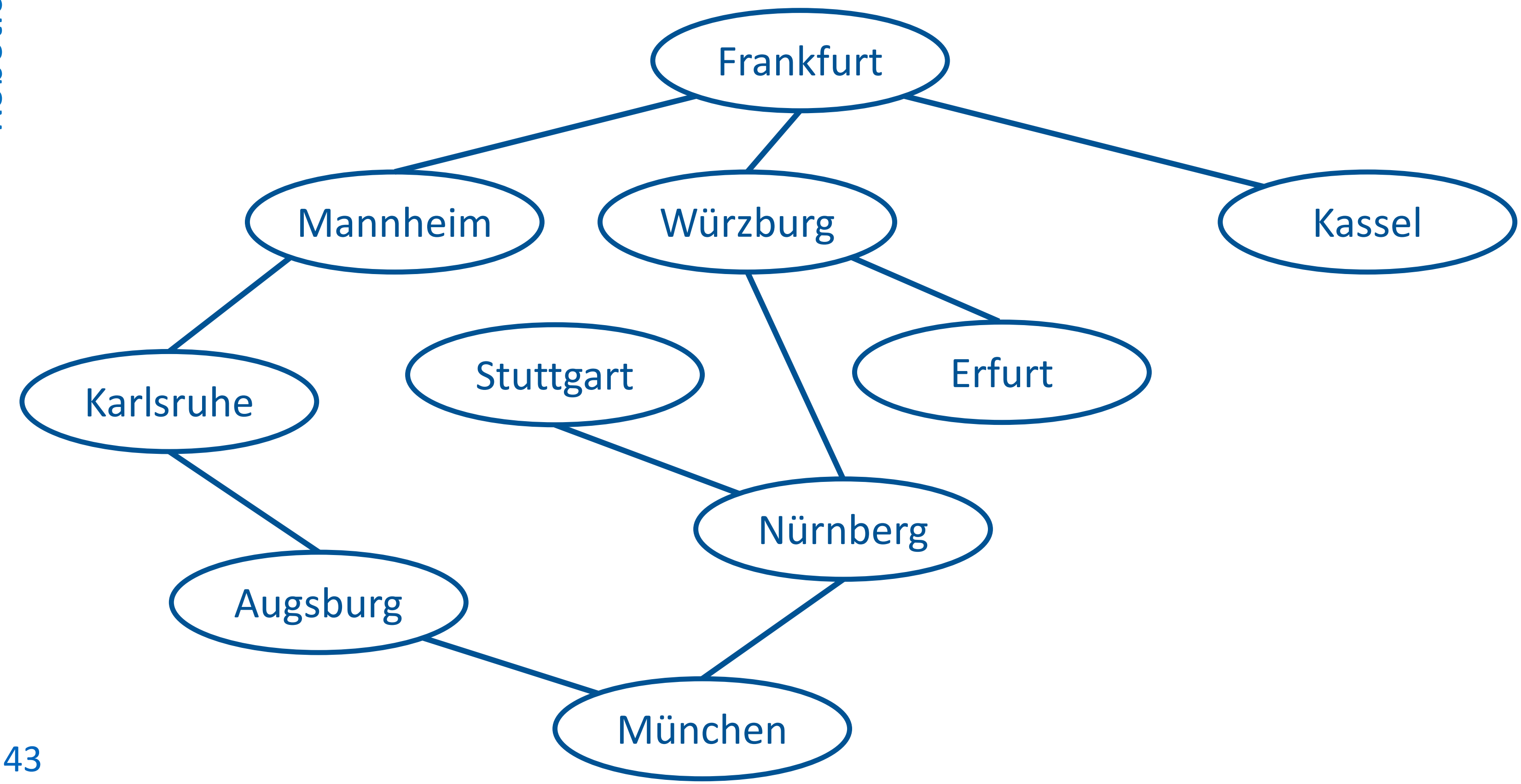
## Breadth-first Search

- Explore all the neighbors in one level, then continue with the next level



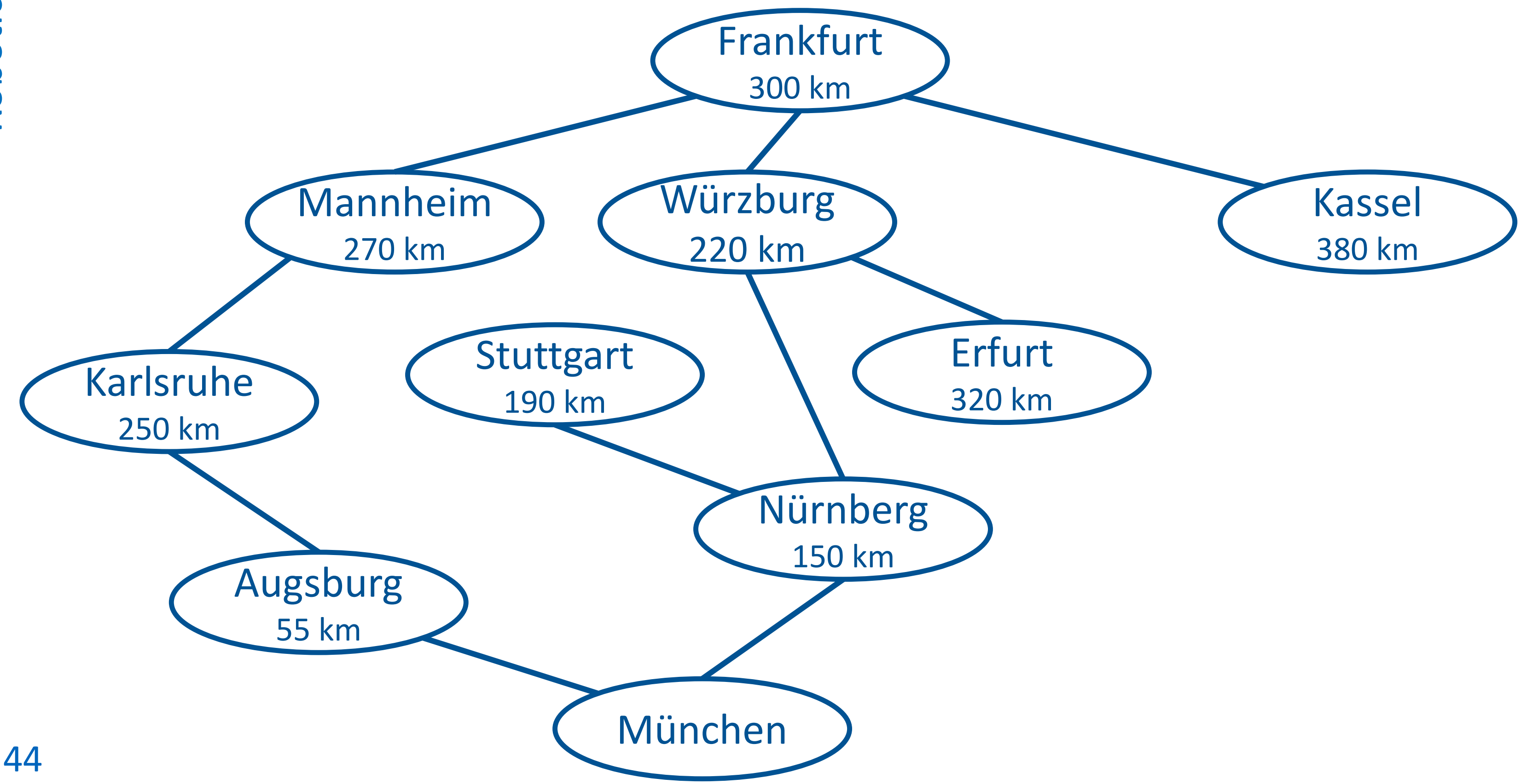
# Search Methods
## Depth-first Search

- Explore as far as possible along each branch



# Search Methods
## Best-first Search (Greedy Search)

- Explore the most promising node according to a specified rule



# Search Methods

## Heuristic Search (A* Search)

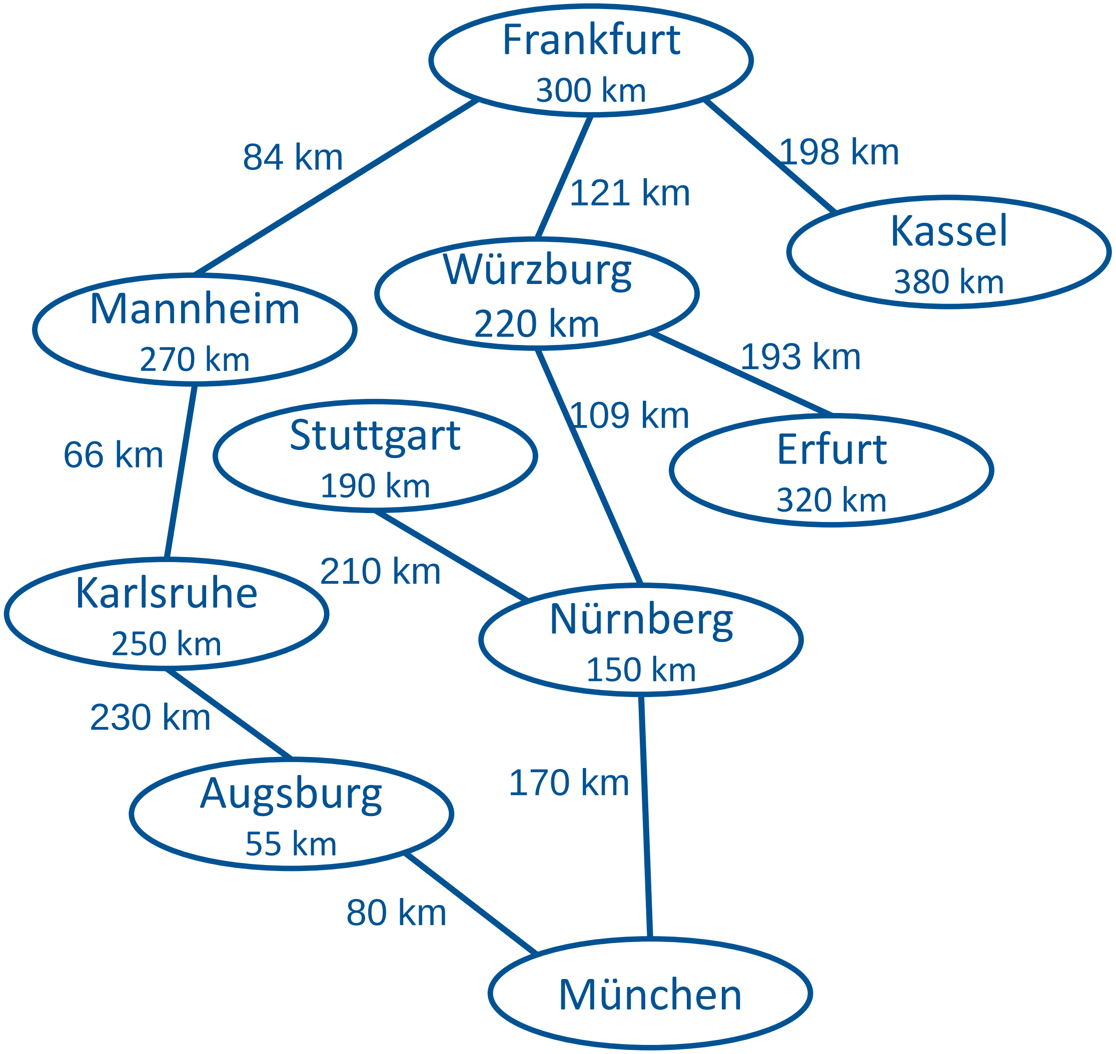
- Heuristic cost h: the estimated cost to reach the goal.
- Actual cost g: the sum of the edge weight from the start node.
- Total cost f: the sum of heuristic cost and actual cost.

$$f = h + g$$

- Explore the node with the minimum total cost.

# Search Methods
## Heuristic Search (A* Search)

- $f = h + g$
  - Frankfurt(300+0)
  - Würzburg(220+121)
  - Mannheim(270+84)
  - Nürnberg(150+230)
  - München (0+400)



# Search Methods

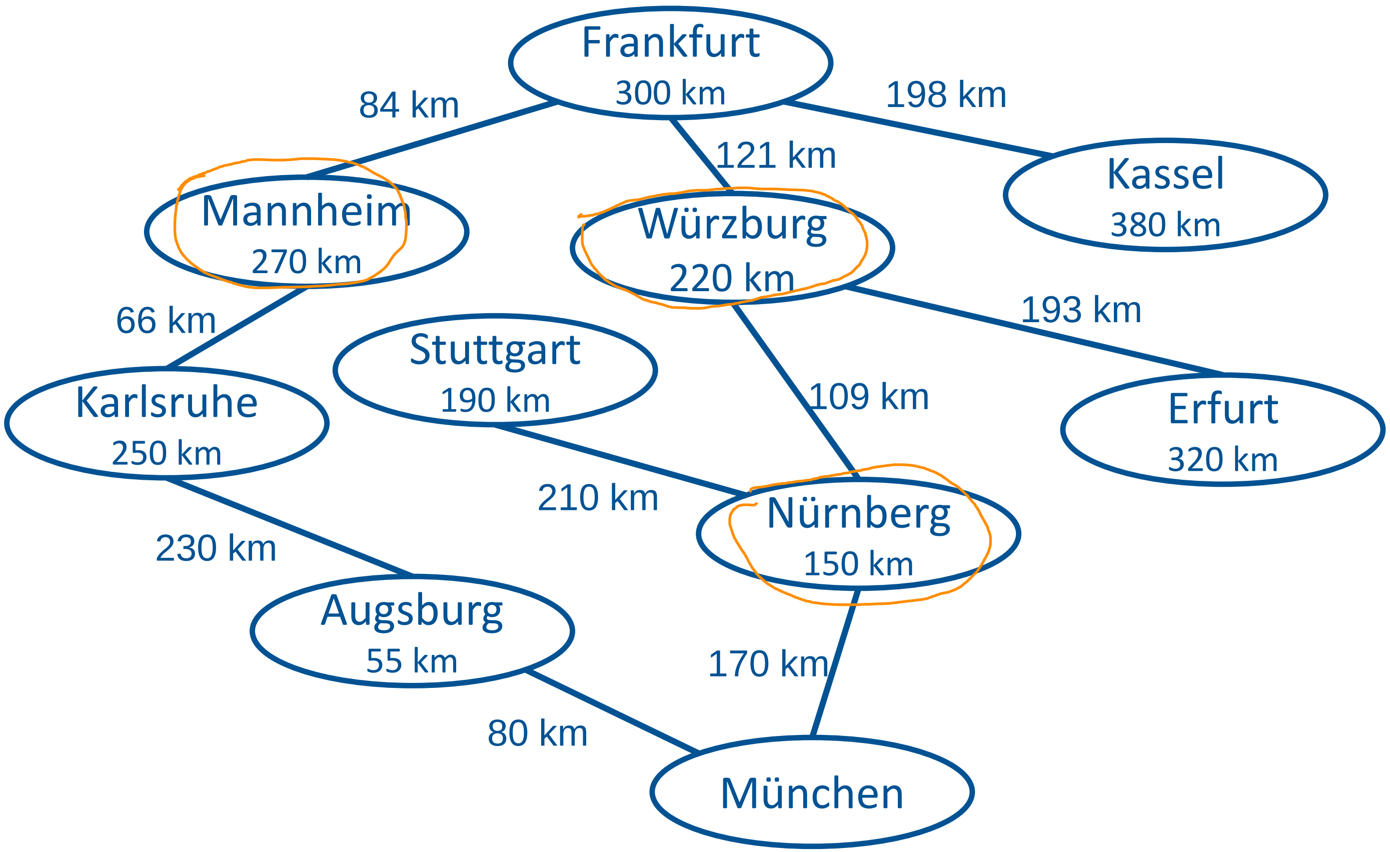
## Heuristic Search (A* Search)

- A* Search Algorithm
  - Node: q
    - Parent node
    - Actual cost
    - Total cost
  - Open set: $S_{open}$
    - Holds all the nodes waiting for expansion
    - Sorted after total costs
  - Closed set: $S_{closed}$
    - Holds all the visited nodes
  - Expand node x -> y
    - parent (y) = x
    - actual_cost(y) = actual_cost(x) + actual_cost(x, y)
    - total_cost(y) = actual_cost(y) + heuristic_estimate(y)
  - Construct path
    - Backtrack through parents

$S_{closed} \leftarrow \varnothing$;
$S_{open} \leftarrow q_{start}$;
While($S_{open} \neq \varnothing$)
    $q \leftarrow pop(S_{open})$;
    if ($q \in S_{closed}$)
        continue;
    if ($q == q_{goal}$)
        return path(q);
    $S_{closed} \leftarrow S_{closed} + \{q\}$;
    $\{q_i\} \leftarrow expand(q)$;
    $S_{open} \leftarrow S_{open} + \{q_i\}$;
return failed;

# Search Methods
## Heuristic Search (A* Search)

- **Open Set**: the frontier of the exploration. (Only border orange)
- **Closed Set**: the internal of the exploration area (Text orange).



# Search Methods
## Heuristic Search (A* Search)

- Admissible condition:
  - The heuristic estimation is always optimistic (not conservative).
  - heuristic_cost(q, goal) ≤ actual_cost(q, goal)
  - A* is optimally efficient when the heuristic is admissible, i.e., takes the minimum number of nodes to find the result.
- Monoton condition (consistent):
  - heuristic_cost(x) ≤ actual_cost(x, y) + heuristic_cost(y)
  - Each node only needs to be evaluated once.

# Search Methods

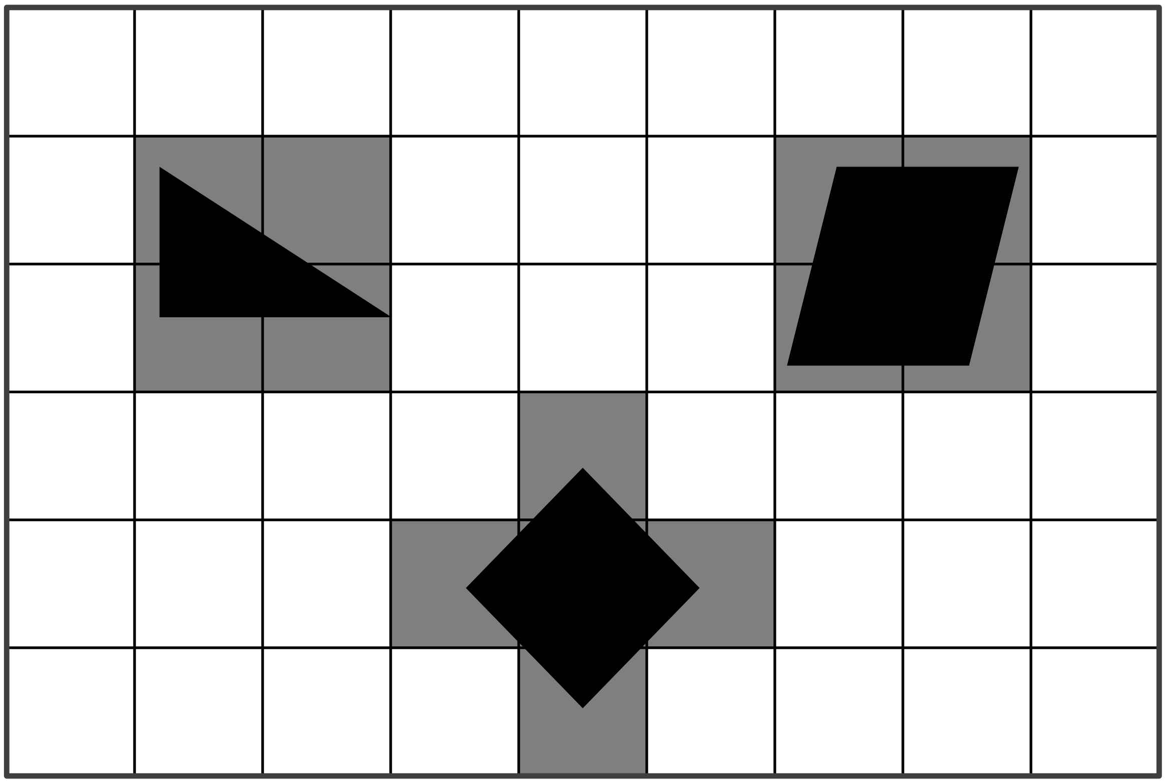
## Heuristic Search (A* Search)

- Finding the best heuristic is as difficult as the search problem itself.
- How to create a heuristic
  - Relax one or several constraints in the problem
  - e.g., ignore the obstacles, ignore the kinematics constraints
- How to combine multiple heuristics
  - Example: $h_1$, $h_2$
    - $h = \max(h_1, h_2)$

# Search Methods

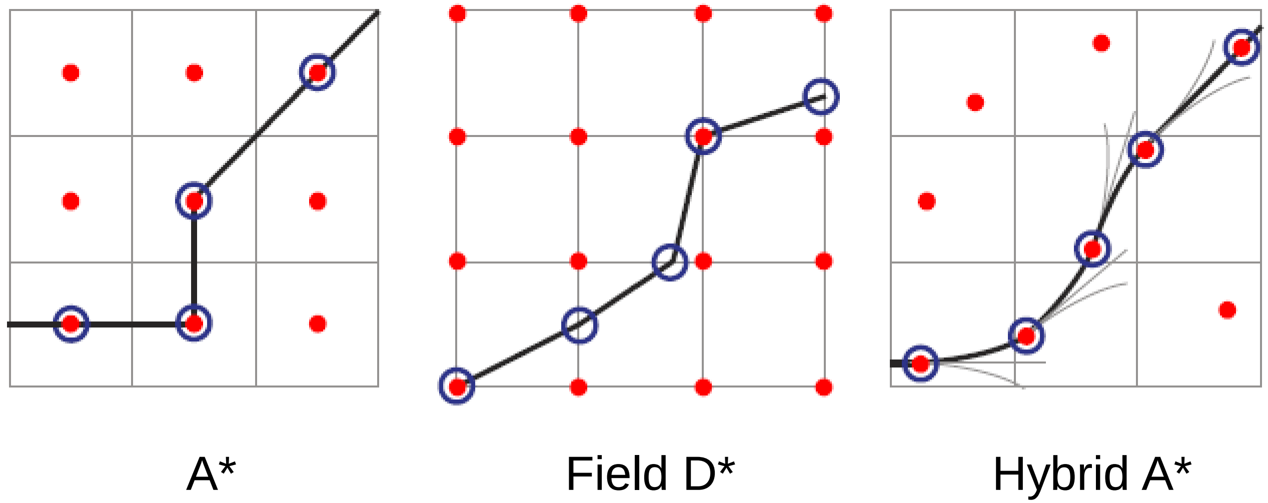
## Search in Continuous Configuration Space

- Infinite number of states
- Grid-based discretization



# Search Methods
## Search in Continuous Configuration Space

- Continuous motion
- Improve the grid discretization



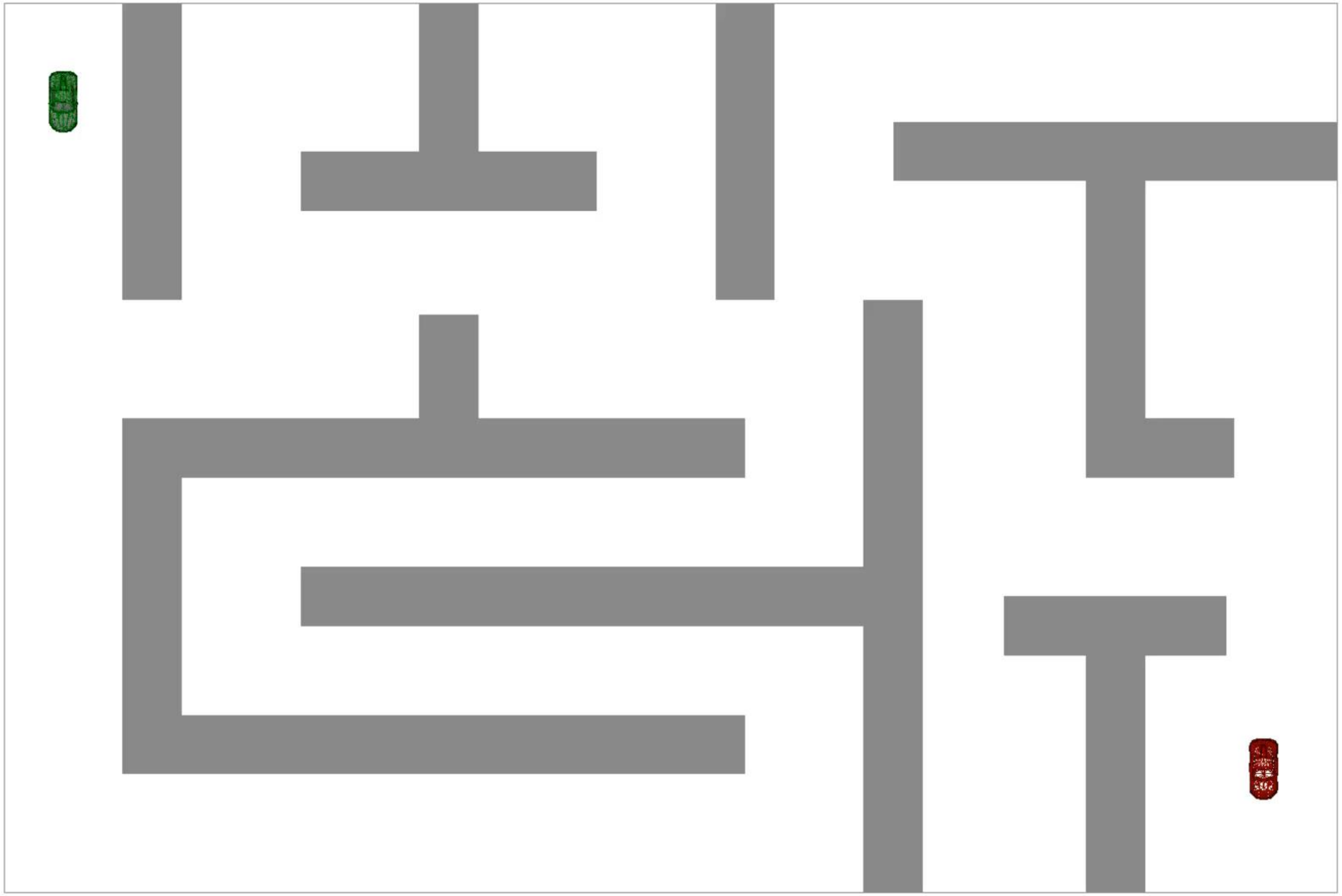

[Dolgov2010]

# Search Methods

## Hybrid A* Search

- Mapping a continuous configuration to a discrete grid
- Primitive motion: different combinations of control inputs
- Heuristics
  - Actual cost (eg. Reed Shepp)
  - h_1: Grid-based distance with obstacles
  - h_2: Nonholonomic distance without obstacles
  - $f = g + h$, $h = \max(h_1, h_2)$

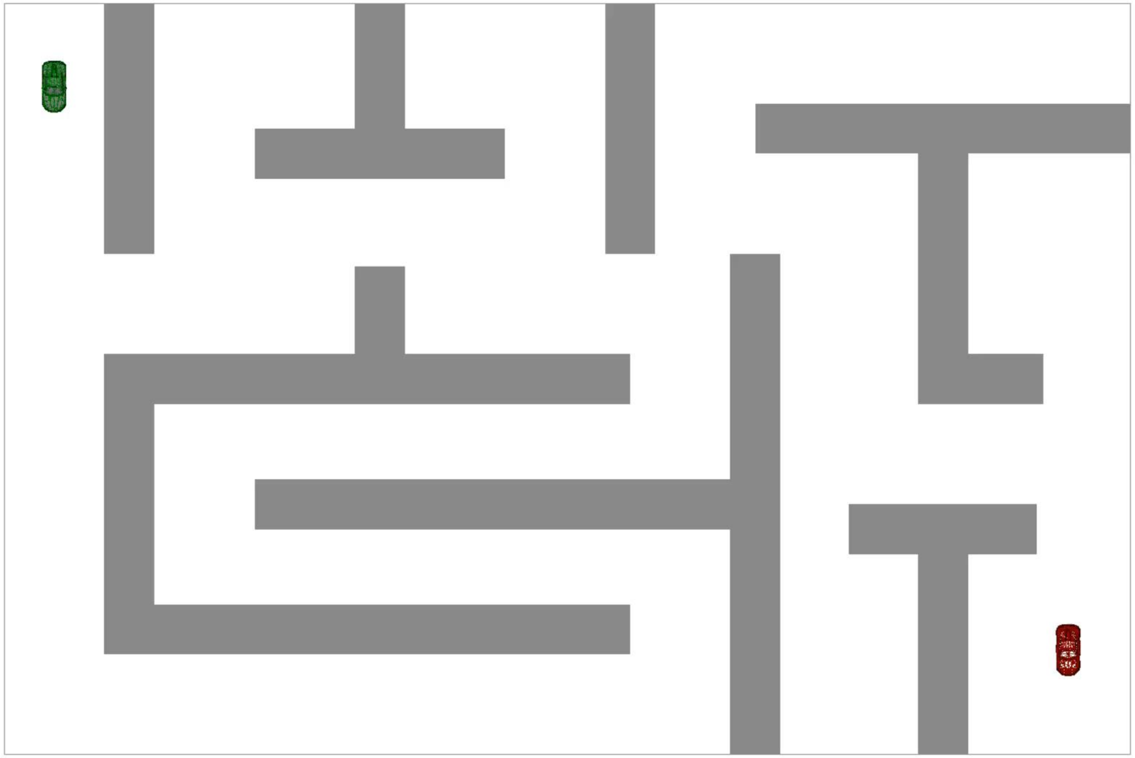
# Search Methods
## Hybrid A* Search



# Search Methods

## Heuristic Search with Space Exploration

- Heuristic from workspace exploration result
- Search step-size adaptation



[Chen2013]

# Search Methods
## Summary

- Heuristic search expand the node with the least total cost:
  - total cost = actual cost + heuristic cost
- Admissible heuristic:
  - heuristic_cost(x, goal) ≤ actual_cost(x, goal)
- Monoton heuristic:
  - heuristic_cost(y) ≤ heuristic_cost(x) + actual_cost(x, y)
- Continuous space:
  - Grid discretization
  - Primitive motions

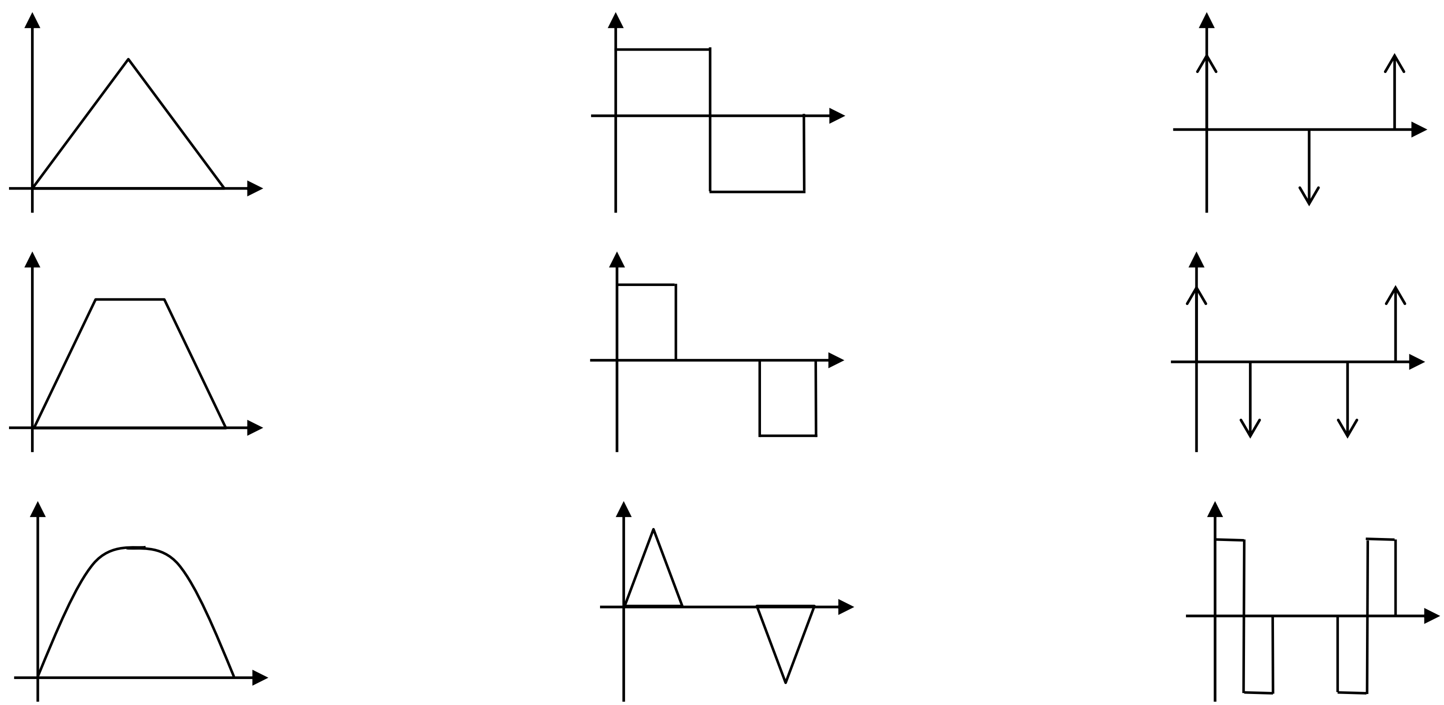
# Path Optimization

- Cost function
  - Efficiency: path length
  - Safety: distance to obstacles
  - Comfort: smoothness of the path
- Methods
  - Newton's method
  - Gradient descent

# Trajectory Generation

- Speed profile

Velocity $\dot{x}$ Acceleration $\ddot{x}$ Jerk $\dddot{x}$

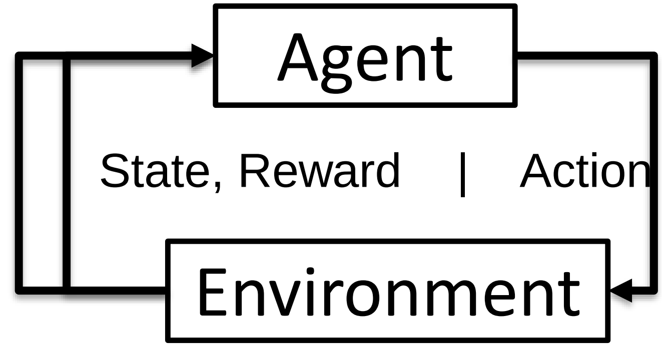


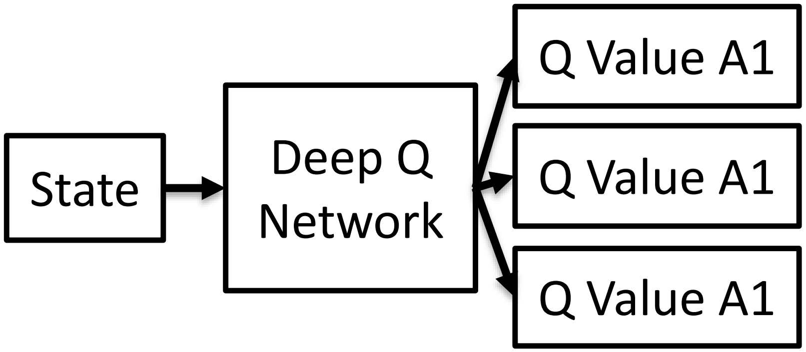
# Path Planning for Autonomous Driving

- Applications
  - Autonomous parking
  - Adaptive cruise control
  - Autonomous park house
- Further aspects
  - Traffic rules
  - Interaction with the traffic
  - Online motion planning

# Deep Q Learning

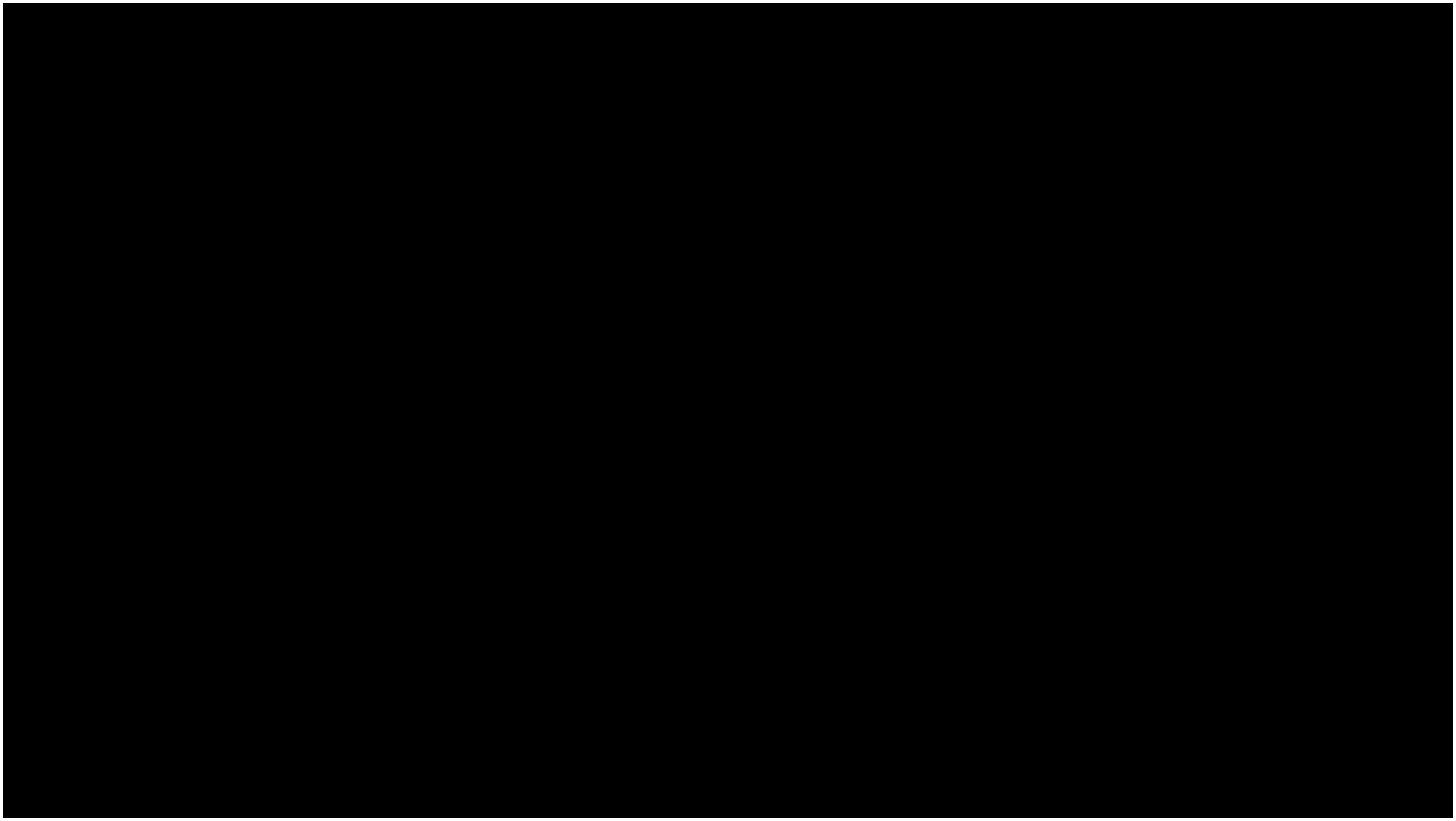
- Learn agent policy to provide best action to take depending on state
- Executed action provides according reward to support learning
- Neural Network approximates Q function




$$NewQ(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma\, max\, Q'(s',a') - Q(s,a)]$$

# Reinforcement Learning



# References


- [Laumond1998] Jean-Paul Laumond, Ed., *Robot Motion Planning and Control*. 1998.
- [LaValle2004] Steven LaValle, Ed., *Planning Algorithms*. 2004.
- [Khatib1986] Oussama Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,” *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, Spring 1986.
- [Barraquand1991] Jerome Barraquand and Jean-Claude Latombe, “Robot Motion Planning: A Distributed Representation Approach,” *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, Dec. 1991.
- [Quinlan1993] Sean Quinlan and Oussama Khatib, “Elastic Bands: Connecting Path Planning and Control,” presented at the IEEE International Conference on Robotics and Automation, 1993, pp. 802–807.
- [Brooks1991] Rodney Brooks, “Intelligence without Representation,” *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.
- [Kavraki1996] Lydia Kavraki, Petr Svcstka, Jean-Claude Latombe, and Mark Overmars, “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
- [LaValle1998] Steven LaValle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” Computer Science Dept., Iowa State University, Oct. 1998.
- [Karaman2011] Sertac Karaman and Emilio Frazzoli, “Sampling-Based Algorithms for Optimal Motion Planning,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
- [Dolgov2010] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel, “Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments,” *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, Apr. 2010.
- [Chen2013] Chao Chen, Markus Rickert, and Alois Knoll, “Combining Space Exploration and Heuristic Search in Online Motion Planning for Nonholonomic Vehicles,” presented at the IEEE Intelligent Vehicles Symposium, 2013, pp. 1307–1312.

**Gereon Hinz**

**STTech GmbH**

Floriansmühlstraße 8 - 80939 München

**Tel**: 089/905499430

gereon.hinz@sttech.de

www.sttech.de