Institute of Automotive Technology
Faculty for Mechanical Engineering
Technical University of Munich

Autonomous Driving Software Engineering

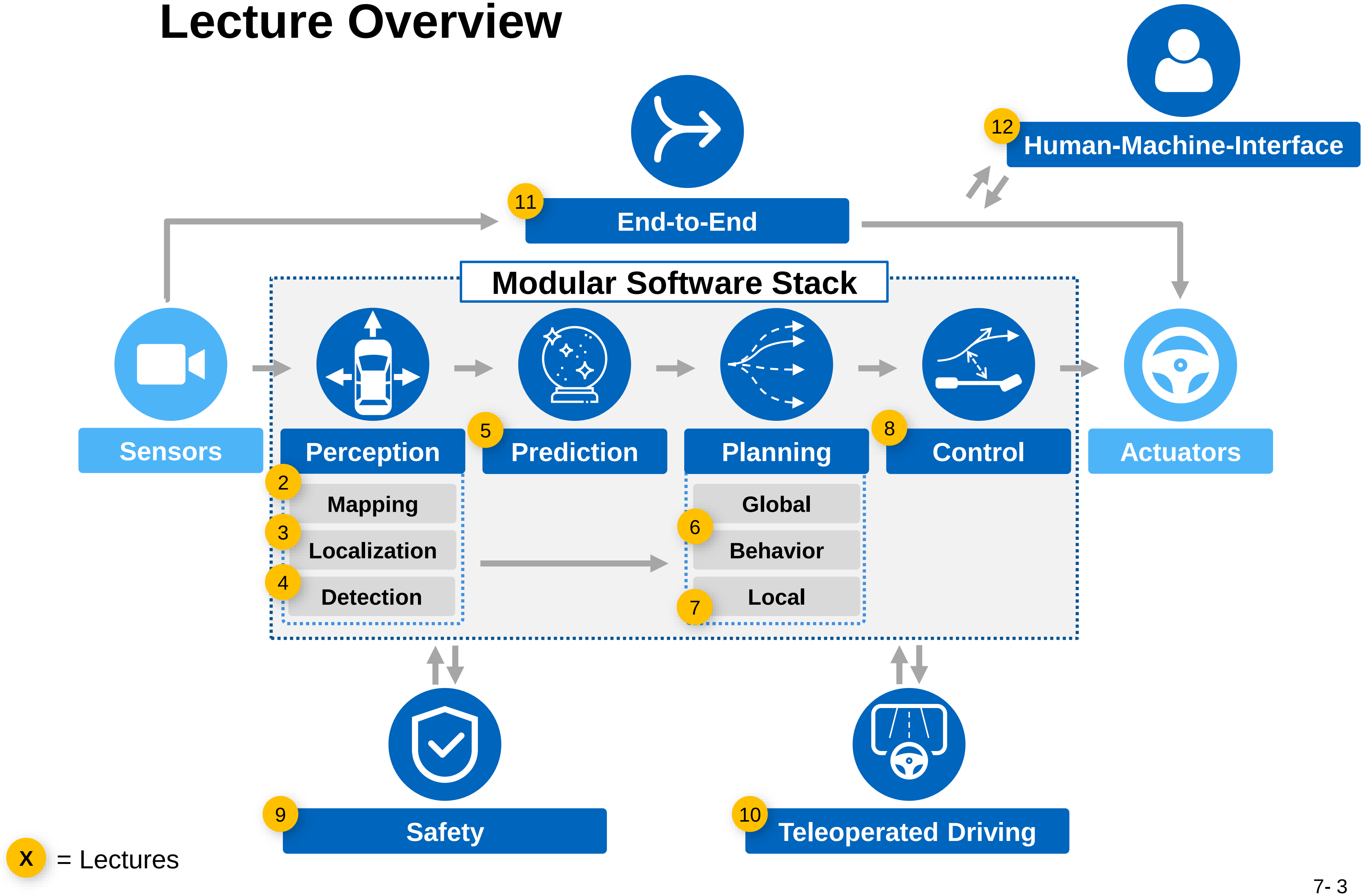
Prof. Dr.-Ing. Markus Lienkamp

Phillip Karle, M. Sc.

Lecture Overview

Lecture – 90min	Practice – 45min
1 Introduction: Autonomous Driving Karle	**1 Practice** Karle
2 Perception I: Mapping Sauerbeck	**2 Practice** Sauerbeck
3 Perception II: Localization Sauerbeck	**3 Practice** Sauerbeck
4 Perception III: Detection Huch	**4 Practice** Huch
5 Prediction Karle	**5 Practice** Karle
6 Planning I: Global Planning Trauth	**6 Practice** Trauth
7 Planning II: Local Planning Ögretmen	**7 Practice** Ögretmen
8 Control Wischnewski	**8 Practice** Wischnewski
9 Safety Assessment Stahl	**9 Practice** Stahl
10 Teleoperated Driving Feiler	**10 Practice** Feiler
11 End-to-End Betz	**11 Practice** Betz
12 From Driver to Passenger Fank	**12 Practice** Karle

Lecture Overview



Objectives for Lecture 7: Local Planning

Depth of understanding

After the lecture you are able to…

	Remember	Understand	Apply	Analyze	Evaluate	Develop
… explain the need of local planning	■	■				
…. name the requirements on local planning	■					
… compare different methodologies	■	■				
… implement and use graph-based planning methods	■	■	■			
… explain methods for collision checking	■	■				

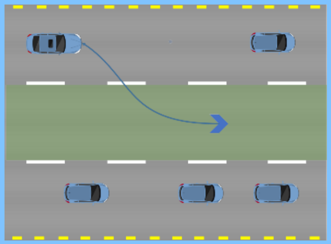
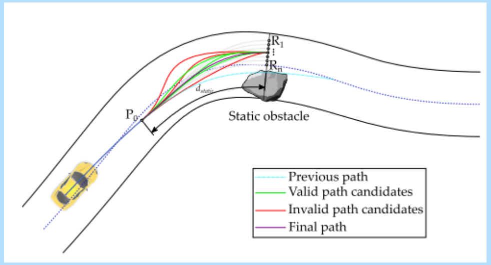
Local Planning
Prof. Dr. Markus Lienkamp

Levent Ögretmen, M. Sc.

Agenda

1. **Introduction**
2. Local Planning Methodologies
3. Deep-Dive Graph-Based Planning
4. Collision Checking
5. Summary

Introduction
Definition of terms

Global Planning	Uses map for planning without information of local environment. <u>Focus: hours to minutes.</u>	
Behavior Planning	High-level description of the vehicle motion. <u>Focus: minutes to seconds.</u>	
Local Planning	Consideration of local objects. Deals with reactive decisions. <u>Focus: seconds to milliseconds.</u>	 R1 Rn P0 Static obstacle Previous path Valid path candidates Invalid path candidates Final path

Last Lecture

This Lecture

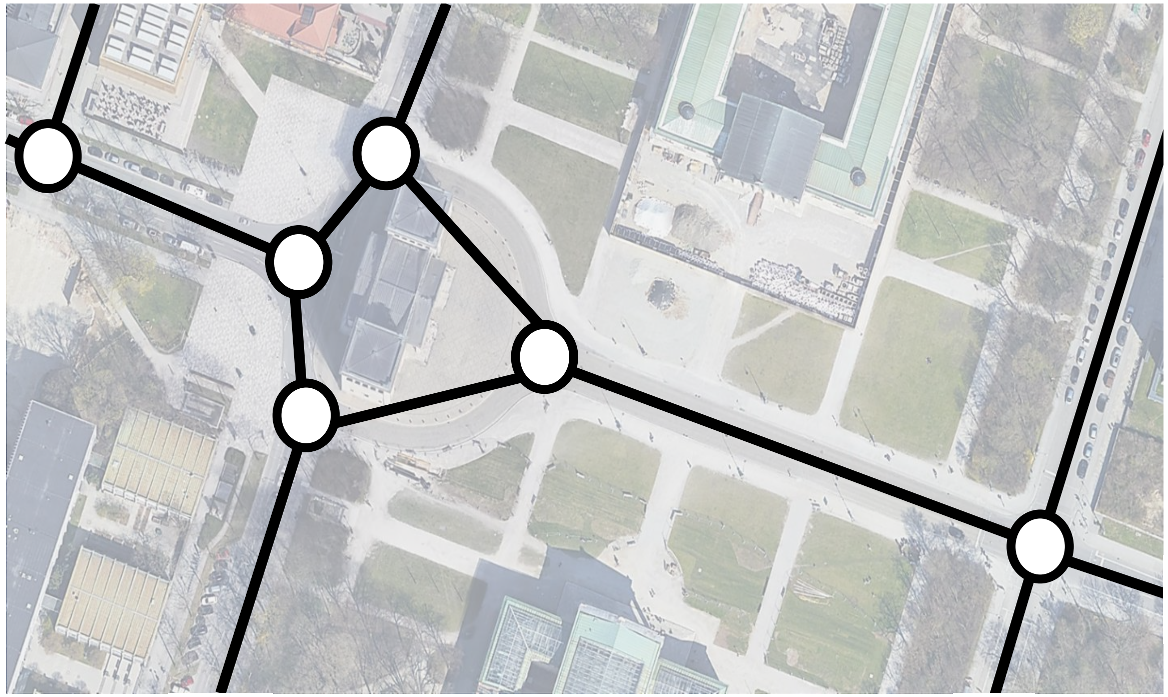
Introduction – Global to Local Planning
Differentiation

Global Planning	Local Planning
Map-based	Sensor-based (reactive planning)
Relatively slower response	Fast response
Known workspace	Suppose incomplete workspace
Generate path/route before moving	Planning and moving at the same time
No strict requirements on calculation time	Requirements on calculation time

Introduction - Path- and Trajectory Planning
Differentiation

Route Planning:

- Sequence of discrete geometrical nodes in map network

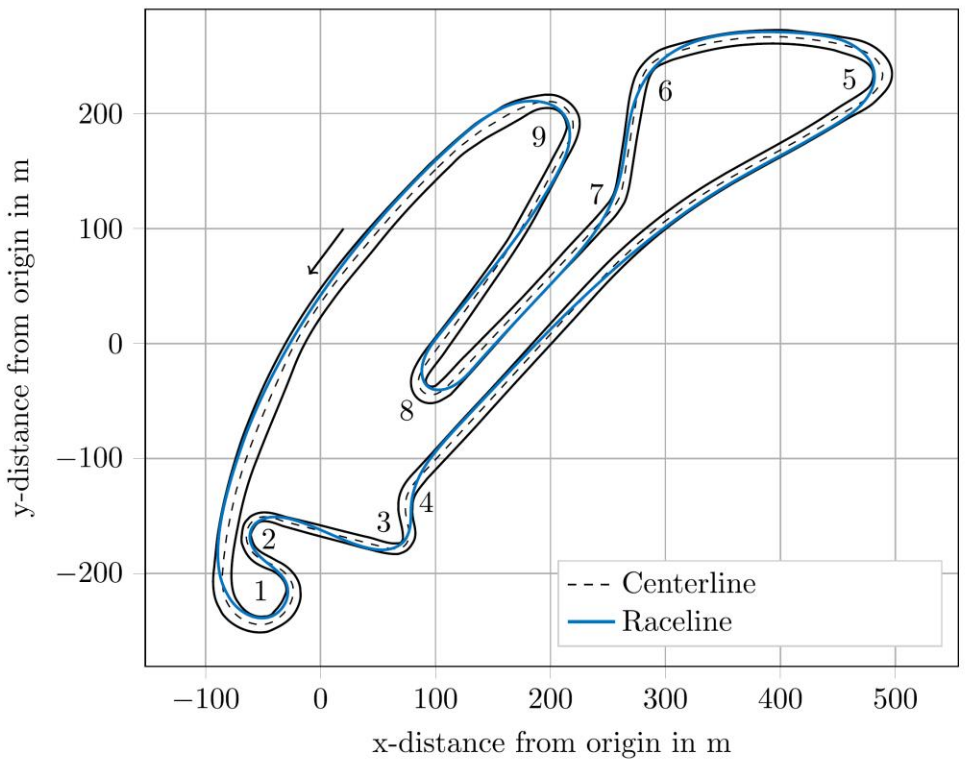


Source: Google Earth

Introduction - Path- and Trajectory Planning
Differentiation

Path Planning:

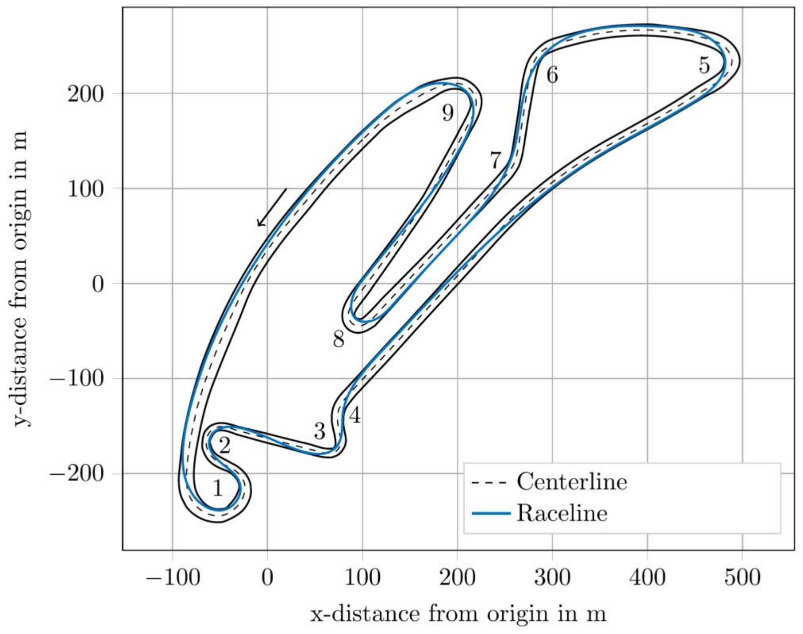
- Continuous curve in <u>spatial</u> domain



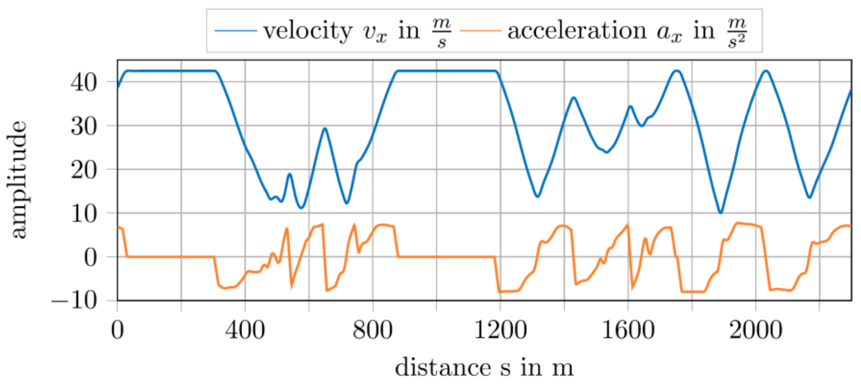
Introduction - Path- and Trajectory Planning
Differentiation

Trajectory Planning:

- Continuous curve in <u>spatio-temporal</u> domain



+



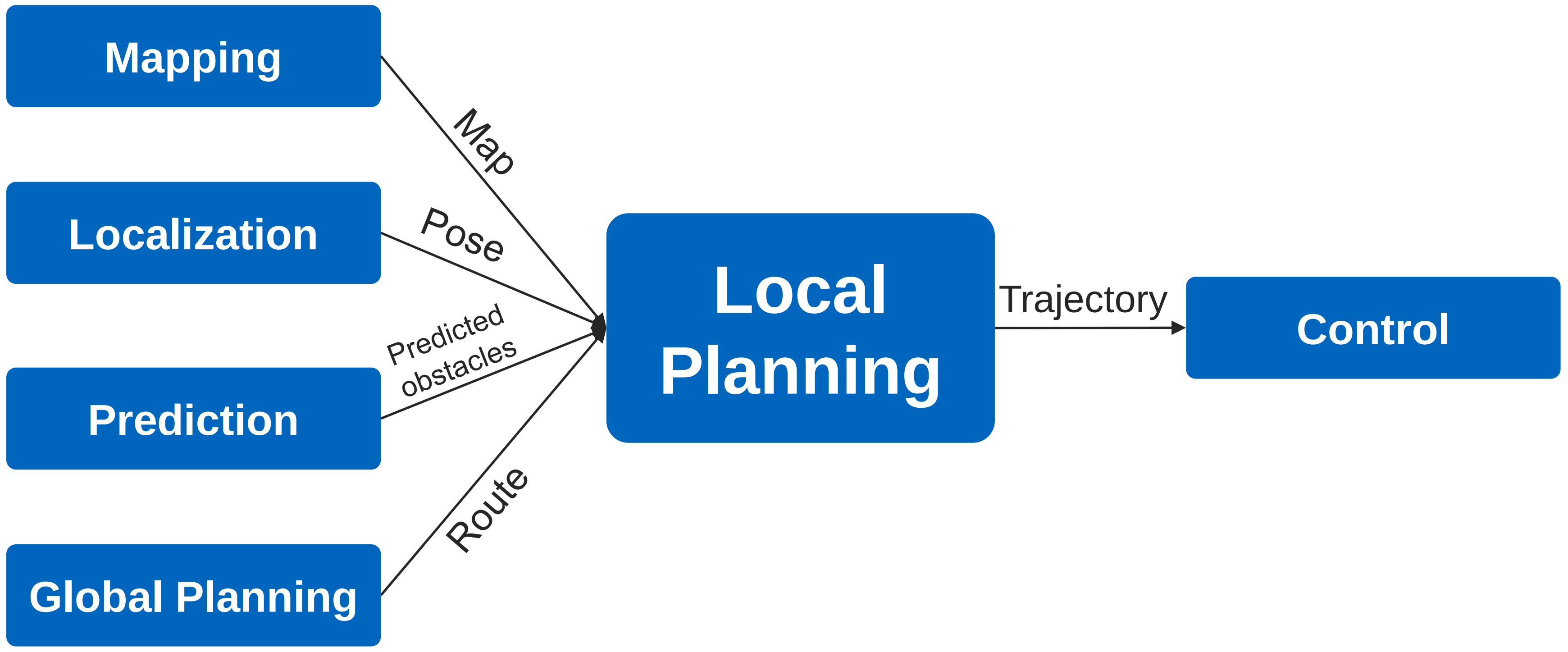
Additional Slides

The last lecture introduced methods to find the optimal route in a road network modelled as a graph from the starting point to the goal. This optimization provides only a coarse path with only spatial information. This means we do not know yet when the vehicle is supposed to be where. Furthermore, the presented search algorithms mainly use offline available information like a road map and do not consider current local conditions including other traffic participants in close proximity, blocked lanes due to constructions or the status of a traffic light.

From this, the need for a local trajectory planner becomes apparent. Besides the road network the local planner additionally has all information available that are generated from the sensors and determine the current scenario in which the vehicle is operating. The main task of the local planner is to generate a **feasible and collision free trajectory (time dependent path)** that leads the vehicle in the current **local** environment and **follow the global planning target as good as possible** while **obeying the traffic rules**.

This lecture provides an overview of requirements on a local trajectory planner and methodologies to generate a trajectory that can be tracked by a controller.

Interfaces


Mapping
Localization
Prediction
Global Planning
Map
Pose
Predicted obstacles
Route
Local Planning
Trajectory
Control

Additional Slides

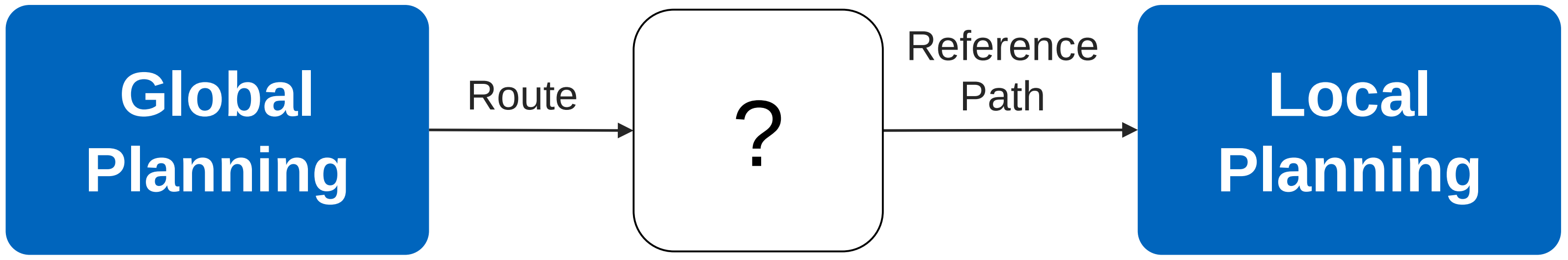
Like the global planner, the local planning requires a map to obtain a reference path and a driving corridor. Before planning a trajectory, the vehicle must be located on the map. The information of the current position, orientation and velocity is provided by the localization module.

As a reference, the local planner receives a global target by the global planning. This can be a sequence of lanelets or another representation that guides through a road network.

If there were no other traffic participants or obstacles on the road the only task of the local planning would be to generate a feasible velocity profile along the sequence of reference paths provided by the global planning. However, driving on public roads is characterized by dynamic environments. Therefore, the local planner needs a recent representation of the local environment. To plan a time dependent path, it is not enough to know the current positions of other traffic participants but also the predicted movement.

With these inputs a local trajectory with a certain time horizon can be generated. The trajectory is sent to the controller which tracks it as good as possible.

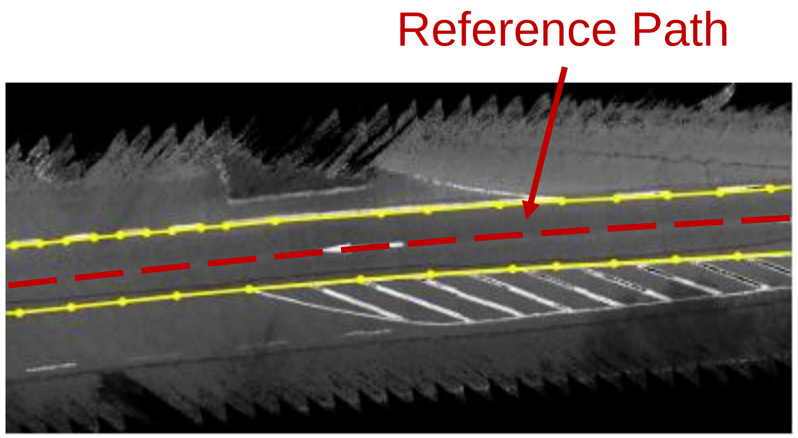
Interface between Global and Local Planning

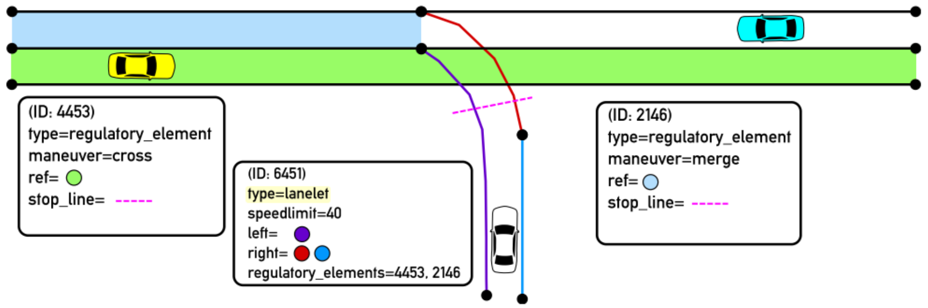


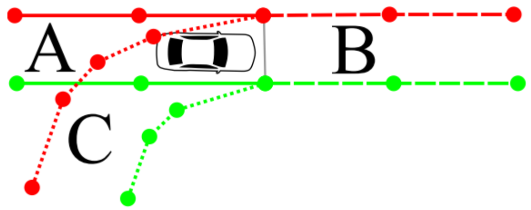
How to extract a reference path from a route?

Lanelet network as map representation

- Lanelet: atomic lane segment
- Lanelet network represents map for structured environments
- Provide a reference path and driving corridor
- Contain regulatory elements





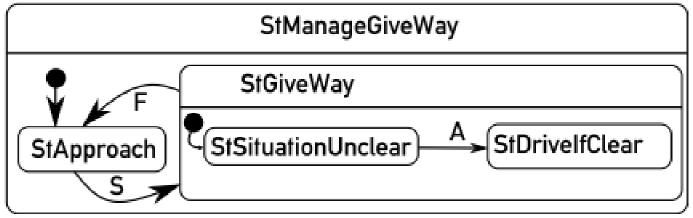


Bender, Ziegler, Stiller - 2014 - Lanelets Efficient map representation for autonomous driving

Additional Slides

A lanelet is an atomic lane segment which is characterized by its left and right bound. Concatenating lanelets results in a lanelet network. This lanelet network is a graph with a well defined adjacency of lanelets. The global route planning provides a sequence of drivable lanelets. The centerlines of these lanelets can be used as reference paths and the boundaries provide a driving corridor for the autonomous vehicle.

Besides geometric information a lanelet contains regulatory elements like traffic signs and rules and relations between lanelets like cross or merge. These regulatory elements can be utilized by a behavior state machine to trigger state transitions. The states in turn provide constraints like top speed for the detailed local trajectory planning.

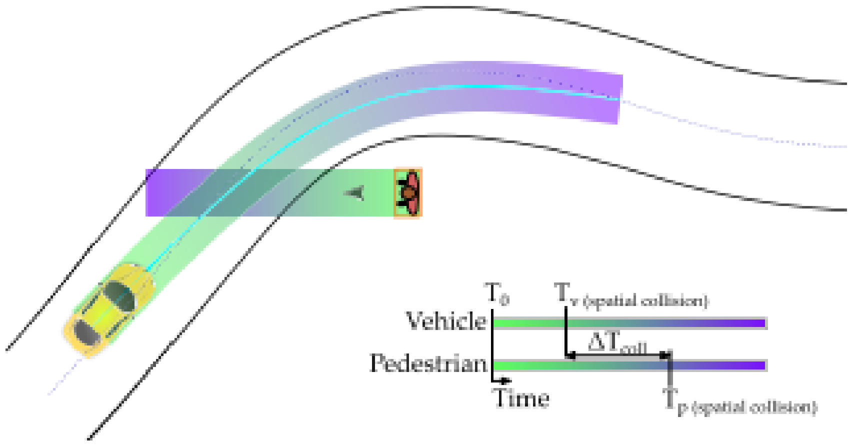


Bender, Ziegler, Stiller - 2014 - Lanelets Efficient map representation for autonomous driving

Requirements

… for the planner

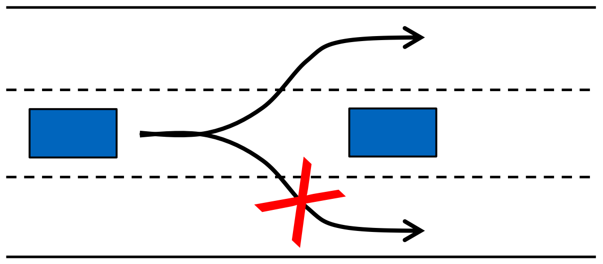
Operate in a dynamic environment



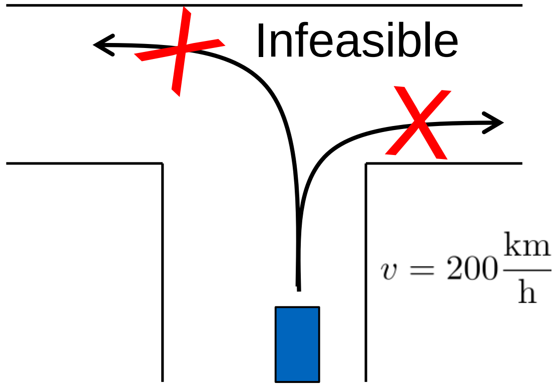
T_0
$T_{v\ (\text{spatial collision})}$
Vehicle
ΔT_{coll}
Pedestrian
Time
$T_{p\ (\text{spatial collision})}$

Antonio Artunedo, Jorge Villagra, and Jorge Godoy. "Real-Time Motion Planning Approach for Automated Driving in Urban Environments". In: IEEE Access 7 (2019)

Consider the traffic rules



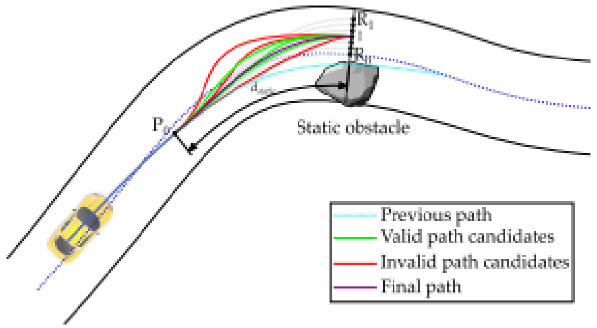
Recursive feasibility



Requirements

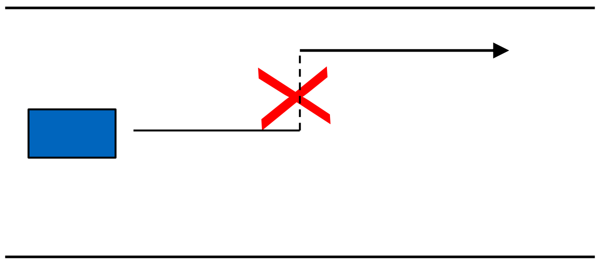
… for the generated trajectory

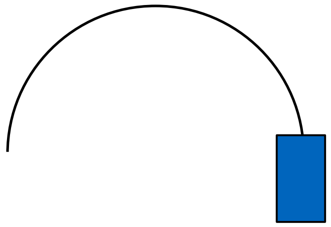
Collision-free



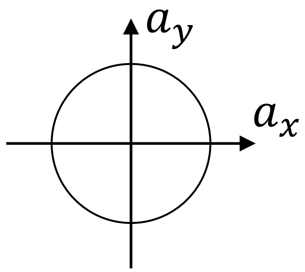
Antonio Artunedo, Jorge Villagra, and Jorge Godoy. “Real-Time Motion Planning Approach for Automated Driving in Urban Environments”. In: IEEE Access 7 (2019)

Continuity

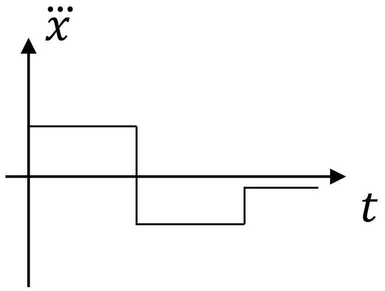




Kinematically feasible



Dynamically feasible



Comfort

Additional Slides

The local planner and the generated trajectory need to fulfill certain requirements.

Besides the obligatory requirement of obeying traffic rules, the planner must cope with dynamically changing environments including dynamic obstacles and e.g. switching of a traffic light. Therefor, it must be real-time capable to react on changes accordingly and ensure a safe operation.

The planner must operate in a recursive feasible manner. This means a generated trajectory must not lead to a situation in which no further feasible and collision free trajectory can be found. One measure to obtain recursive feasibility, especially at high speeds, is to use a long planning horizon.

Besides the requirements on the planner there are requirements the generated trajectory must meet.

In order to ensure a safe operation, the generated trajectory must be collision-free and feasible. A feasible trajectory comprises both the kinematic constraints and the dynamic constraints. While the kinematic constraints like the minimum turning radius mainly come into effect at low speeds the dynamic constraints limit the trajectories at higher speeds. The tires and the engine limits must be considered when both the longitudinal and lateral movement of the vehicle is planned.

State of the art controller concept that should track the planned trajectory furthermore require continuity regarding the path, velocity and the curvature. Continuity is also beneficial for the comfort that the passenger is experiencing. Jumps in the curvature profile induce peaks in jerk which results in uncomfortable movements for the passengers.

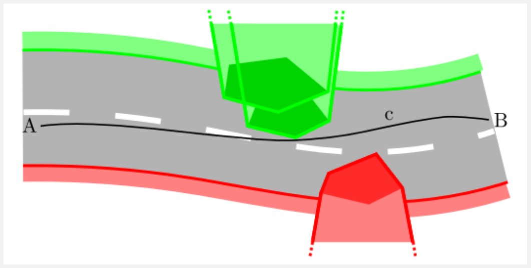
Local Planning
Prof. Dr. Markus Lienkamp

Levent Ögretmen, M. Sc.

Agenda

Planning Methodologies

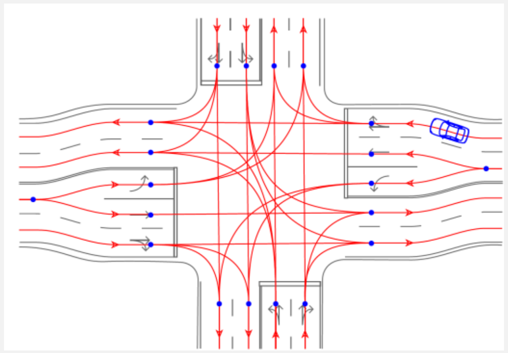
Variational Methods



$$\min J(\boldsymbol{x}(t))$$

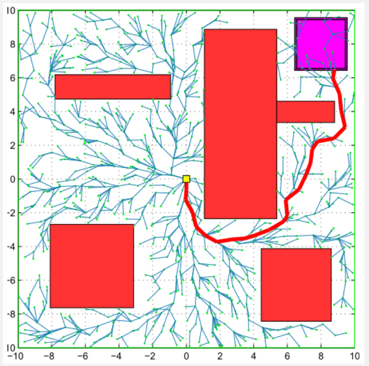
$$J(\boldsymbol{x}(t)) = \int_{t_0}^{t_0+T} L(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \dddot{\boldsymbol{x}})dt$$

Graph-Based Methods



G=(V, E)

Incremental Methods

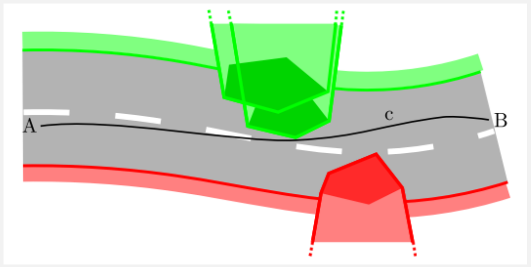


```
$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $i \leftarrow 0$;
while $i < N$ do
    $G \leftarrow (V, E)$;
    $x_{\text{rand}} \leftarrow$ Sample($i$); $i \leftarrow i + 1$;
    $(V, E) \leftarrow$ Extend($G, x_{\text{rand}}$);
```

Hybrid Methods

Planning Methodologies

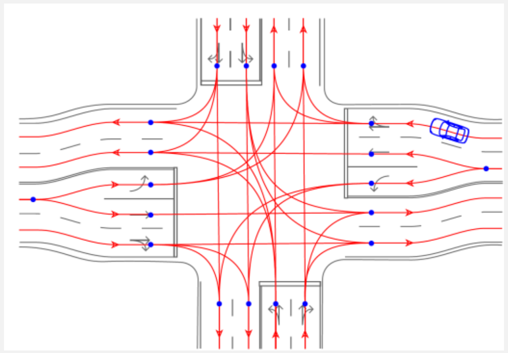
Variational Methods



$$\min J(\boldsymbol{x}(t))$$

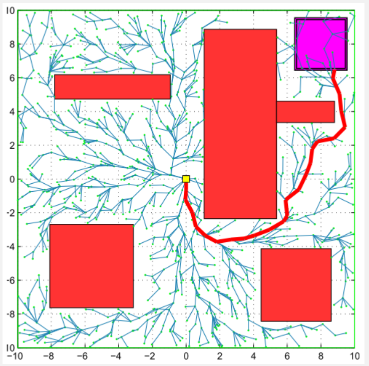
$$J(\boldsymbol{x}(t)) = \int_{t_0}^{t_0+T} L(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \dddot{\boldsymbol{x}})dt$$

Graph-Based Methods



G=(V, E)

Incremental Methods



```
$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $i \leftarrow 0$;
while $i < N$ do
    $G \leftarrow (V, E)$;
    $x_{\text{rand}} \leftarrow$ Sample($i$); $i \leftarrow i + 1$;
    $(V, E) \leftarrow$ Extend($G, x_{\text{rand}}$);
```

Hybrid Methods

Variational Methods

Cost functional

$$\text{minimize} \quad J(\boldsymbol{x}(t)) = \int_{t_0}^{t_0+T} L(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \dddot{\boldsymbol{x}})\, dt$$

Constraints

$$\begin{aligned} \text{subject to} \quad & \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \quad \text{and} \quad \boldsymbol{x}(t_0+T) \in X_{\text{goal}} \\ & f(\boldsymbol{x}, \dot{\boldsymbol{x}}, \dots) = 0 \quad \forall t \in [t_0, t_0+T] \\ & g(\boldsymbol{x}, \dot{\boldsymbol{x}}, \dots) \leq 0 \quad \forall t \in [t_0, t_0+T] \end{aligned}$$

Internal $\left\{ \begin{array}{l} \text{Kinematic} \\ \text{Dynamic} \end{array} \right.$

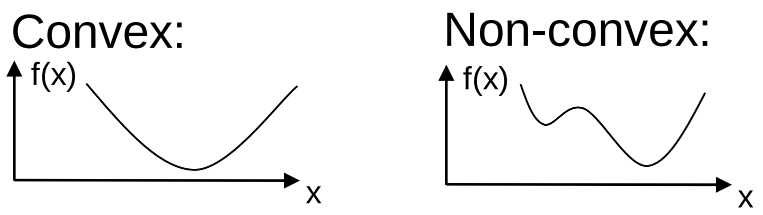
External $\left\{ \begin{array}{l} \text{Driving corridor} \\ \text{Obstacles} \end{array} \right.$

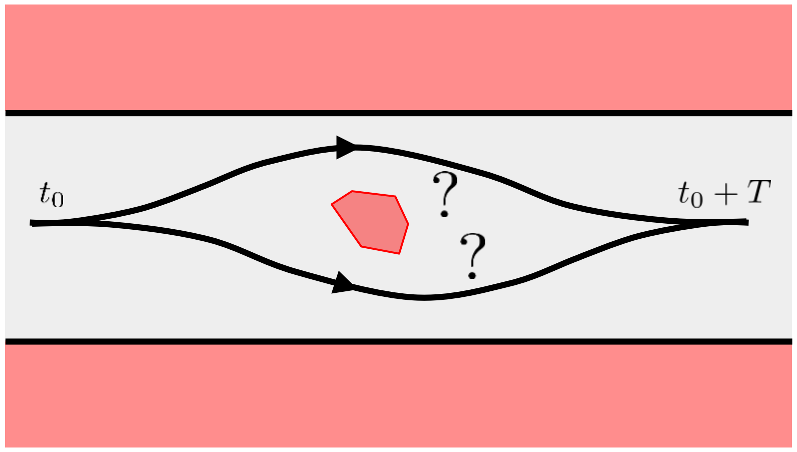
Example:

$$L = \underbrace{w_{\text{off}} \left| \boldsymbol{x} - \boldsymbol{x}_{\text{ref}} \right|^2}_{\text{Follow the reference path}} + \underbrace{w_{\text{vel}} \left| \dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\text{ref}} \right|^2}_{\text{Reach the desired velocity}} + \underbrace{w_{\text{acc}} \left| \ddot{\boldsymbol{x}} \right|^2}_{\text{Reduce accelerations}} + \underbrace{w_{\text{jerk}} \left| \dddot{\boldsymbol{x}} \right|^2}_{\text{Reduce jerk}}$$

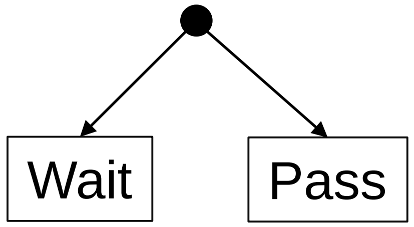
Variational Methods
Limitations

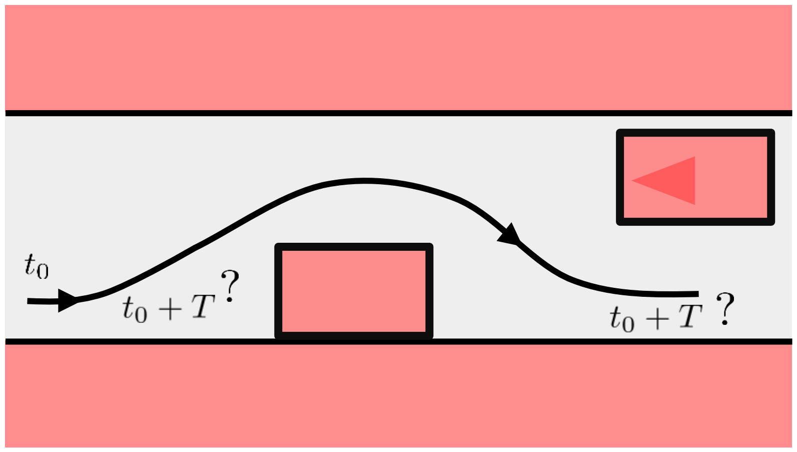
Global solutions





Combinatorial aspects





Additional Slides

The basis of variational methods is a cost functional. This usually is an integral over the cost function from the current time up to the planning time horizon. The cost function includes terms that usually work against each other. E.g. terms that penalize a deviation of the reference path or the target velocity act against terms that should induce smoothness of the trajectory and penalize high accelerations or jerks.

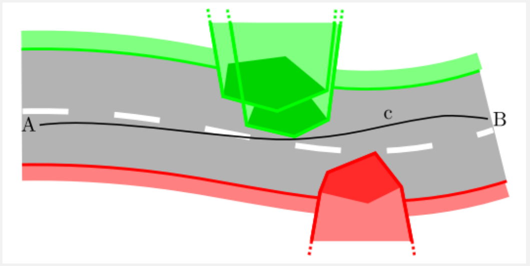
Such optimization problems are solved using nonlinear optimization methods. Given an initial guess of the solution current solvers can find a local minimum with only a few iterations. The planning problem however in general is a non-convex problem where the local optimum is not necessarily the global optimum. The simple example whether to perform an evasive maneuver on the left or right is a problem where the feasible region and therefore the constraints are non-convex.

There are solvers available than can find the global optimum which however require exponential time in the number of variables and constraints.

To maintain the advantage of fast computations local methods require a prior decision to find the correct local solution to the problem. This decision can be derived from traffic rules e.g. “only overtake left”, “give way” or a superior behavioral planner that considers combinatorial aspects of a scenario.

Planning Methodologies

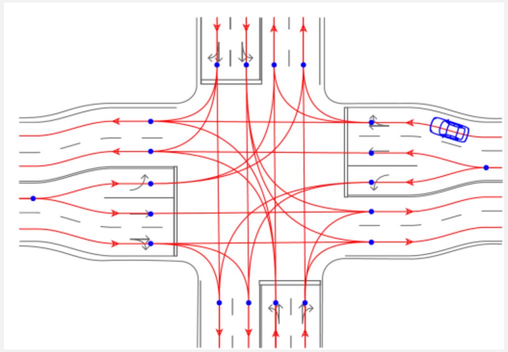
Variational Methods



$$\min J(\boldsymbol{x}(t))$$

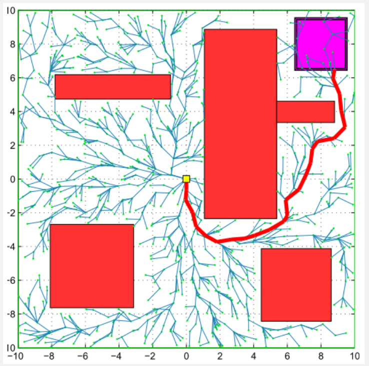
$$J(\boldsymbol{x}(t)) = \int_{t_0}^{t_0+T} L(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \dddot{\boldsymbol{x}})dt$$

Graph-Based Methods



G=(V, E)

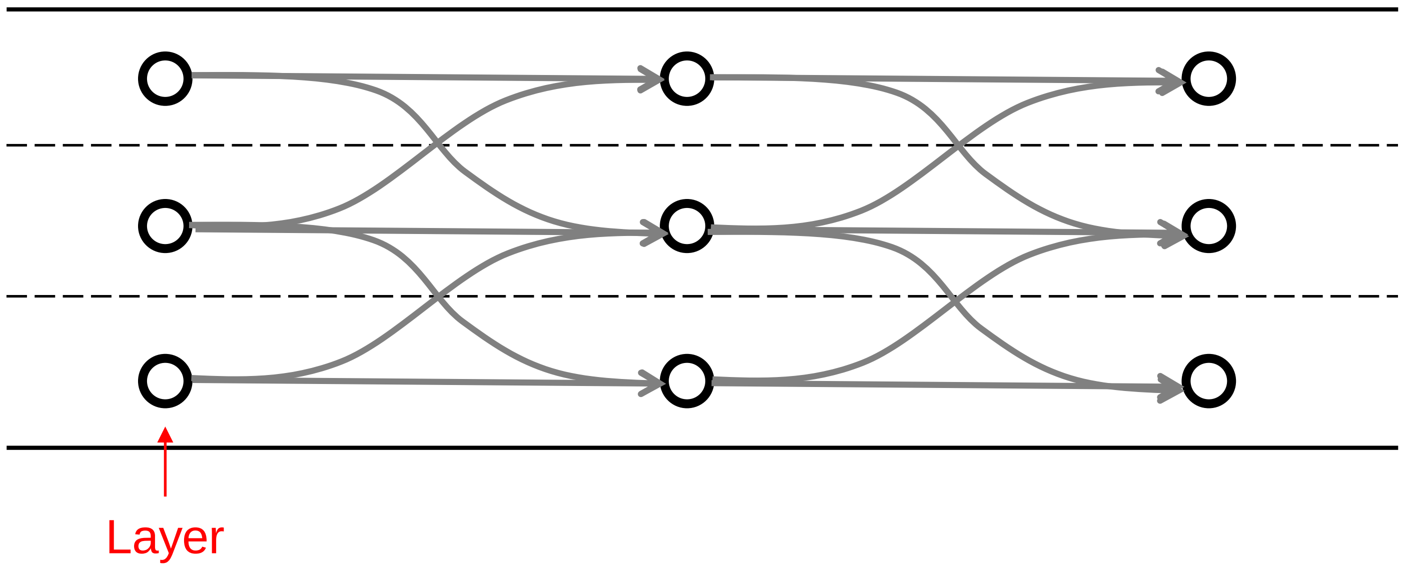
Incremental Methods

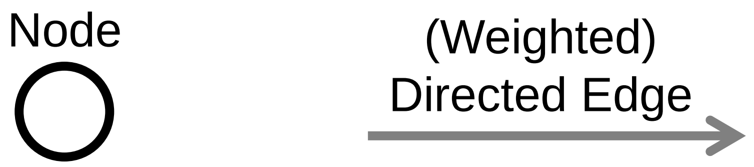


```
$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $i \leftarrow 0$;
while $i < N$ do
    $G \leftarrow (V, E)$;
    $x_{\text{rand}} \leftarrow$ Sample($i$); $i \leftarrow i + 1$;
    $(V, E) \leftarrow$ Extend($G, x_{\text{rand}}$);
```

Hybrid Methods

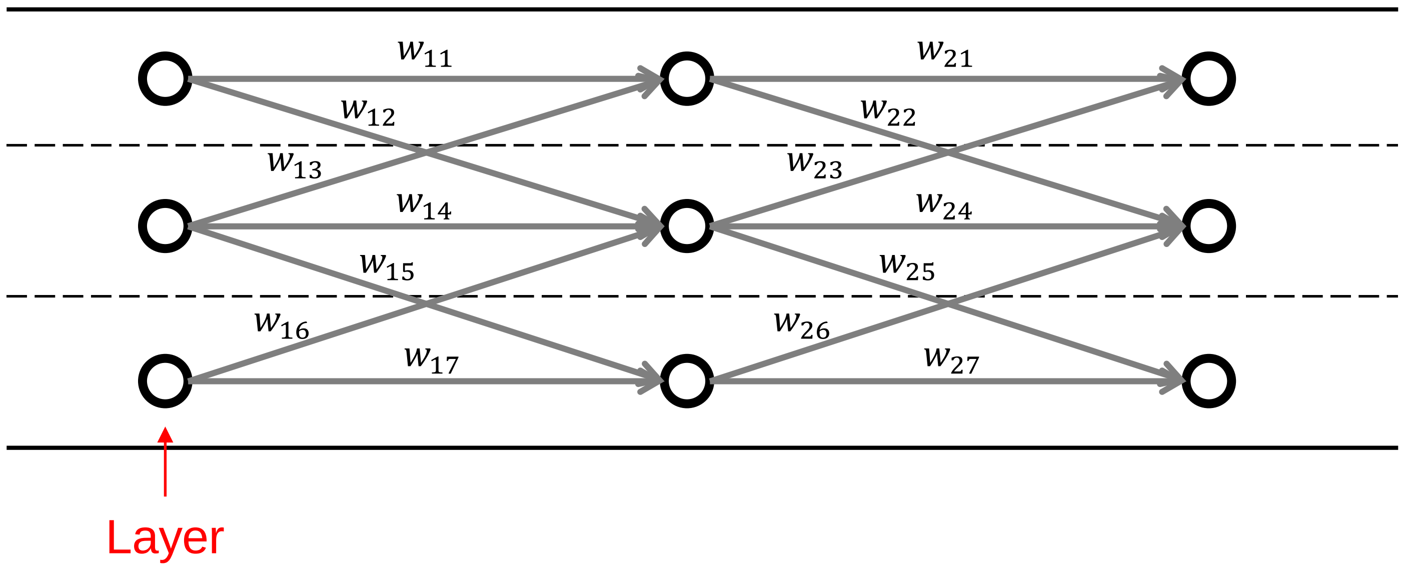
Graph-Based Methods
Example: Highway Path Planning

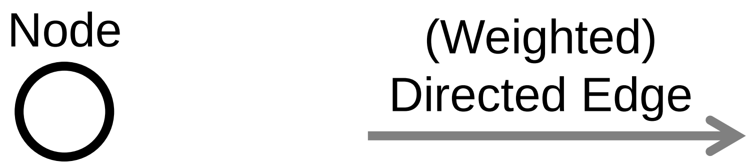




Edge weights are interpretable as **cost** (e.g. time, distance …)

Graph-Based Methods
Example: Highway Path Planning





Edge weights are interpretable as **cost** (e.g. time, distance …)

Additional Slides

Recap: A graph consists of nodes and edges connecting the nodes. If the edges are weighted, you can find the cost optimal path in a graph (see lecture 6 for details).

But how to use graphs for path planning?

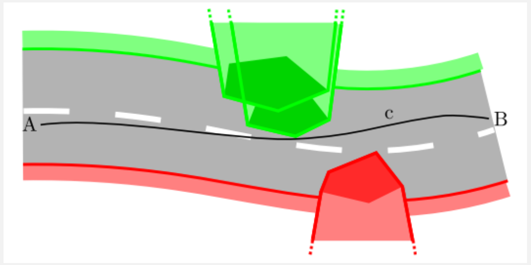
In graph-based methods the configuration space or state space is discretized. You place nodes at specific positions one the road. In the highway example there are three layers with three nodes each. The directed edges connecting the nodes represent possible paths between the layers. After assigning costs w_{ij} to each edge, it is possible to find the cost optimal path. The edges must ensure continuity at the transitions if feasibility is desired.

For trajectory planning it is necessary to extend the nodes with temporal information like time and velocity (this is not visualized in the highway example). By that we need to consider multiple nodes for every “path node” in the example. This can result in the curse of dimensionality.

In contrast to variational methods, the graph-based methods are not limited to local minima. But due to the fixed discretization it is only possible to find a suboptimal solution within the graph.

Planning Methodologies

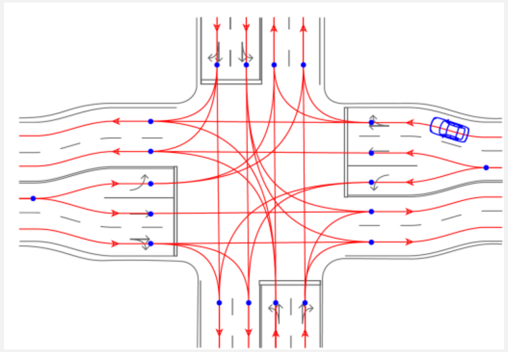
Variational Methods



$$\min J(\boldsymbol{x}(t))$$

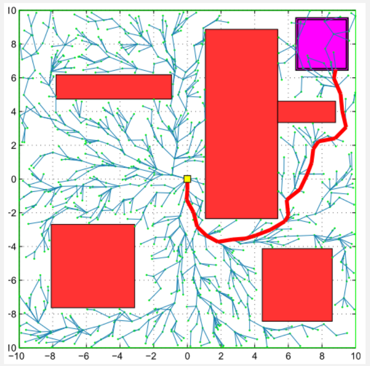
$$J(\boldsymbol{x}(t)) = \int_{t_0}^{t_0+T} L(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \dddot{\boldsymbol{x}})dt$$

Graph-Based Methods



G=(V, E)

Incremental Methods

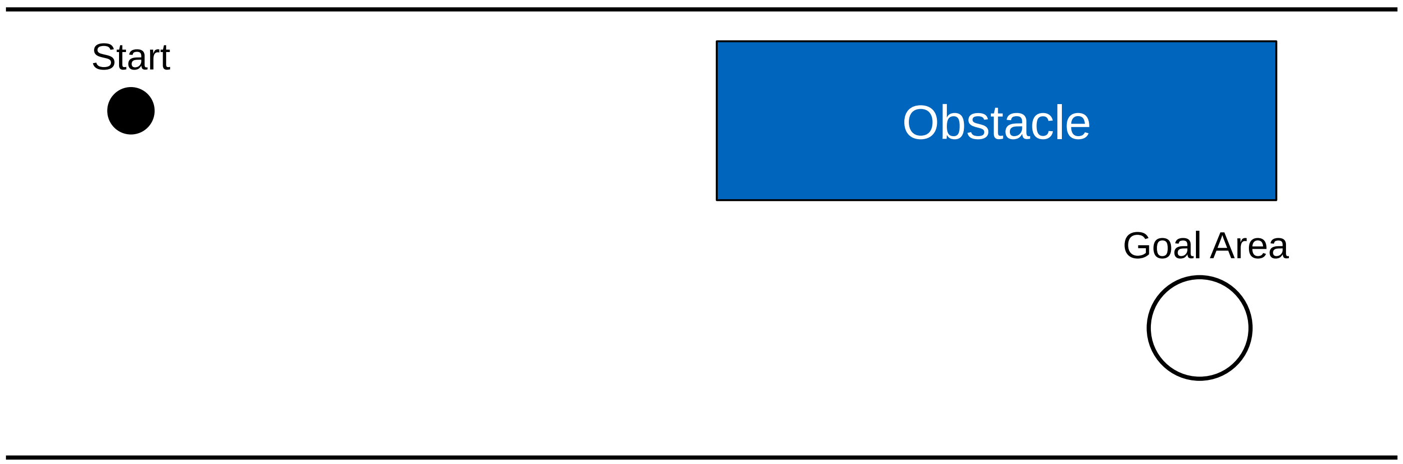


```
$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $i \leftarrow 0$;
while $i < N$ do
    $G \leftarrow (V, E)$;
    $x_{\text{rand}} \leftarrow$ Sample($i$); $i \leftarrow i + 1$;
    $(V, E) \leftarrow$ Extend($G, x_{\text{rand}}$);
```

Hybrid Methods

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

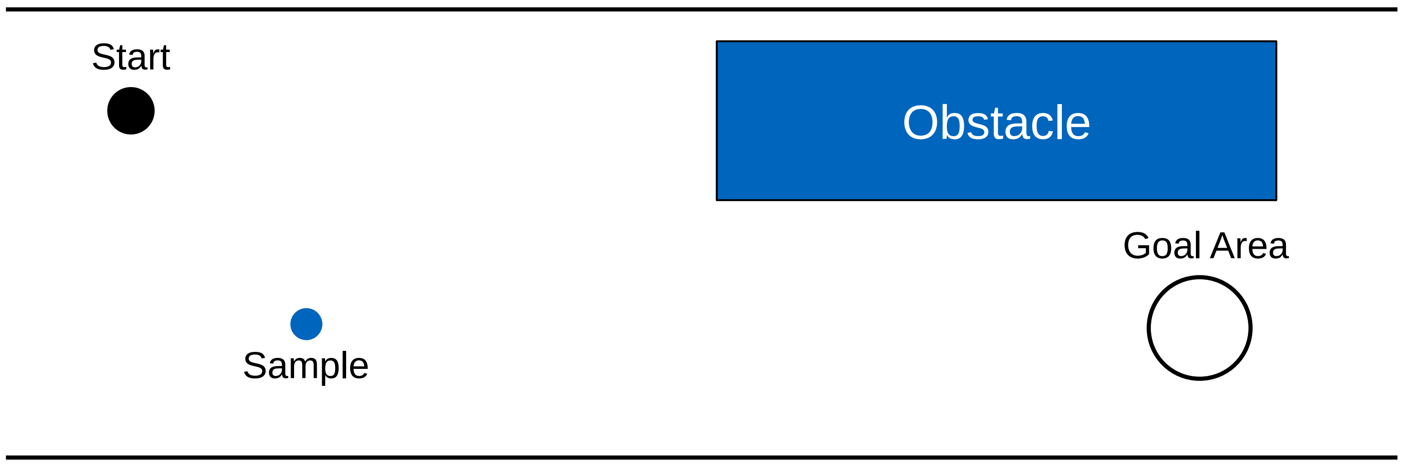
- Path planning algorithm proposed by La Valle in 1998



Repeat until goal are is reached:

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

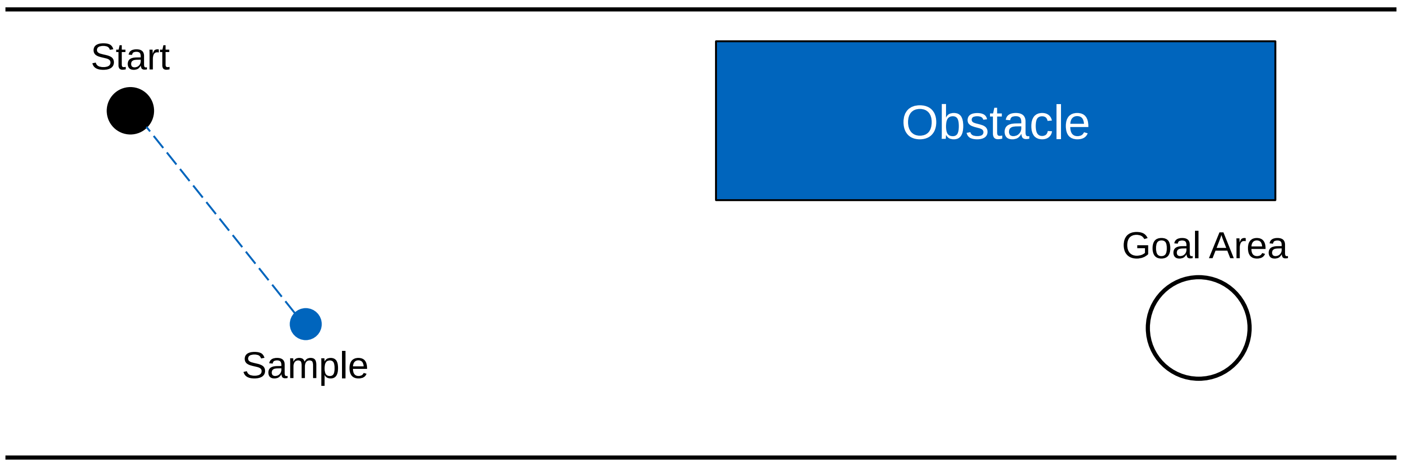


Repeat until goal are is reached:

1. Sample in configuration space

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

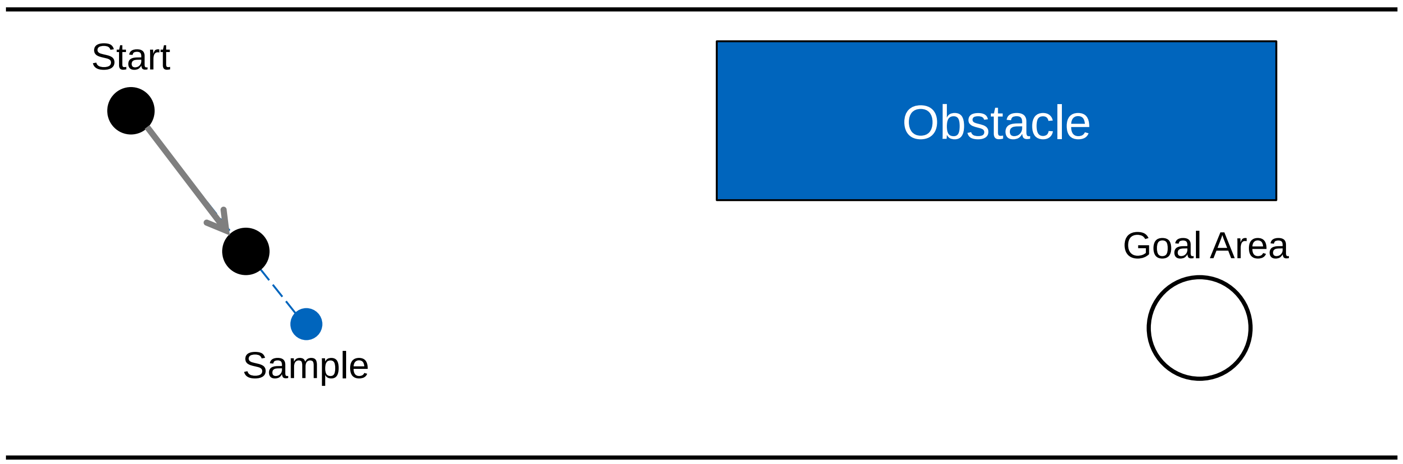


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

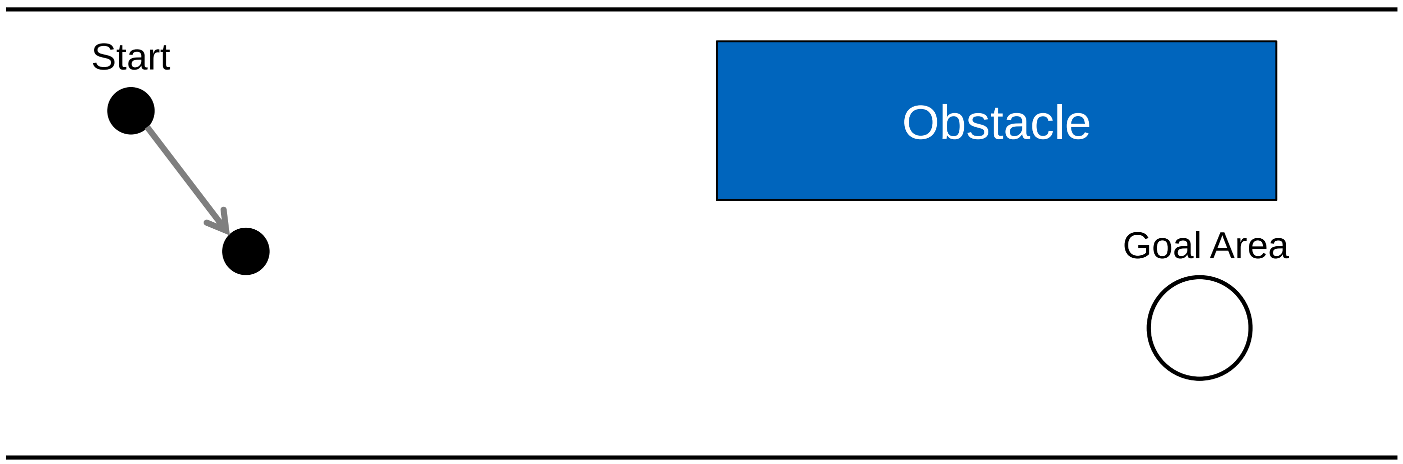


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point
   (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

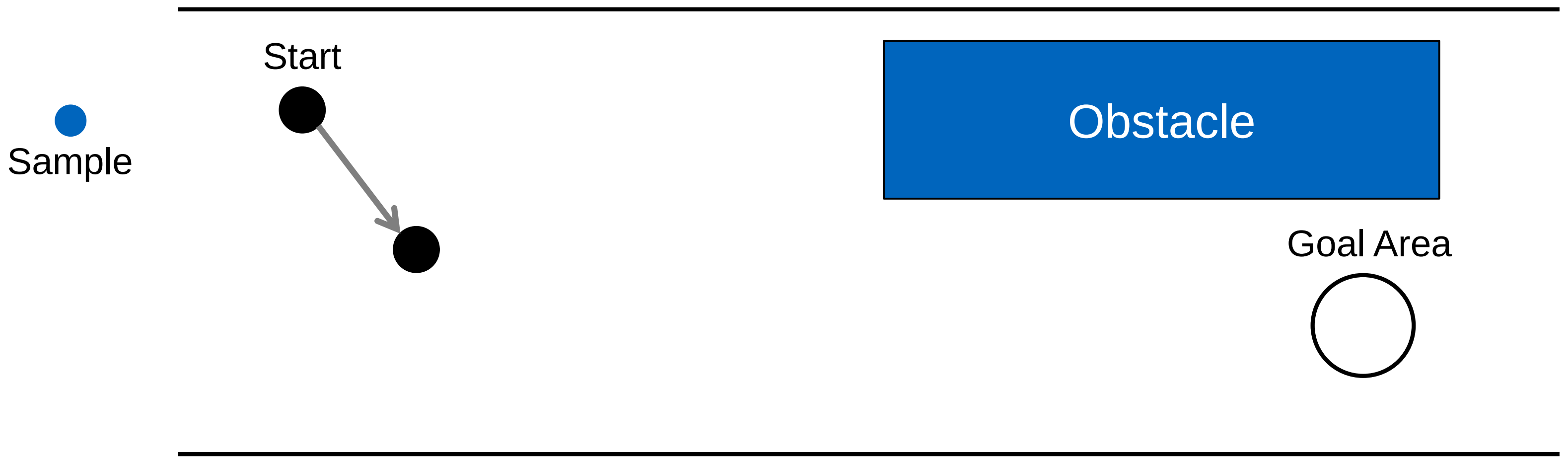


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point
   (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

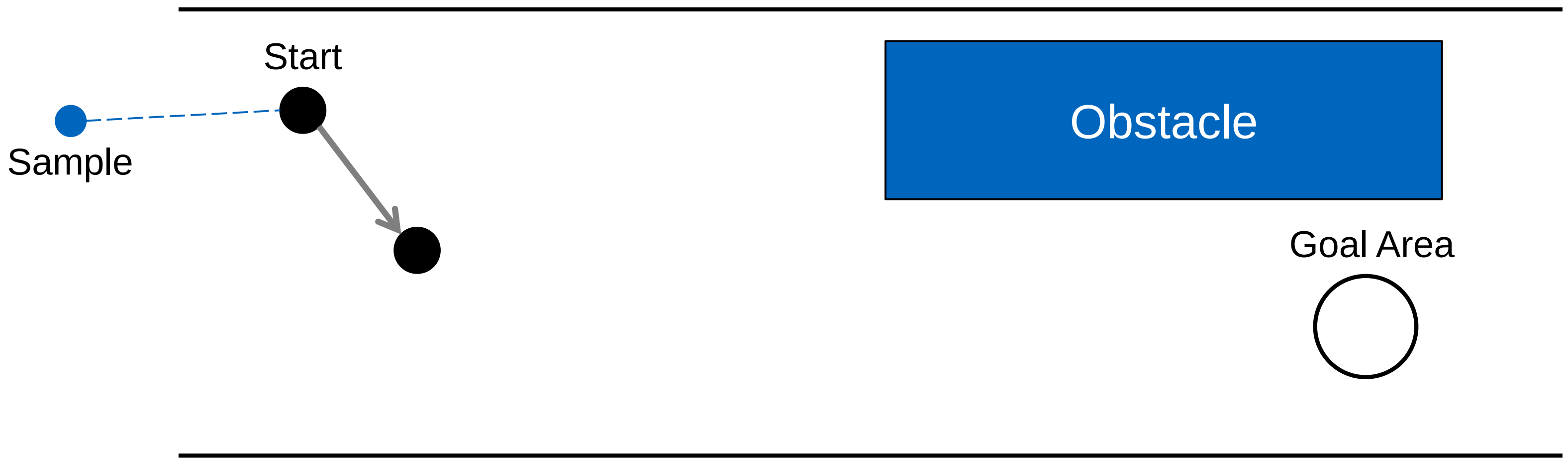


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

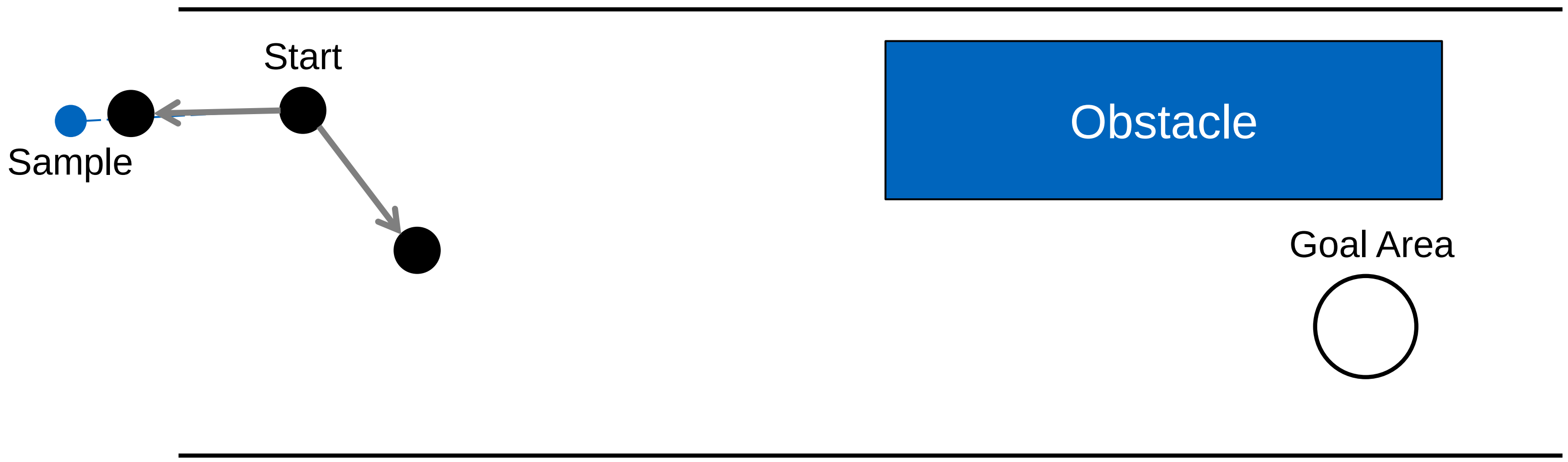


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

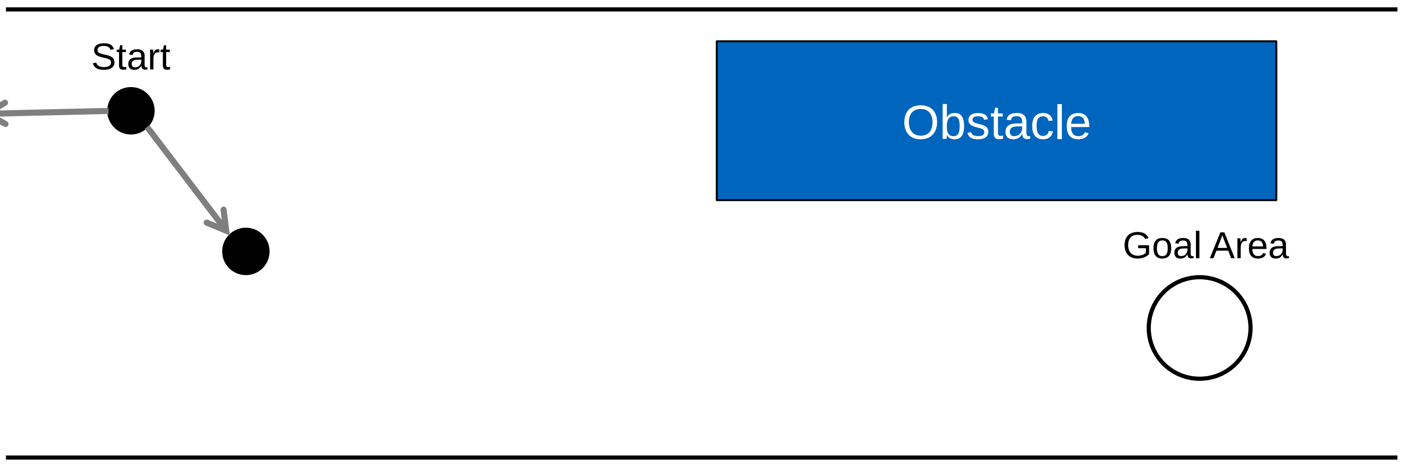


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point
   (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

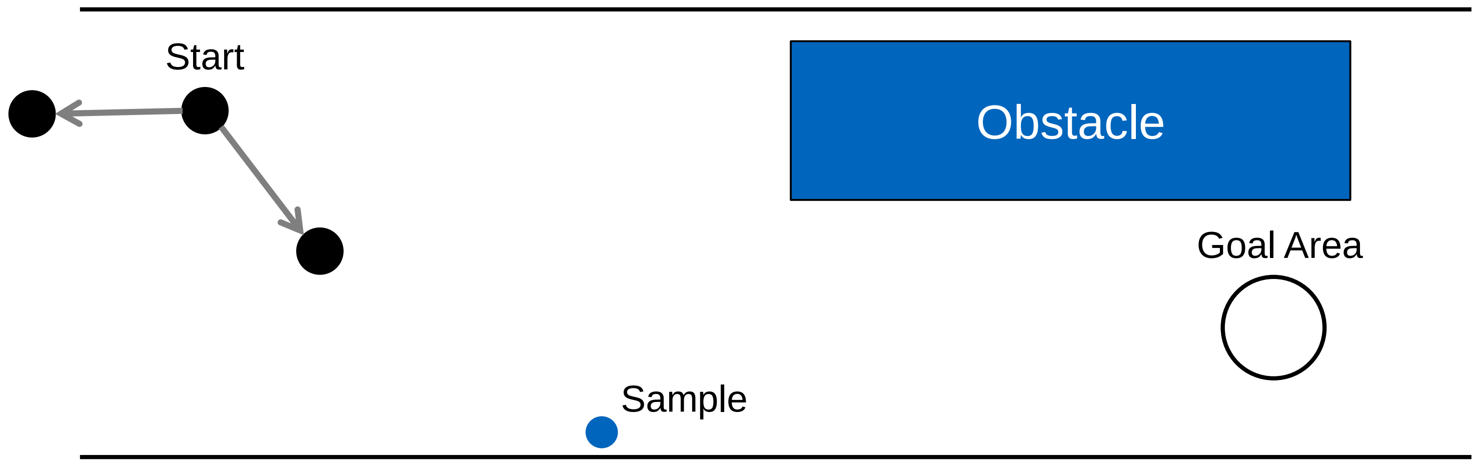


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point
   (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

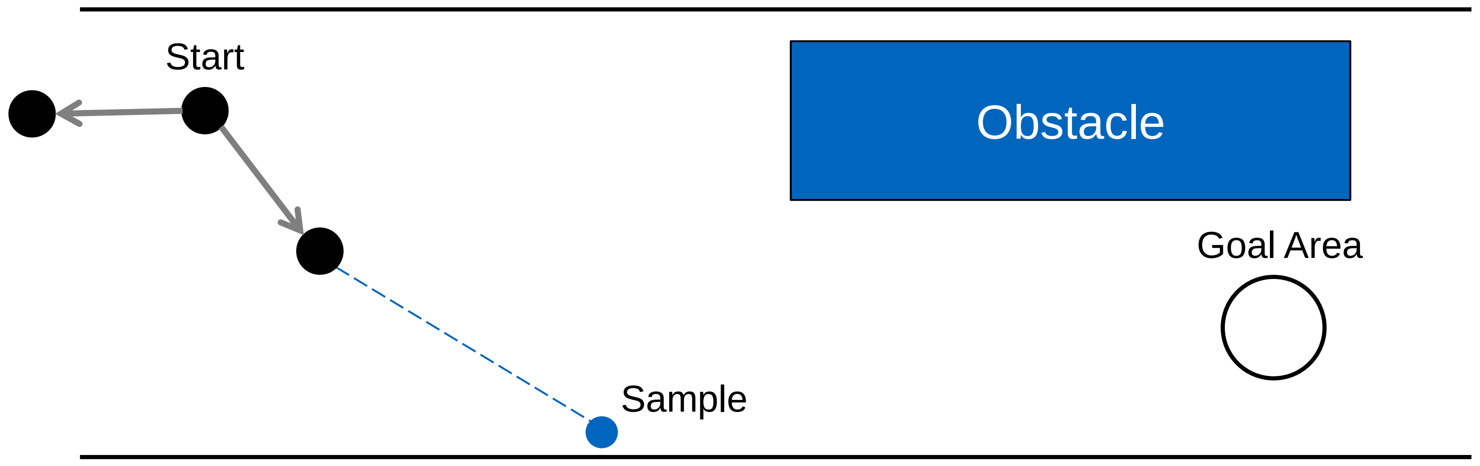


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

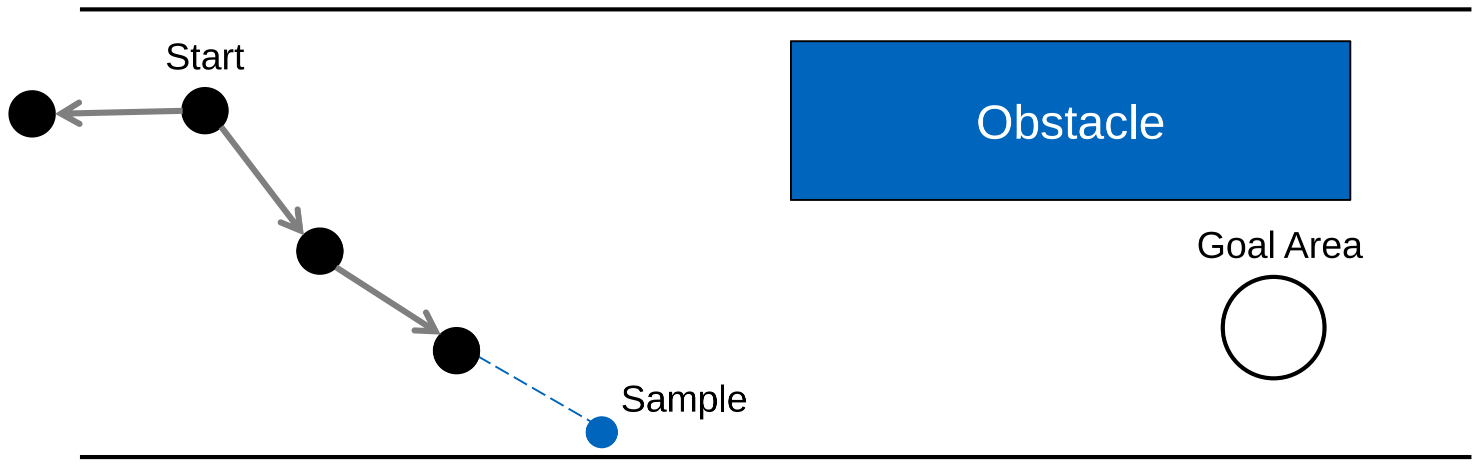


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point
   (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

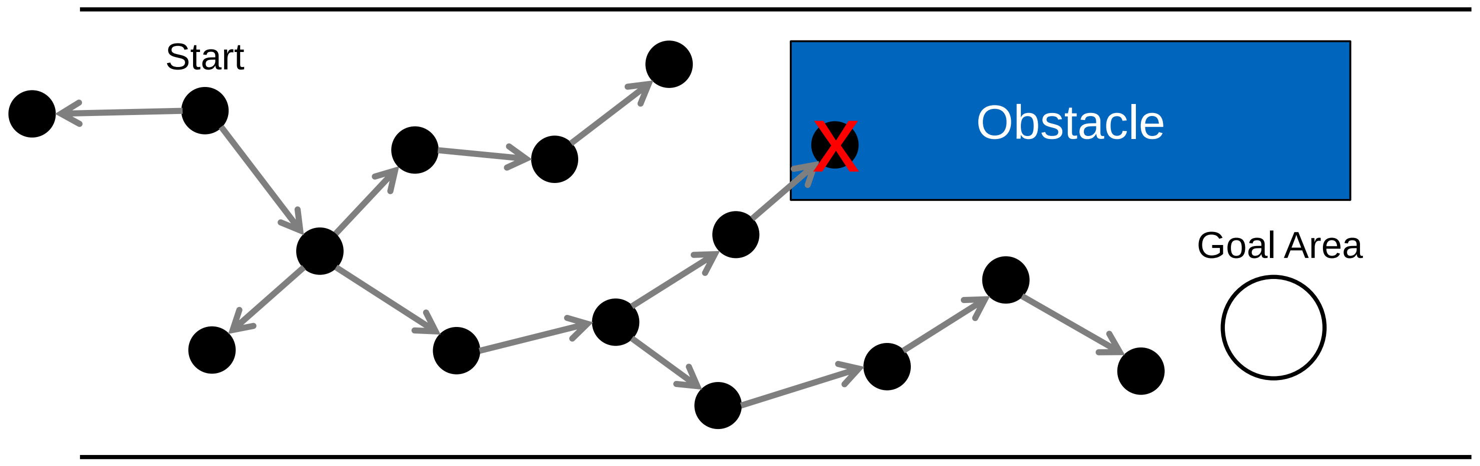


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

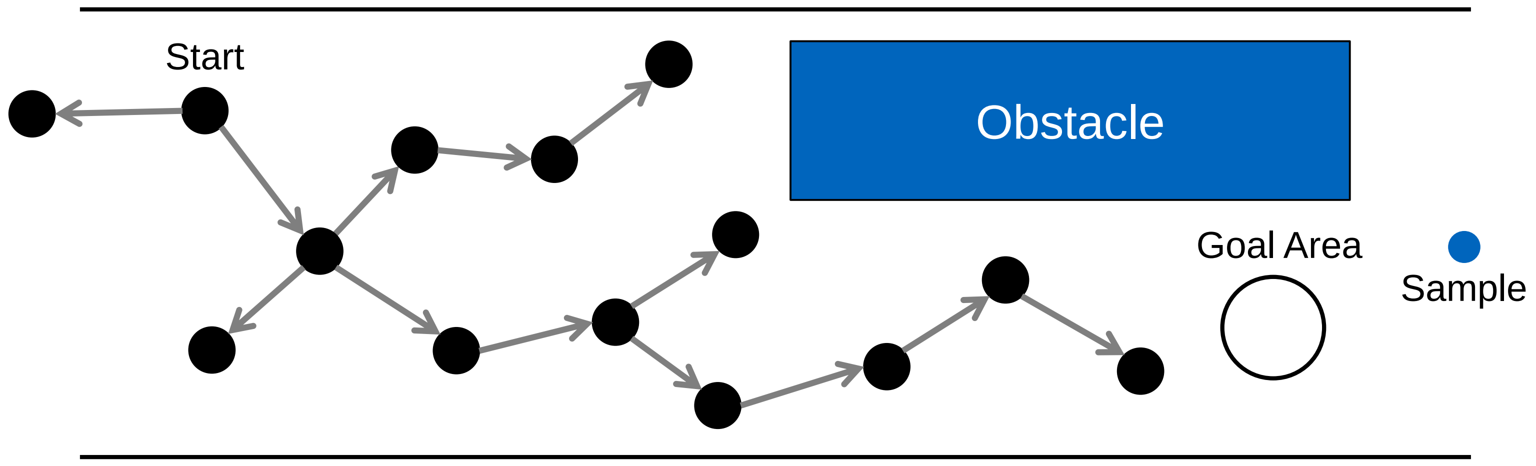


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

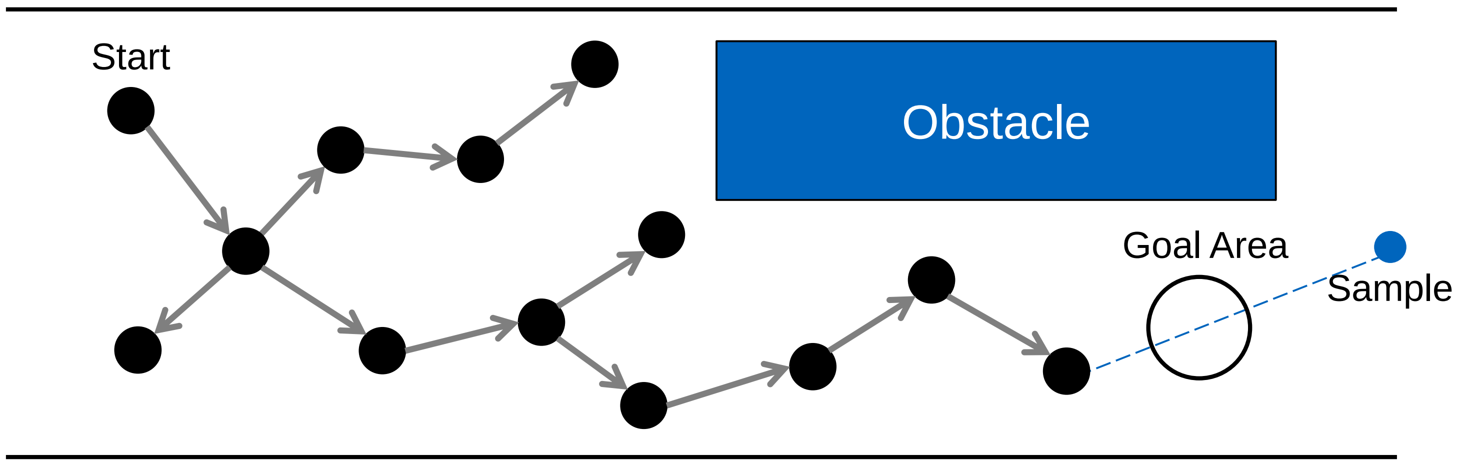


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point
   (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

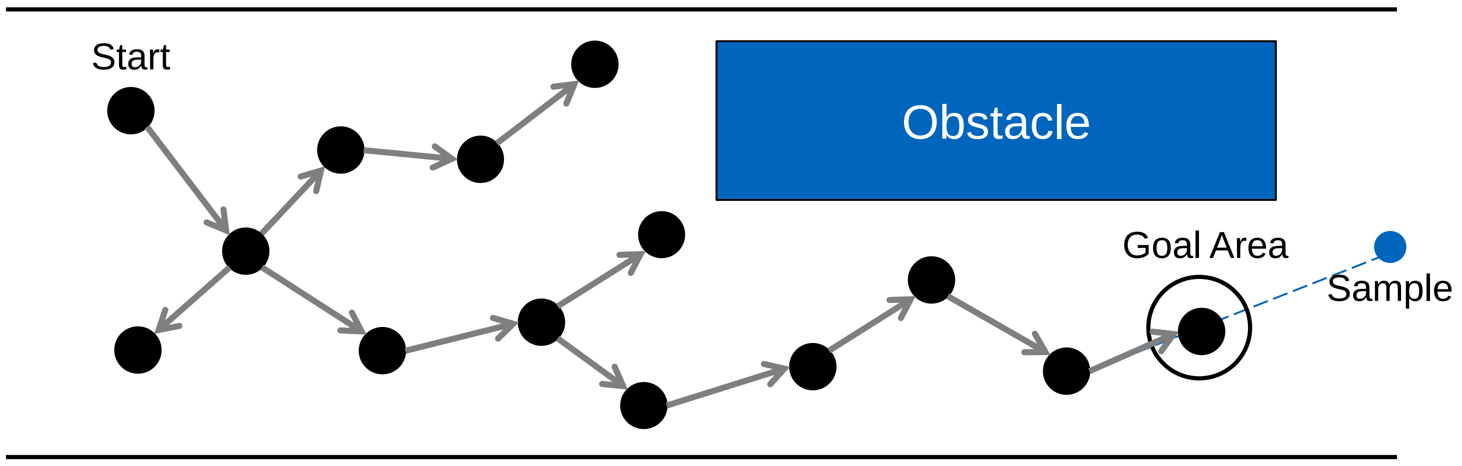


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point
   (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998

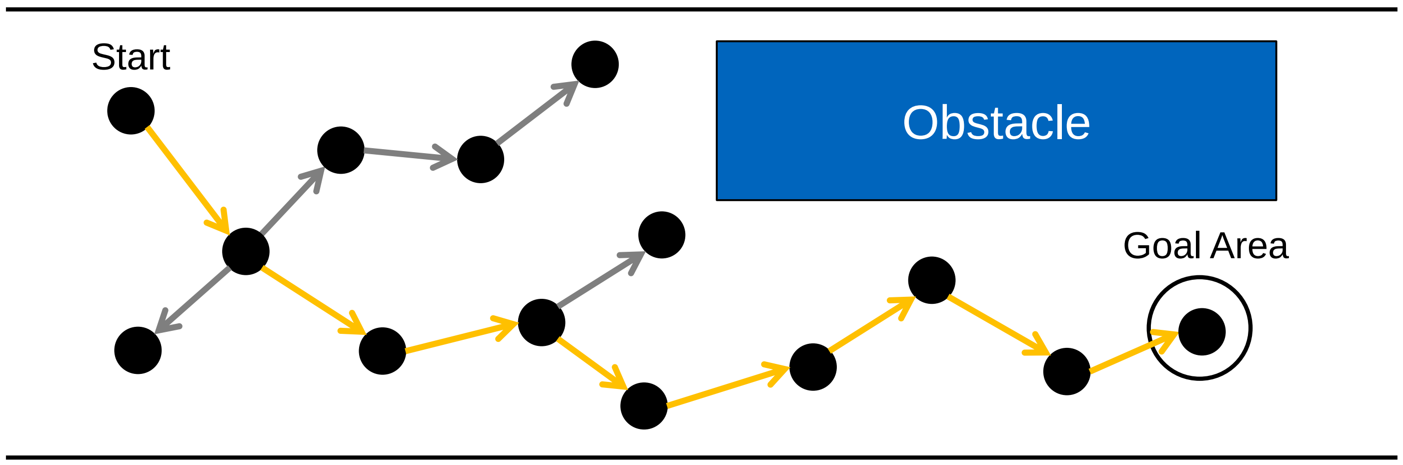


Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point
   (if new node is collision free)

Incremental Methods
Example: Rapidly-exploring random tree (RRT)

- Path planning algorithm proposed by La Valle in 1998
- Many extensions like RRT* possible



Repeat until goal are is reached:

1. Sample in configuration space
2. Get nearest node
3. Extend tree in direction of sample point (if new node is collision free)

Additional Slides

Incremental methods explore the state space incrementally. In contrast to graph-based methods these methods do not search within a fixed discretization. Therefore the incremental methods will find a solution (if one exists) with probability approaching one, given enough computation time. This property is called *probabilistically completeness.*

Typically, the discretization of these methods is getting incrementally finer. That is why in simple scenarios the solution will be found quickly, but in general the computation is unbounded.

One of the simplest incremental methods is the introduced RRT algorithm for path planning proposed by La Valle [1] in 1998. In every step a random point in configuration space is sampled. Next, the nearest node from the current tree to the sampled point is determined. From the nearest node the tree is extended with a specific length in the direction of the sampled point. This procedure is repeated until there exists a direct path between start node and goal area.

RRT will always find a solution, given enough computation time. But this solution will not be optimal (you can see, that the resulting path is ragged). To tackle this problem Karaman and Frazzoli [2] proposed in 2010 the RRT* algorithm, which will determine the optimal solution with probability one by time. This property is called *asymptotically optimality*.

In addition to RRT*, there are other variants of the basic RRT algorithm that are beyond the scope of this lecture.

[1] S. M. La Valle, “Rapidly-exploring random trees a new tool for path planning,” tech. rep., Computer Science Dept., Iowa State University, 1998.

[2] S. Karaman and E. Frazzoli, “Optimal kinodynamic motion planning using incremental sampling-based methods,” in Conference on Decision and Control, pp. 7681–7687, IEEE, 2010.

Comparison

	+	-
Variational Methods	• No discretization • Low computation time	• Finds local optimum • Cost dependent solver
Graph-Based Methods	• Finds global optimum • Flexible cost function	• Discretized solution • Curse of dimensionality
Incremental Methods	• Probabilistically complete	• Solution in finite time not guaranteed

Local Planning
Prof. Dr. Markus Lienkamp

Levent Ögretmen, M. Sc.

Agenda

1. Introduction
2. Local Planning Methodologies
3. **Deep-Dive Graph-Based Planning**
4. Collision Checking
5. Summary

Graph-Based Planning Deep-Dive
Naïve Approach

- Goal: Graph-based trajectory planning
- Approach: Extending the spatial nodes by velocity and time dimension

- Example: Indianapolis Motor Speedway (4 km racetrack)
  - Nodes ≈ 32.000.000
  - Edges ≈ 288.000.000

→"Curse of Dimensionality"

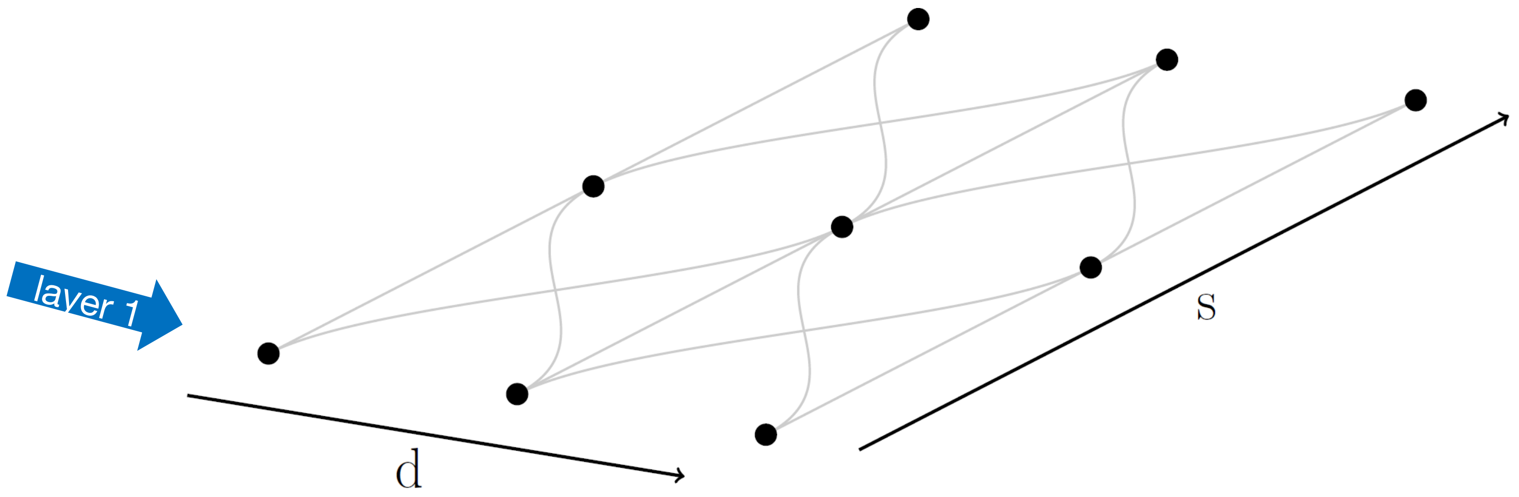
https://www.indianapolismotorspeedway.com/

Additional Slides

To plan a path with a graph, we learned that we can simply discretize the configuration space and use edges as representations of possible paths. The path with minimal cost will be the best path.

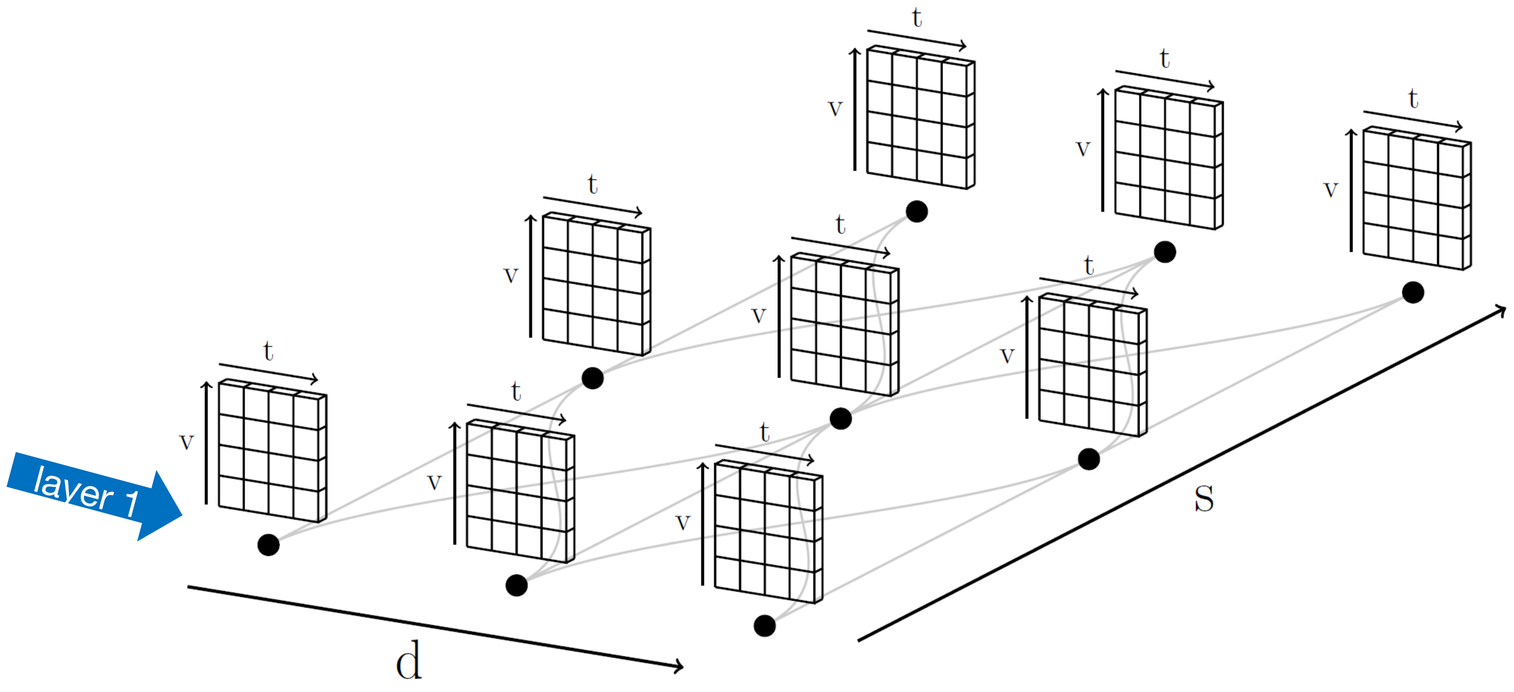
Additional for trajectory planning it is necessary to extend the considered space by the velocity and time dimension. Theoretically this would be possible, but the number of nodes and edges would increase rapidly, which results in a difficult realization. That is why other approaches need to be discussed.

Graph-Based Planning Deep-Dive
Approach by McNaughton [1]



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

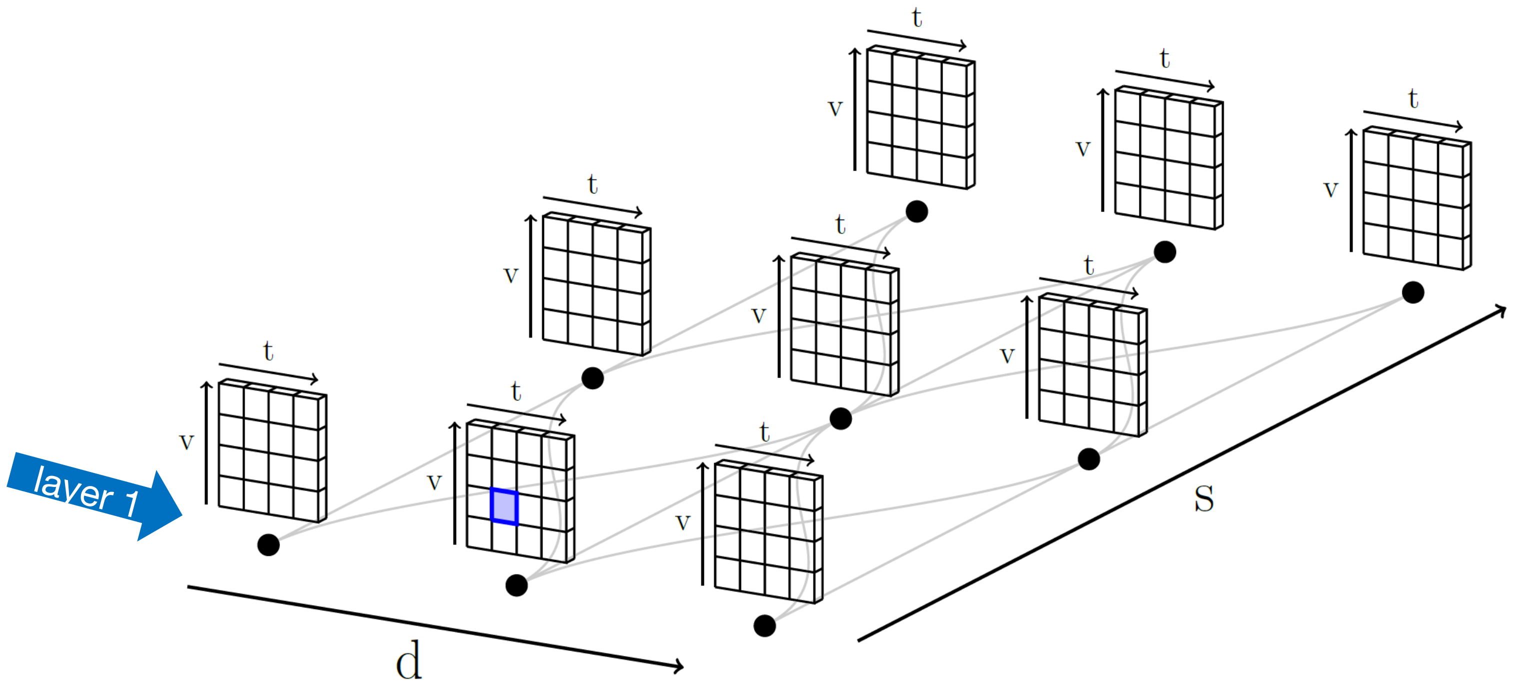
Graph-Based Planning Deep-Dive
Approach by McNaughton [1]



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Graph-Based Planning Deep-Dive
Approach by McNaughton [1]

1. Get start nodes in layer



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

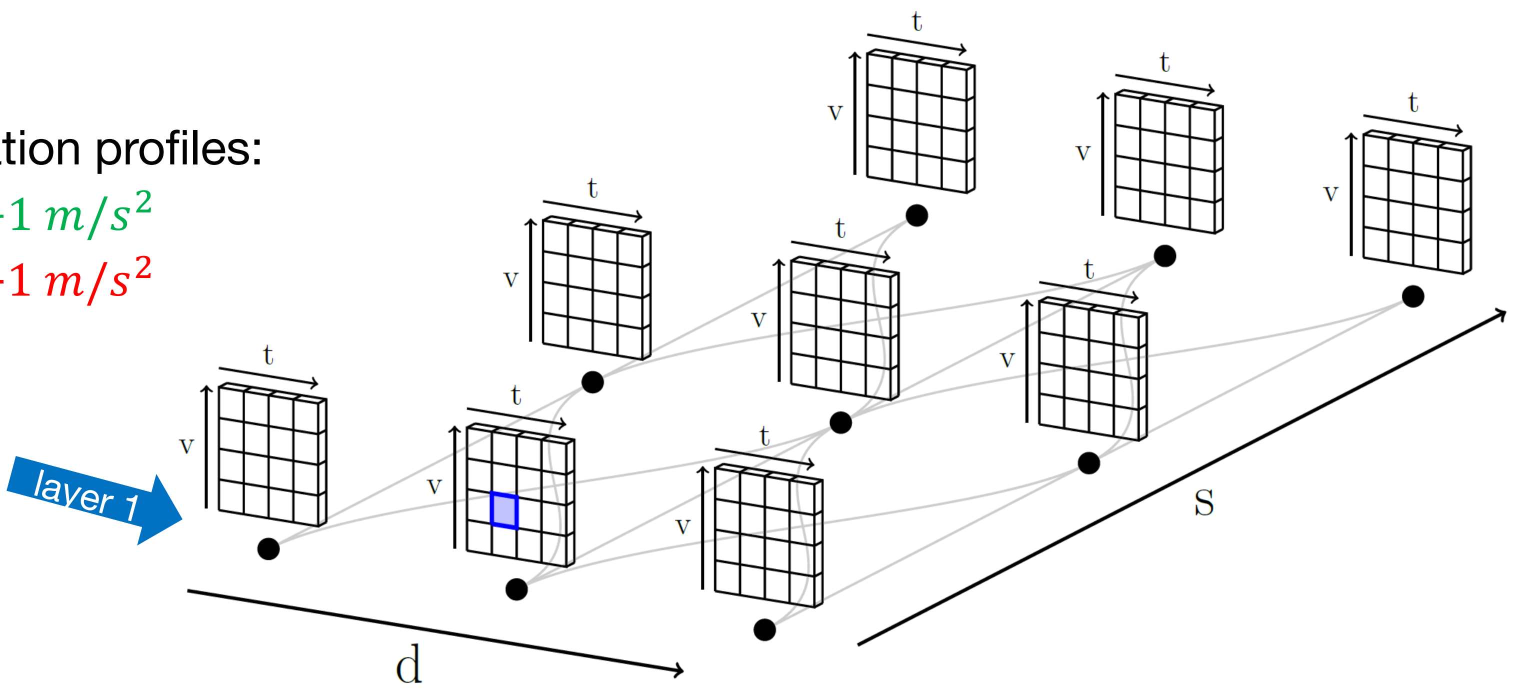
Graph-Based Planning Deep-Dive
Approach by McNaughton [1]

```
1. Get start nodes in layer
2. For every start node
     → Apply acceleration profiles on each path
```

Acceleration profiles:

$a = +1\, m/s^2$

$a = -1\, m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Graph-Based Planning Deep-Dive

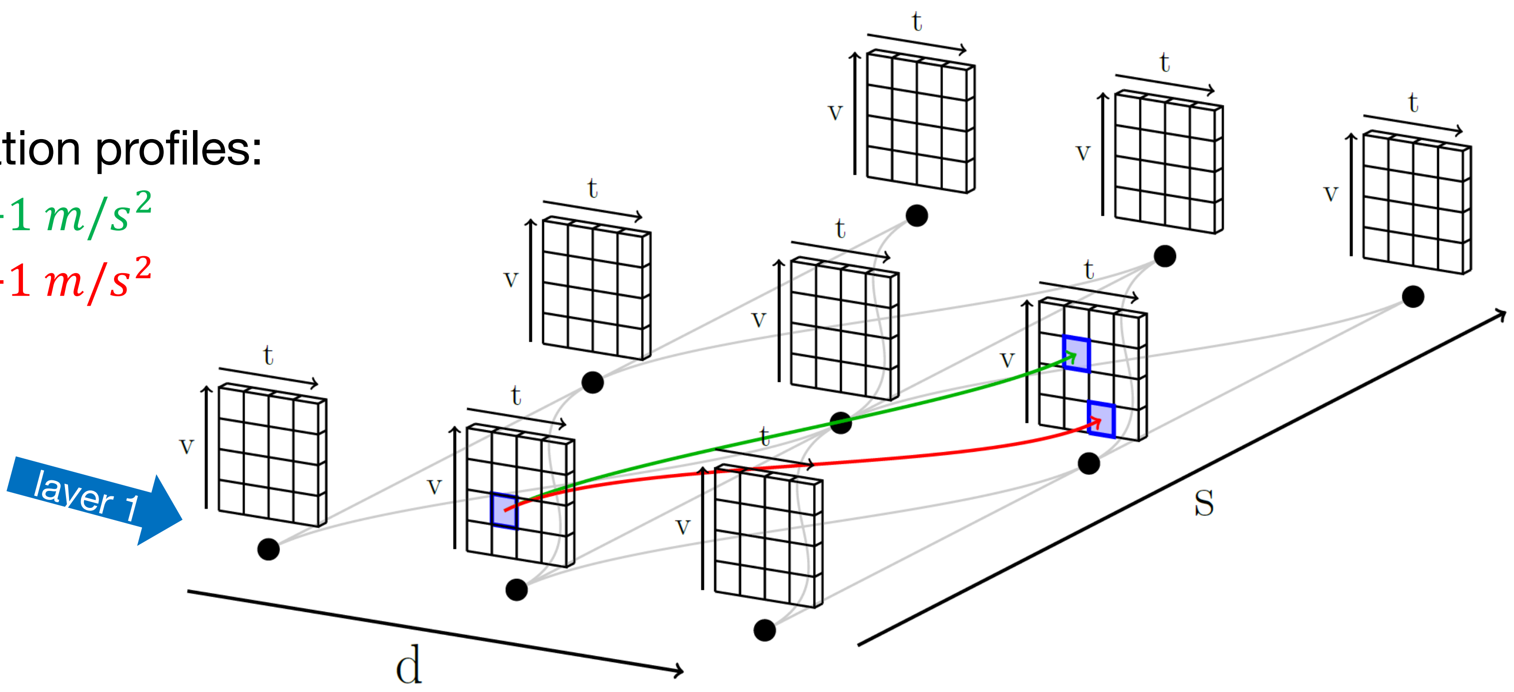
Approach by McNaughton [1]

```
1. Get start nodes in layer
2. For every start node
     → Apply acceleration profiles on each path
```

Acceleration profiles:

$a = +1\, m/s^2$

$a = -1\, m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

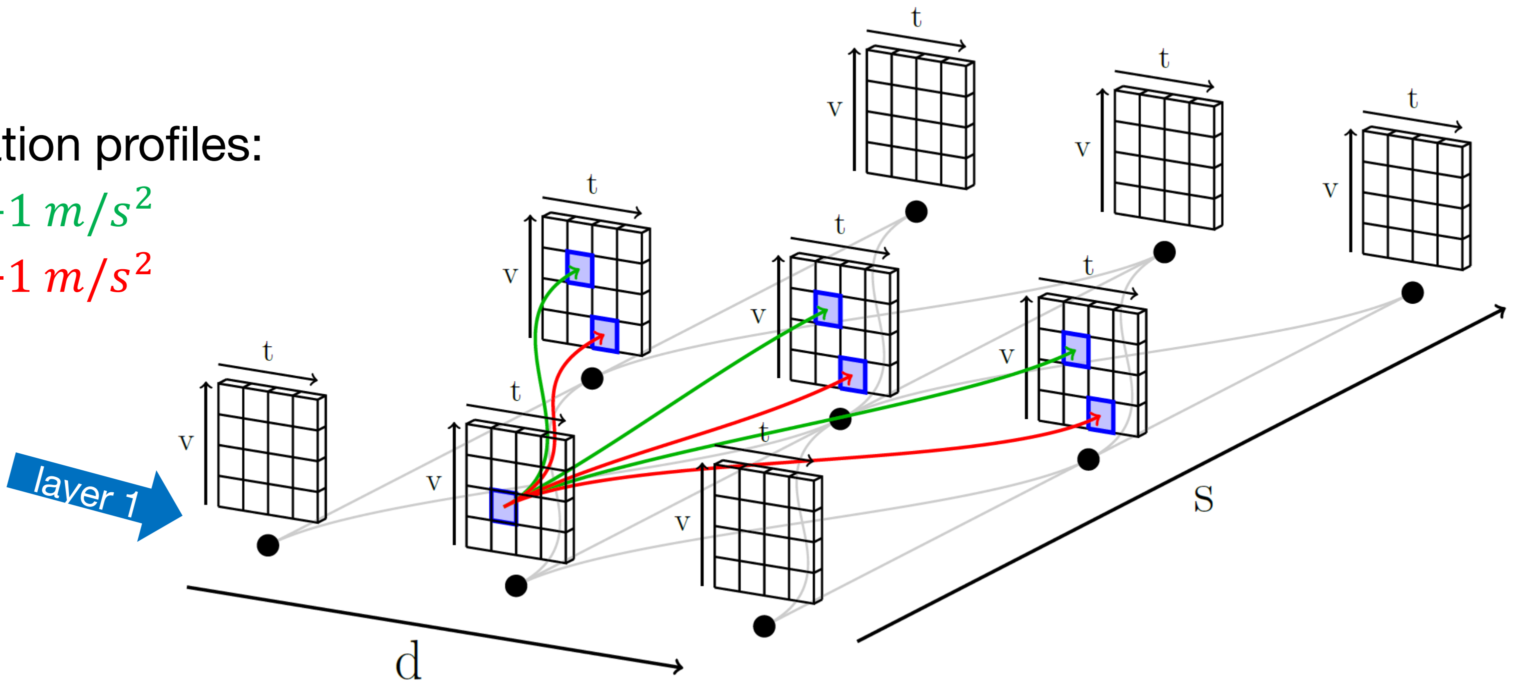
Graph-Based Planning Deep-Dive
Approach by McNaughton [1]

```
1. Get start nodes in layer
2. For every start node
       → Apply acceleration profiles on each path
```

Acceleration profiles:

$a = +1\, m/s^2$

$a = -1\, m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

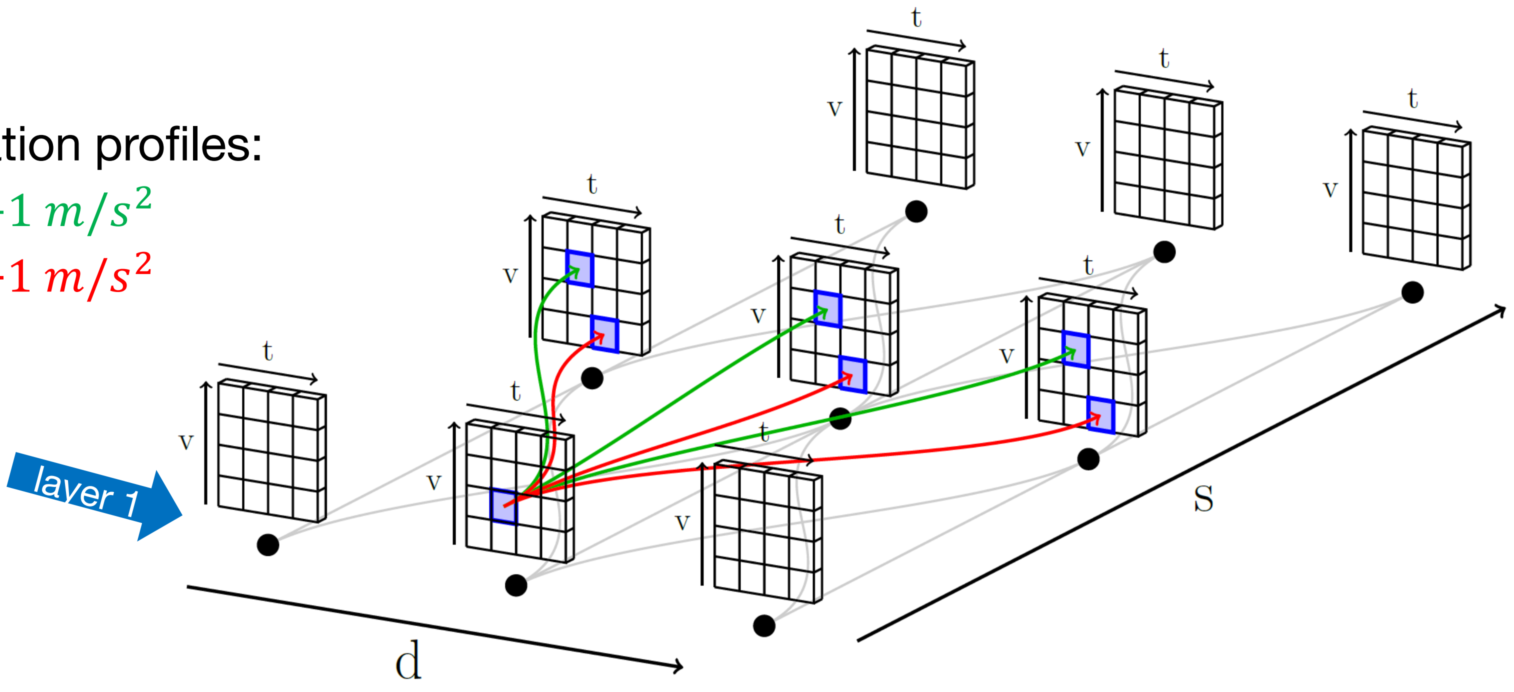
Graph-Based Planning Deep-Dive
Approach by McNaughton [1]

```
1. Get start nodes in layer
2. For every start node
     → Apply acceleration profiles on each path
3. If multiple edges end in same interval
     → Remove edges with higher costs
```

Acceleration profiles:

$a = +1\, m/s^2$

$a = -1\, m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Graph-Based Planning Deep-Dive
Approach by McNaughton [1]

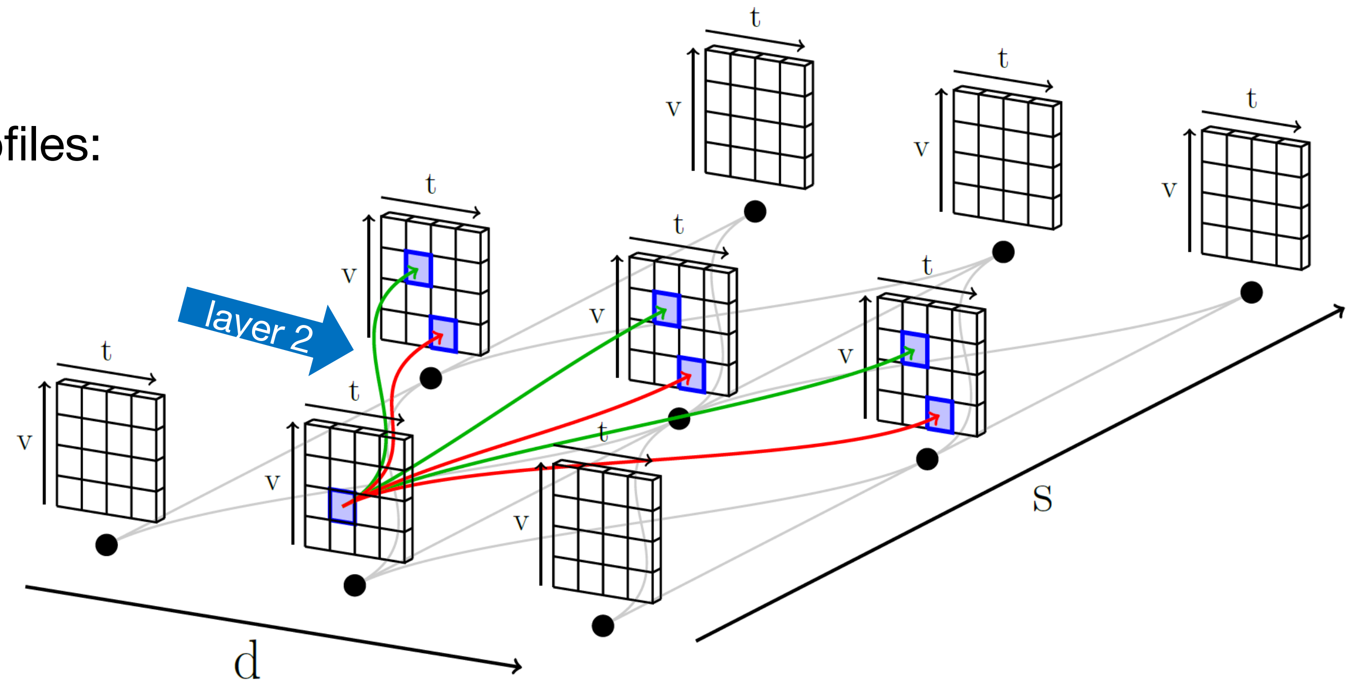
next layer

```
1. Get start nodes in layer
2. For every start node
       → Apply acceleration profiles on each path
3. If multiple edges end in same interval
       → Remove edges with higher costs
```

Acceleration profiles:

$a = +1\,m/s^2$

$a = -1\,m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Graph-Based Planning Deep-Dive
Approach by McNaughton [1]

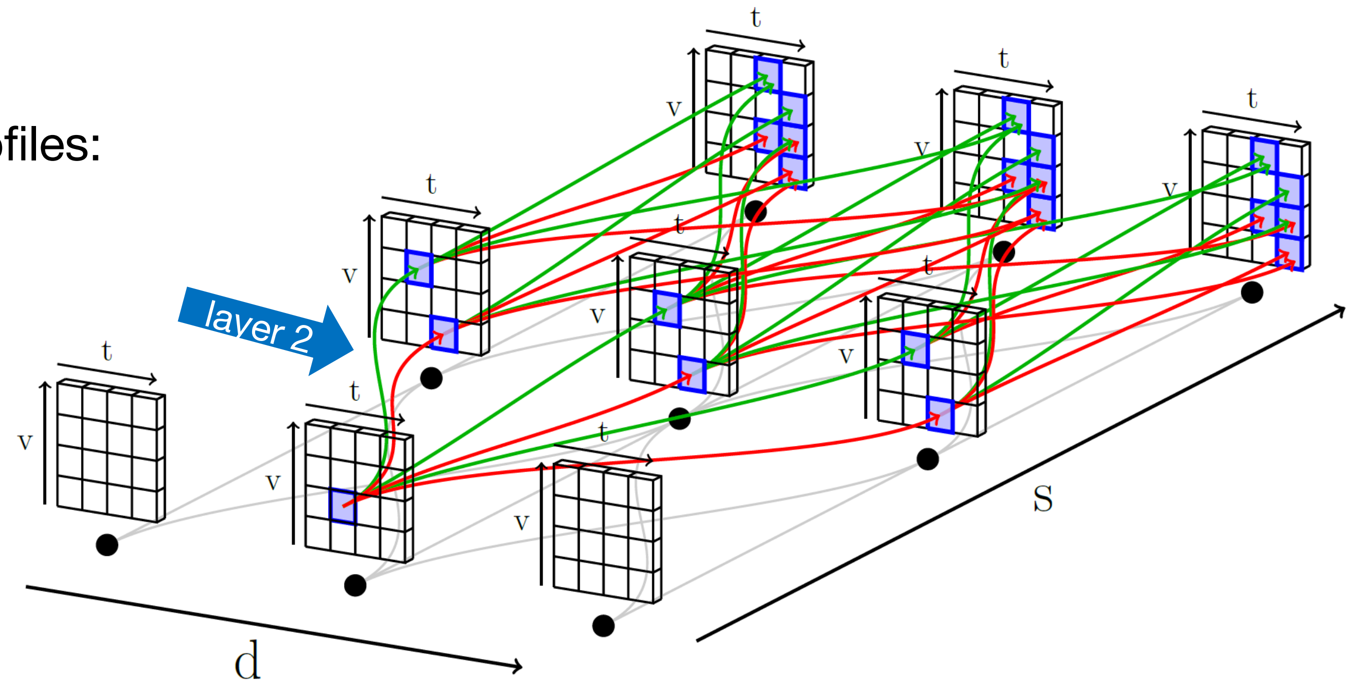
next layer

```
1. Get start nodes in layer
2. For every start node
       → Apply acceleration profiles on each path
3. If multiple edges end in same interval
       → Remove edges with higher costs
```

Acceleration profiles:

$a = +1\, m/s^2$

$a = -1\, m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Graph-Based Planning Deep-Dive

Approach by McNaughton [1]

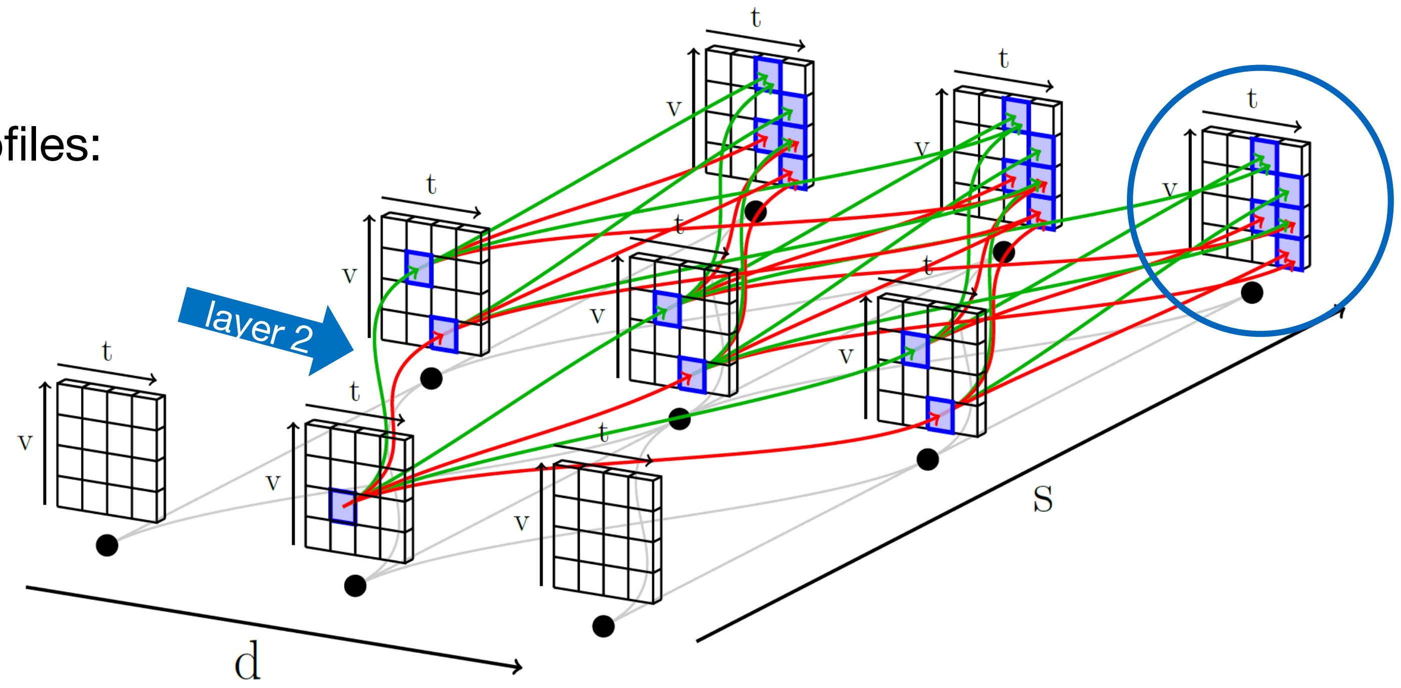
next layer

```
1. Get start nodes in layer
2. For every start node
      → Apply acceleration profiles on each path
3. If multiple edges end in same interval
      → Remove edges with higher costs
```

Acceleration profiles:

$a = +1\,m/s^2$

$a = -1\,m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Graph-Based Planning Deep-Dive

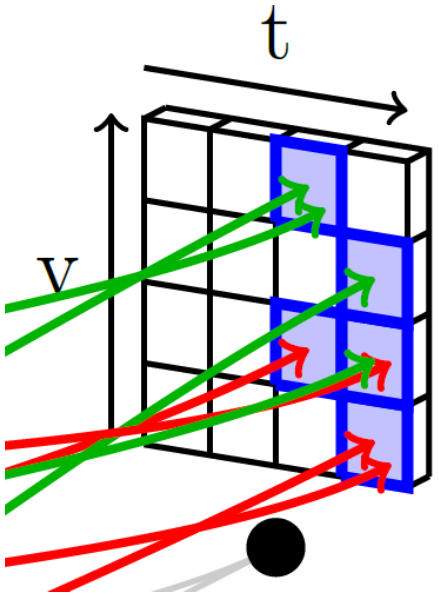
Approach by McNaughton [1]

```
next layer
1. Get start nodes in layer
2. For every start node
      → Apply acceleration profiles on each path
3. If multiple edges end in same interval
      → Remove edges with higher costs
```

Acceleration profiles:

$a = +1\ m/s^2$

$a = -1\ m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Graph-Based Planning Deep-Dive
Approach by McNaughton [1]

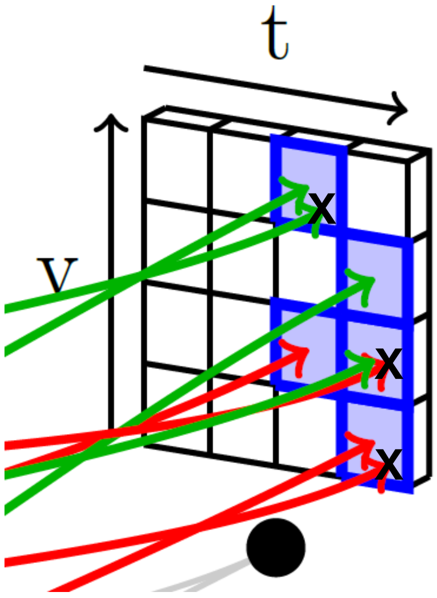
next layer

```
1. Get start nodes in layer
2. For every start node
     → Apply acceleration profiles on each path
3. If multiple edges end in same interval
     → Remove edges with higher costs
```

Acceleration profiles:

$a = +1\, m/s^2$

$a = -1\, m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Graph-Based Planning Deep-Dive

Approach by McNaughton [1]

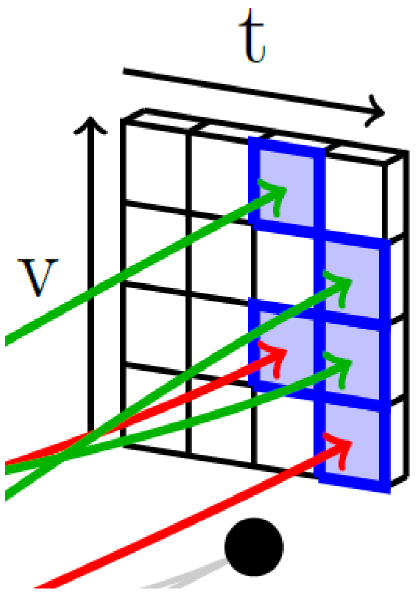
next layer

```
1. Get start nodes in layer
2. For every start node
      → Apply acceleration profiles on each path
3. If multiple edges end in same interval
      → Remove edges with higher costs
```

Acceleration profiles:

$a = +1\, m/s^2$

$a = -1\, m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Graph-Based Planning Deep-Dive

Approach by McNaughton [1]

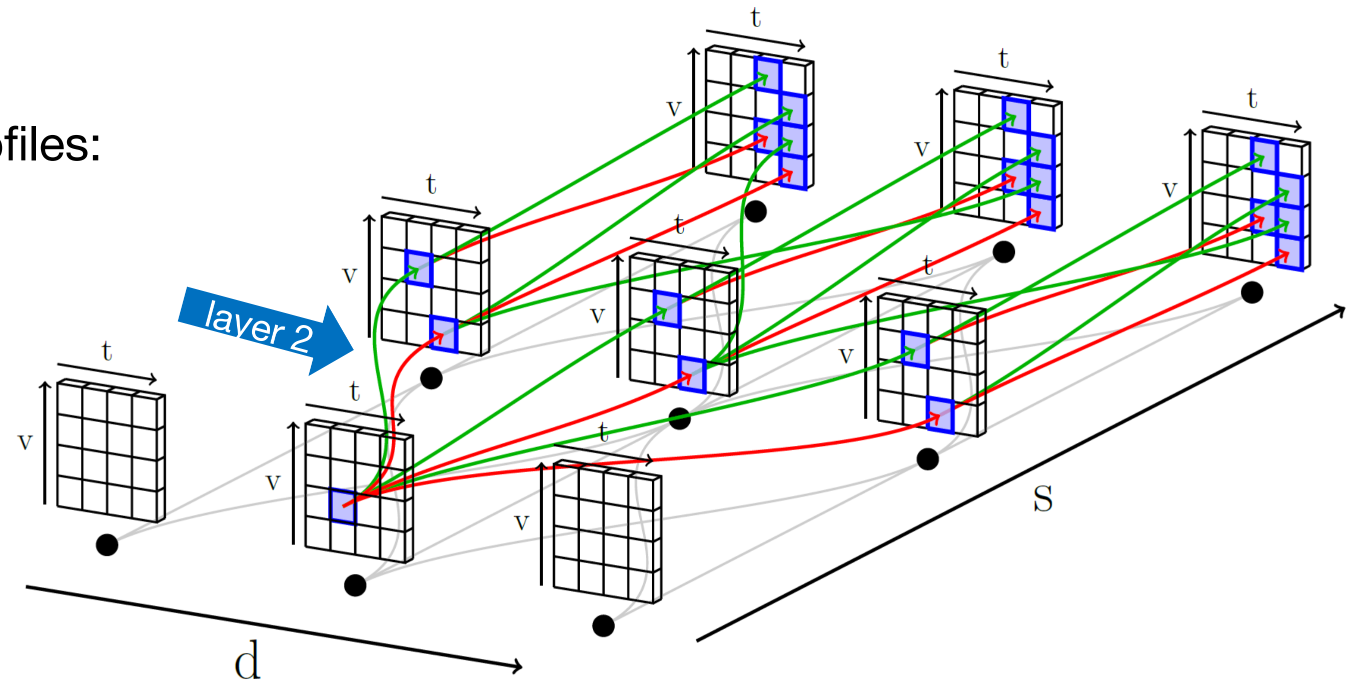
next layer

```
1. Get start nodes in layer
2. For every start node
      → Apply acceleration profiles on each path
3. If multiple edges end in same interval
      → Remove edges with higher costs
```

Acceleration profiles:

$a = +1\,m/s^2$

$a = -1\,m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

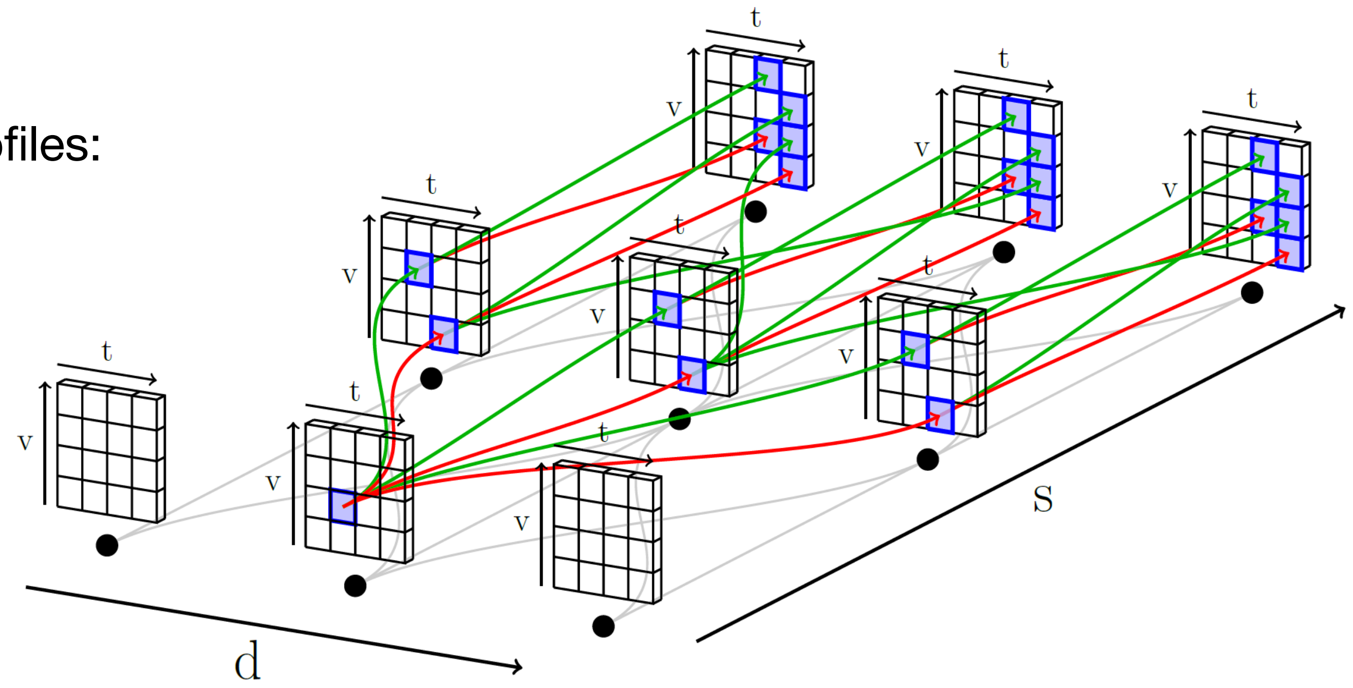
Graph-Based Planning Deep-Dive
Approach by McNaughton [1]

- Algorithm stops, when desired planning horizon is reached
  → Result is a family of possible trajectories

Acceleration profiles:

$a = +1\ m/s^2$

$a = -1\ m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

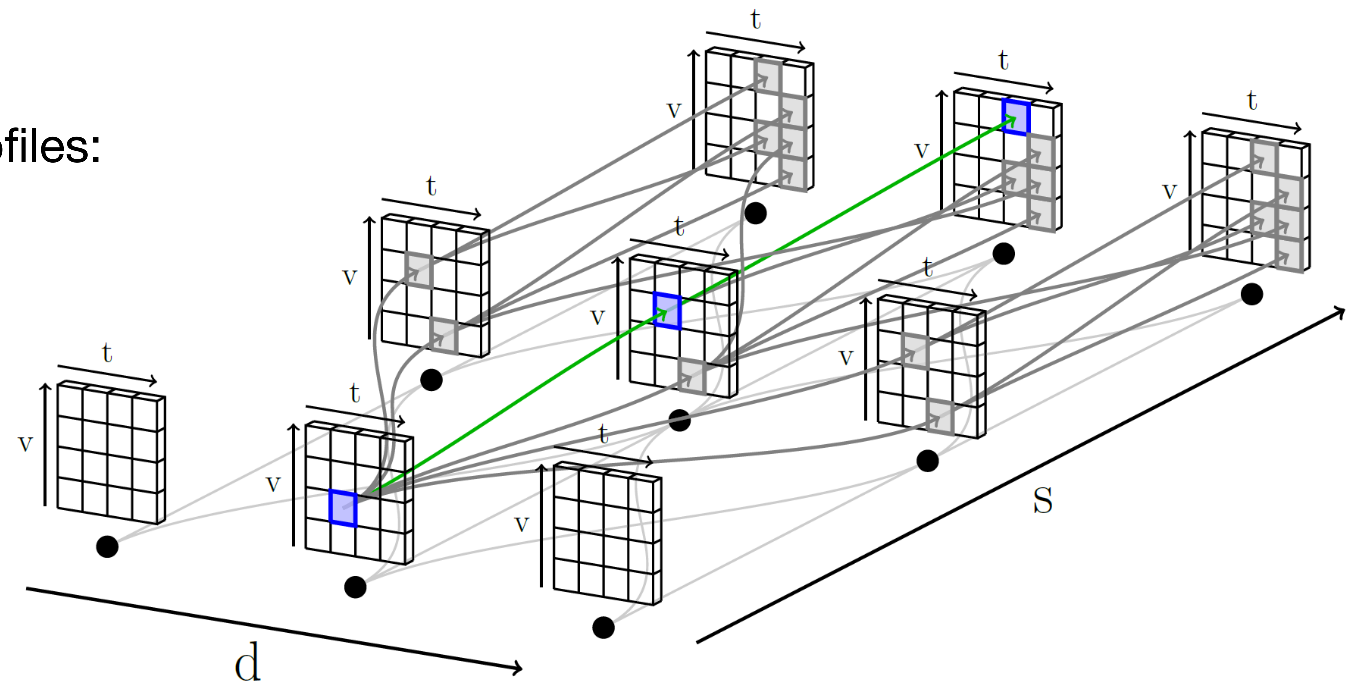
Graph-Based Planning Deep-Dive
Approach by McNaughton [1]

- Algorithm stops, when desired planning horizon is reached
  → Result is a family of possible trajectories
- Choose trajectory with lowest cumulative costs

Acceleration profiles:

$a = +1\ m/s^2$

$a = -1\ m/s^2$



[1] McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J. W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proceedings -IEEE International Conference on Robotics and Automation, 4889–4895. https://doi.org/10.1109/ICRA.2011.5980223

Additional Slides

In McNaughton approach the configuration space is discretized. In time and velocity dimension he does not use a conventional discretization and instead he introduces buckets. By that it is possible to reach any velocity to any time in the discrete spatial nodes.

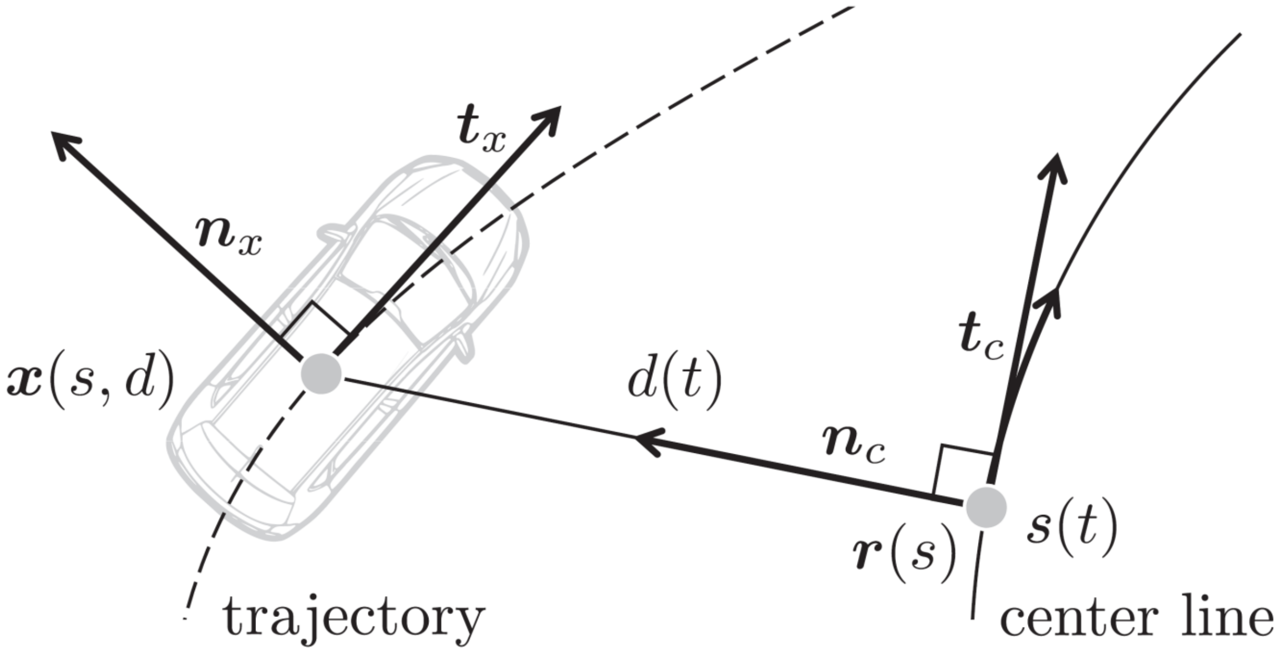
In the following the basic concept of McNaughton is described. Before the online trajectory generation, it is necessary to generate the spatial nodes and paths.

1. First the start node in the start layer has to be determined. (The real state of the vehicle will not match exactly a spatial node, so in reality it is necessary to generate an extra node. This temporally node will be connected via temporally edges with the next reachable layer.)
2. Predetermined constant acceleration profiles will be applied to every outgoing path of the start node to the next layer. The resulting trajectory edges will end in specific buckets in the next layer.
3. In order to tackle the problem of “Curse of Dimensionality”, the number of trajectory edges needs to be reduced. For that, similar edges (edges which end in the same bucket) will be compared and only the edge with best cost will be kept.
4. The steps 1 to 3 will be repeated. The start nodes in the next step will be the end nodes of the previous step.
5. After the desired planning horizon is reached, this loop will be aborted and the trajectory with the best cumulative cost will be choosen.

Graph-Based Planning Deep-Dive

Approach by Werling [1]

- Seperate generation of jerk optimal lateral $d(t)$ and longitudinal movement $s(t)$ in Frénet coordinates



[1] M. Werling „Ein neues Konzept für die Trajektoriengenerierung und –stabilisierung in zeitkritischen Verkehrsszenarien,“ in Schriftenreihe des Instituts für Angewandte Informatik / Automatisierungtechnik: Karlsruhe, 2010.

Graph-Based Planning Deep-Dive
Approach by Werling [1]

- Jerk optimal polynomials

$$J = \underbrace{h(\boldsymbol{p}(t_e), t_e)}_{\text{Minimization of endstate}} + \underbrace{\frac{1}{2}\int_{t_0}^{t_e} u^2\, dt}_{\text{Minimization of jerk}} \qquad \dot{\boldsymbol{p}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{p} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \qquad \boldsymbol{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

- Two cases:
  - $\boldsymbol{p}(t_e)$ fix and $h(\boldsymbol{p}(t_e), t_e) = 0$

    → Solution: Quintic polynomials (fifth order)

  - $p_1(t_e)$ free and $h(\boldsymbol{p}(t_e), t_e)$ is independent of p_1

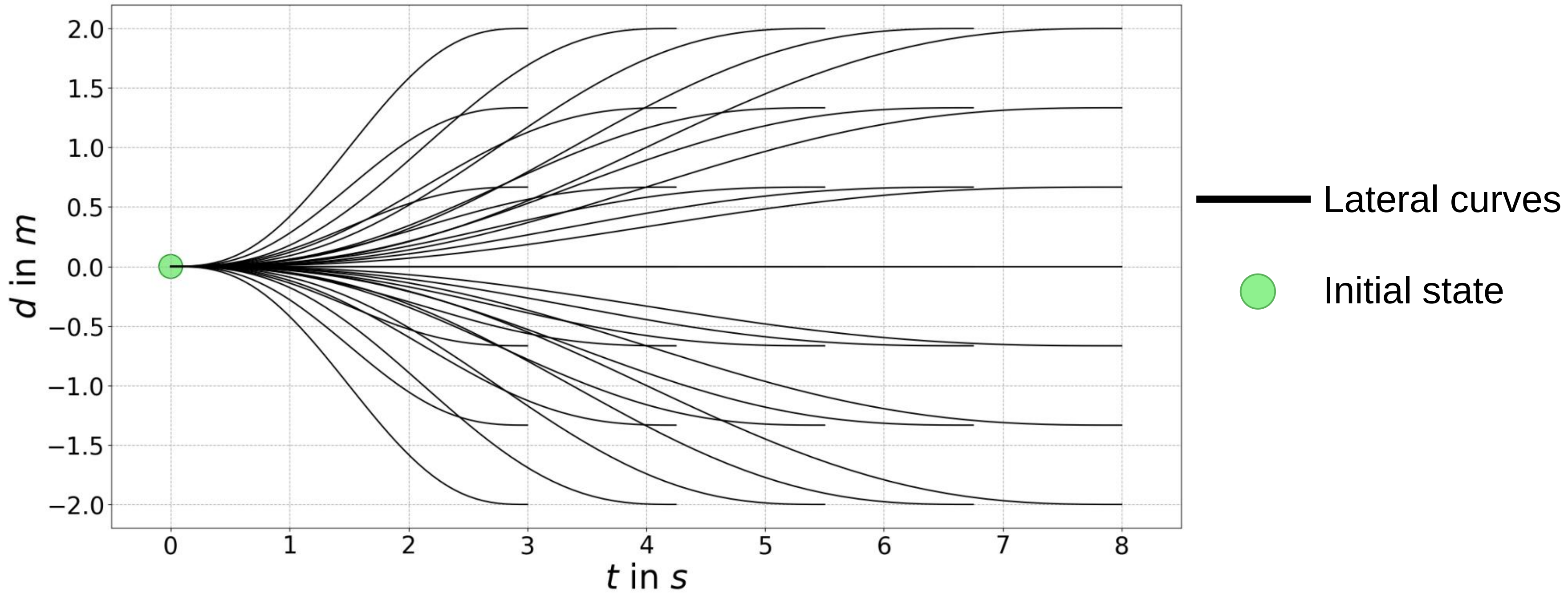
    → Solution: Quartic polynomials (fourth order)

[1] M. Werling „Ein neues Konzept für die Trajektoriengenerierung und –stabilisierung in zeitkritischen Verkehrsszenarien,“ in Schriftenreihe des Instituts für Angewandte Informatik / Automatisierungtechnik: Karlsruhe, 2010.

Graph-Based Planning Deep-Dive
Approach by Werling [1]

- Generating set of lateral curves $d(t)$
  - Variation of $d(t_e)$ and t_e
  - $\dot{d}(t_e)$ and $\ddot{d}(t_e)$ are set to 0

$\Big\}$ Quintic polynomials for $d(t)$

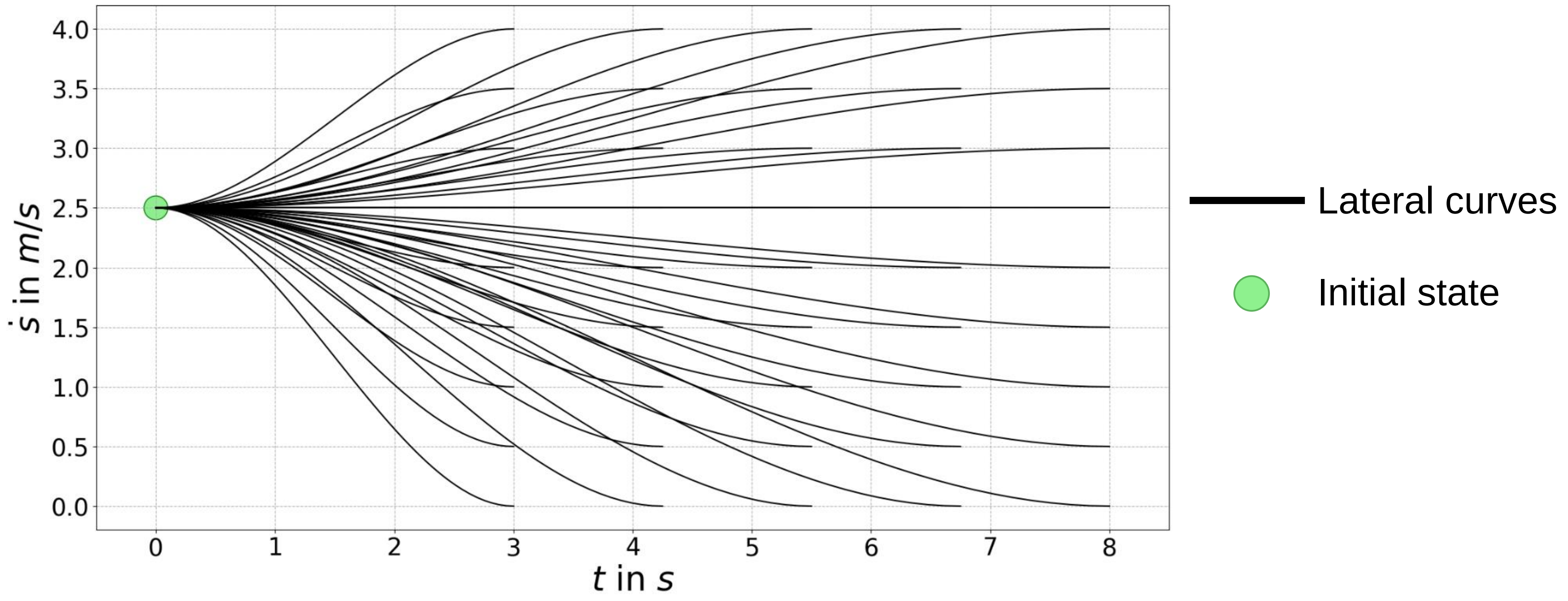


[1] M. Werling „Ein neues Konzept für die Trajektoriengenerierung und –stabilisierung in zeitkritischen Verkehrsszenarien," in Schriftenreihe des Instituts für Angewandte Informatik / Automatisierungtechnik: Karlsruhe, 2010.

Graph-Based Planning Deep-Dive
Approach by Werling [1]

- Generating set of longitudinal curves $s(t)$
  - Variation of $\dot{s}(t_e)$ and t_e
  - $s(t_e)$ is free and $\ddot{s}(t_e) = 0$

$\Big\}$ Quartic polynomials for $s(t)$

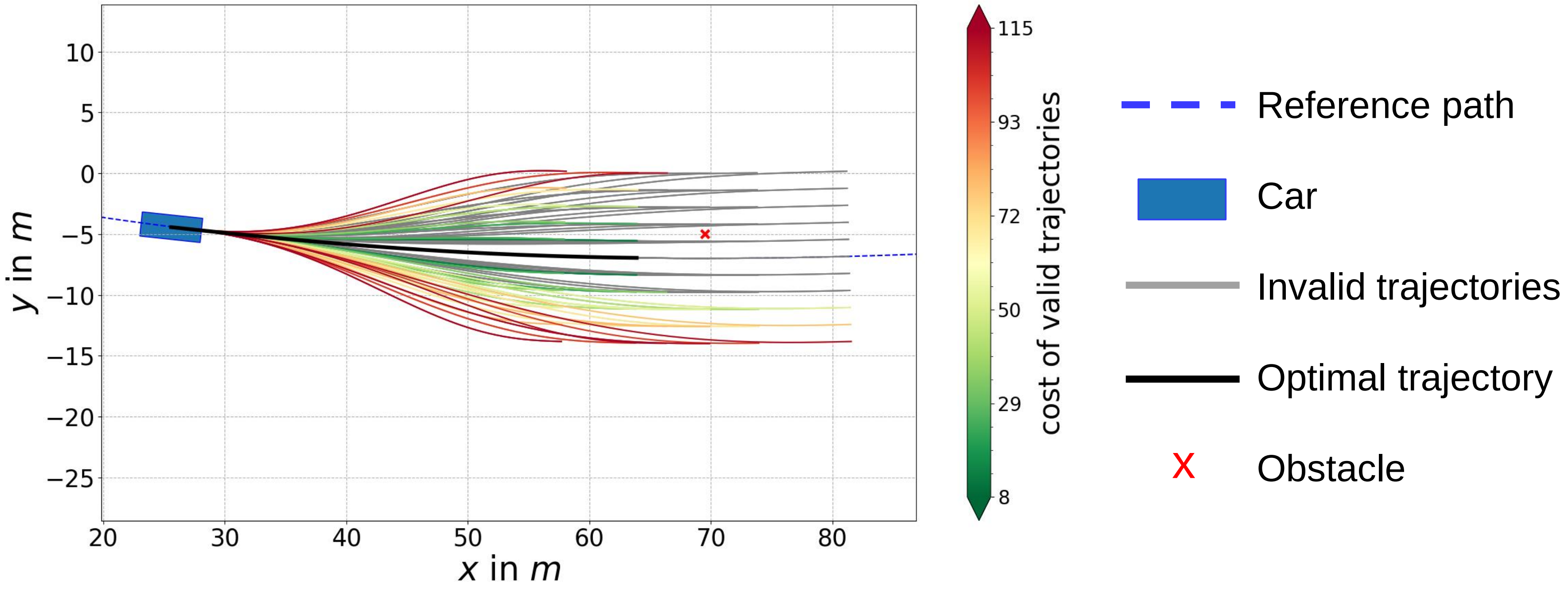


[1] M. Werling „Ein neues Konzept für die Trajektoriengenerierung und –stabilisierung in zeitkritischen Verkehrsszenarien," in Schriftenreihe des Instituts für Angewandte Informatik / Automatisierungtechnik: Karlsruhe, 2010.

Graph-Based Planning Deep-Dive

Approach by Werling [1]

- Crosswise superimpose lateral and longitudinal set
- Back-Transformation to global coordinates



- Choose cost optimal feasible and collision free trajectory

[1] M. Werling „Ein neues Konzept für die Trajektoriengenerierung und –stabilisierung in zeitkritischen Verkehrsszenarien,“ in Schriftenreihe des Instituts für Angewandte Informatik / Automatisierungtechnik: Karlsruhe, 2010.

Additional Slides

It is possible to specify the state of a vehicle $\boldsymbol{x}(s,d)$ relative to a reference line (in most cases the centerline of a lane) with the Frenet coordinates s and d. s is the arc length along the reference line and d the lateral offset to the reference line.

Wergling uses these Frenet coordinates for trajectory planning. In his approach he separately generates a set of jerk free trajectories for both, lateral and longitudinal movement. In both cases the start condition is fixed.

For the lateral movement he varies the lateral offset $\mathrm{d}(t_e)$ in the end condition and sets the first and second derivative (lateral speed and acceleration) to 0. For this case the optimal solution are quantic polynomials.

For the longitudinal movement he varies the longitudinal end speed $\dot{\mathrm{s}}(t_e)$ and sets the second derivative (longitudinal acceleration) to 0. The longitudinal end position $s(t_e)$ is free. For this case the optimal solution are quartic polynomials.

After generating that, it is necessary to combine both sets, which will result in a set of trajectories in Frenet coordinates. In order to check these trajectories for feasibility and collisions, they need to be transformed back into global coordinates. In the end the cost optimal (feasible and collision free) trajectory will be choosen.

Local Planning
Prof. Dr. Markus Lienkamp

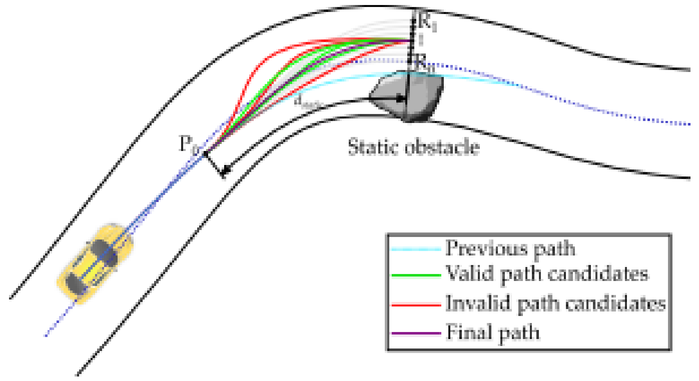
Levent Ögretmen, M. Sc.

Agenda

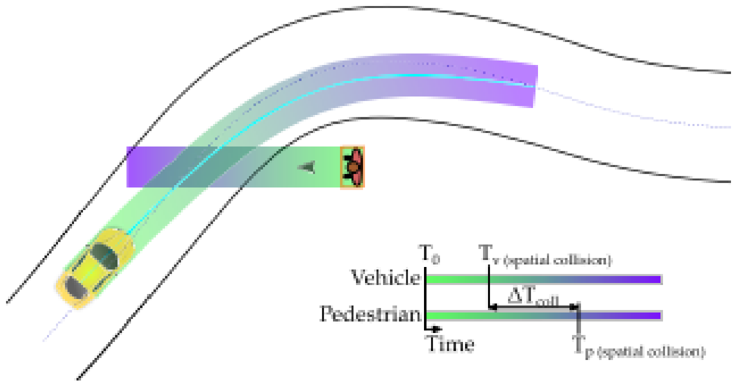
1. Introduction
2. Local Planning Methodologies
3. Deep-Dive Graph-Based Planning
4. **Collision Checking**
5. Summary

Collision Checking
Introduction

Spatial collision checking

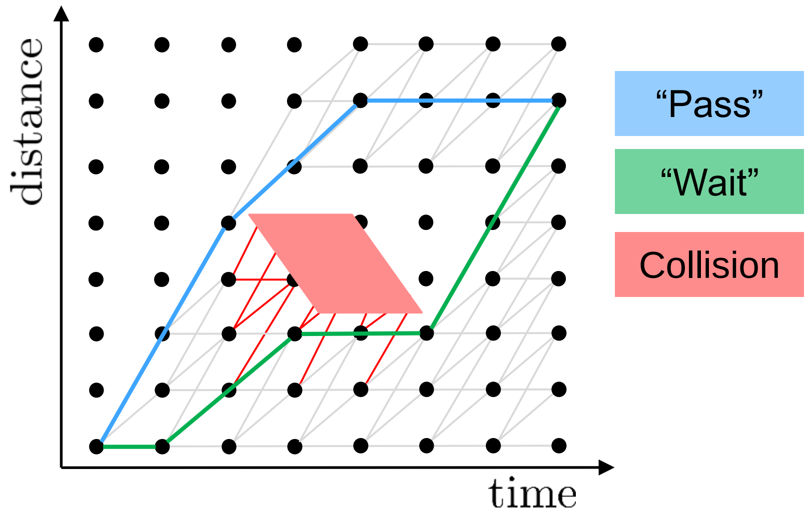


Spatio-temporal collision checking



T_0
$T_{v\text{ (spatial collision)}}$
Vehicle
ΔT_{coll}
Pedestrian
Time
$T_{p\text{ (spatial collision)}}$

- Bottleneck in search-based methods
- Dynamic obstacles require spatio-temporal collision checks

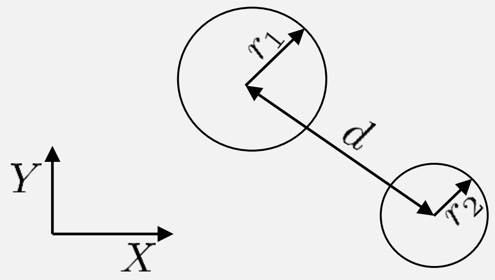


Antonio Artunedo, Jorge Villagra, and Jorge Godoy. "Real-Time Motion Planning Approach for Automated Driving in Urban Environments". In: IEEE Access 7 (2019)

Collision Checking

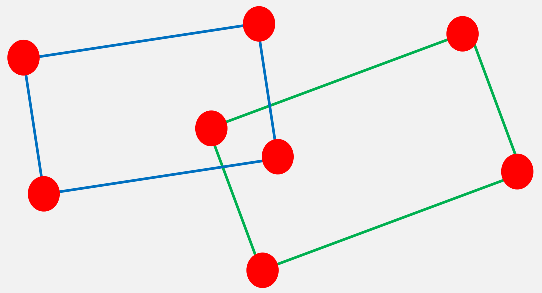
Low-Level Tests (Spatial)

Two circles

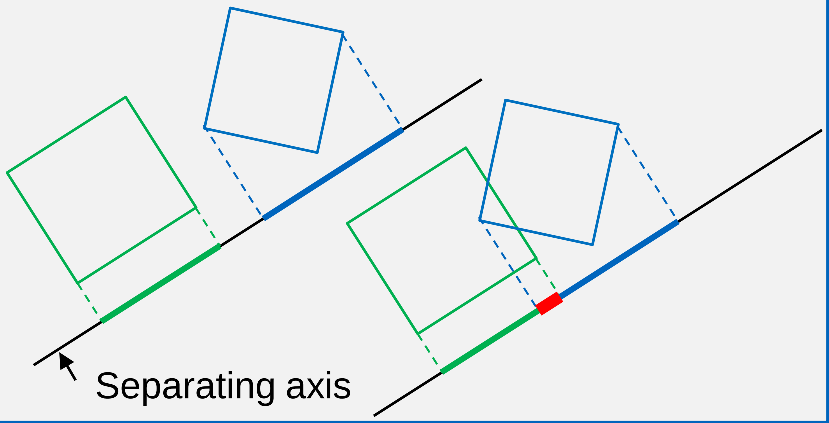


$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \leq r_1 + r_2$$

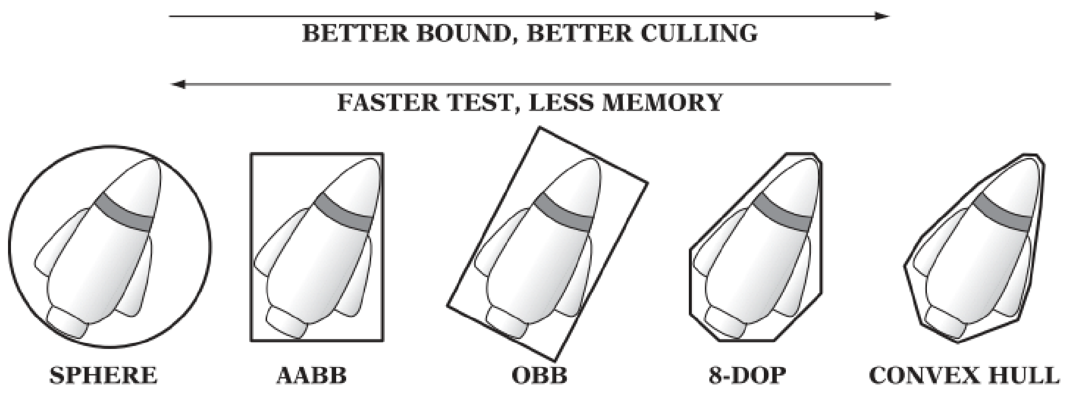
Point in polygon



Separating axis theorem

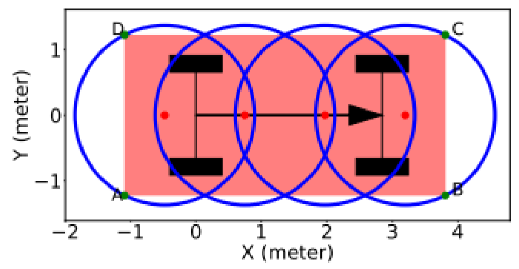


Collision Checking
Bounding Volumes (Spatial)

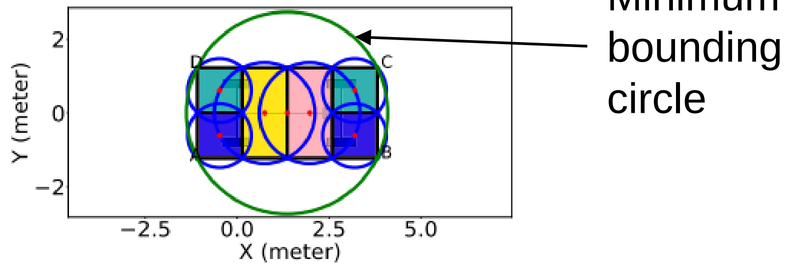


Christer Ericson. Real-Time Collision Detection. Morgan Kaufmann, 2005

Four-circle decomposition



Six-circle decomposition



Yu Zhang et al. "Hybrid Trajectory Planning for Autonomous Driving in Highly Constrained Environments". In: IEEE Access 6 (2018)

Additional Slides

Planned trajectories must be collision free to safely operate in dynamic environments. For variational methods, obstacles can be incorporated into the constraints of the optimization problem. Graph- and incremental methods however, require collision checks for each edge or branch that is possibly visited. Therefore, collision checks are often the computational bottlenecks of these algorithms. Dynamic obstacles even increase the computational effort since not only the spatial domain must be checked for collision but also the spatio-temporal domain.

Spatio-temporal collision checking approaches build upon basic tests if two convex objects overlap. This computation is simple and efficient for circles but requires more time for more complex objects. Commonly used methods utilize the *separating axis theorem* (SAT) or perform *point in polygon* (PIP) checks. The SAT says that two convex objects do not overlap if there exists an axis onto which the projections of the two objects do not overlap.

Complex objects can be over-approximated using bounding volumes to make collision checks more efficient. This however comes at the cost of accuracy regarding the culling of objects. Commonly used bounding volumes are spheres, oriented bounding boxes (OBB) and axis aligned bounding boxes (AABB).

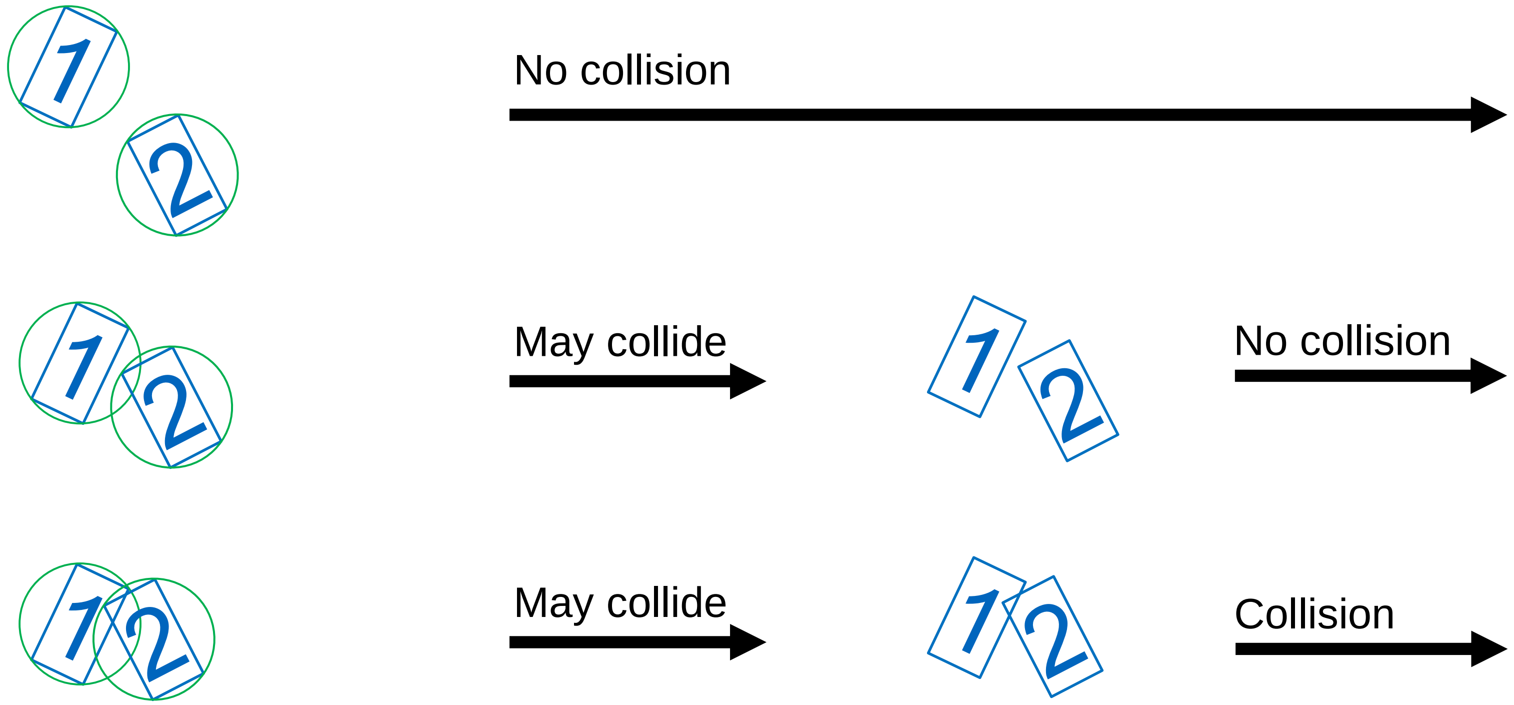
Comment:

There are planning approaches that directly search in the presented spatio-temporal diagram. In the following we only consider collision checks for already generated paths and trajectories.

Collision Checking
Hierarchies (Spatial)

Broad phase:
Pretests to reduce number of exact collision computations

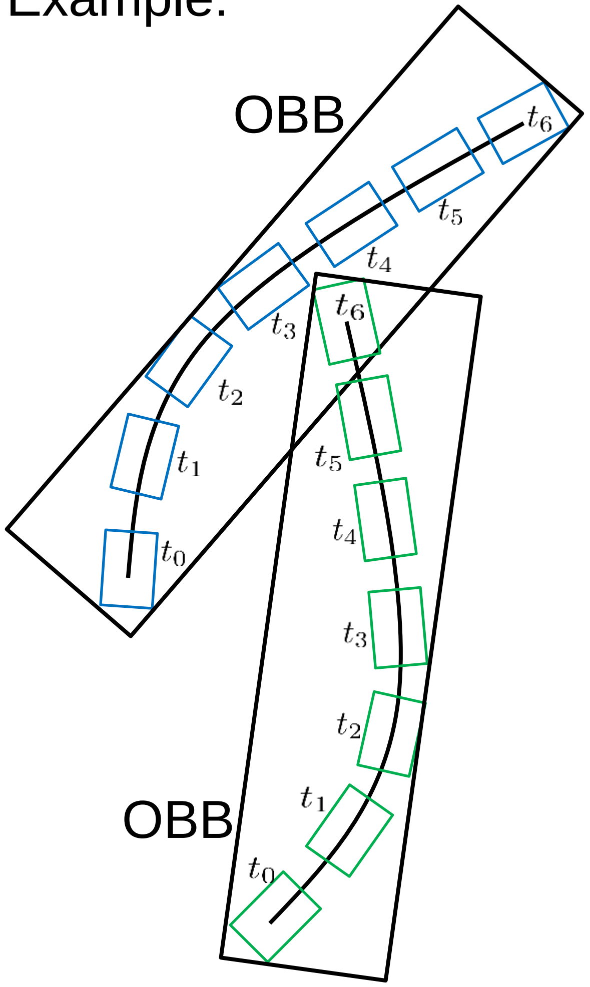
Narrow phase:
Exact collision checking

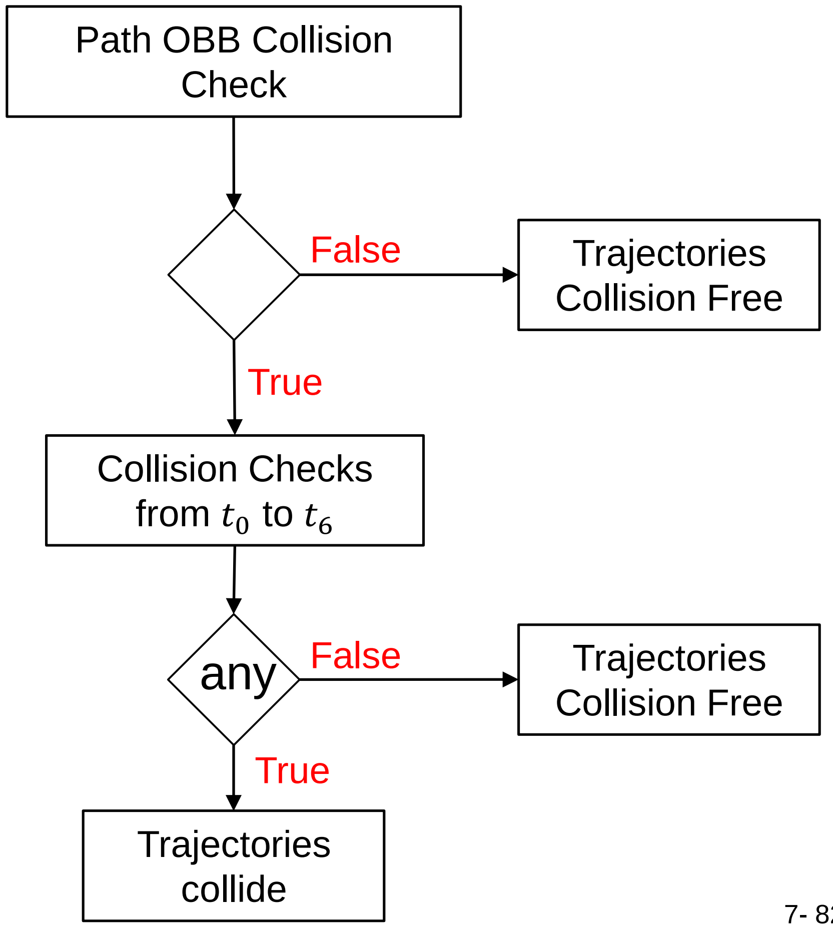


Collision Checking

Hierarchies (Spatio-temporal)

Example:

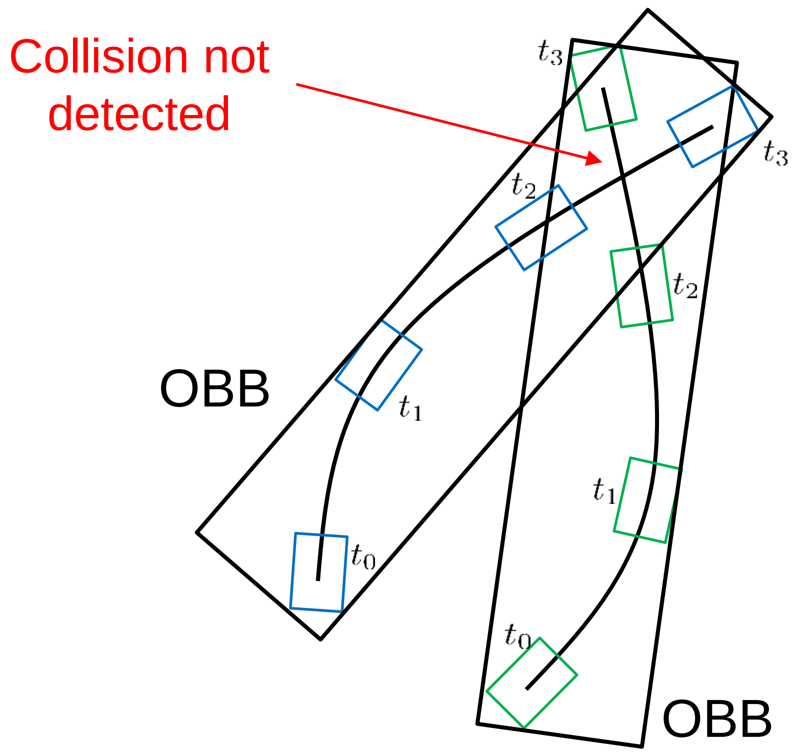




Path OBB Collision Check
False
Trajectories Collision Free
True
Collision Checks from t_0 to t_6
any
False
Trajectories Collision Free
True
Trajectories collide

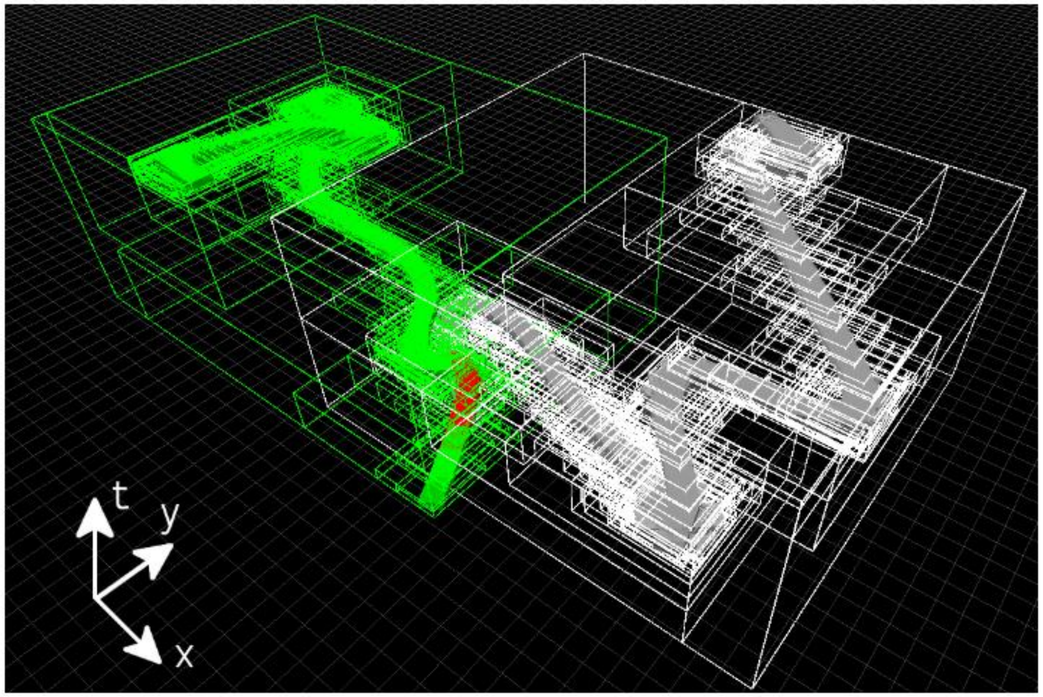
Collision Checking
Hierarchies (Spatio-temporal)

Coarse discretization results in Tunneling Problem



Collision Checking
Hierarchies (Spatio-temporal)

Bounding volumes with time as a dimension



Ulrich Schwesinger, Roland Siegwart, and Paul Furgale. “Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners.

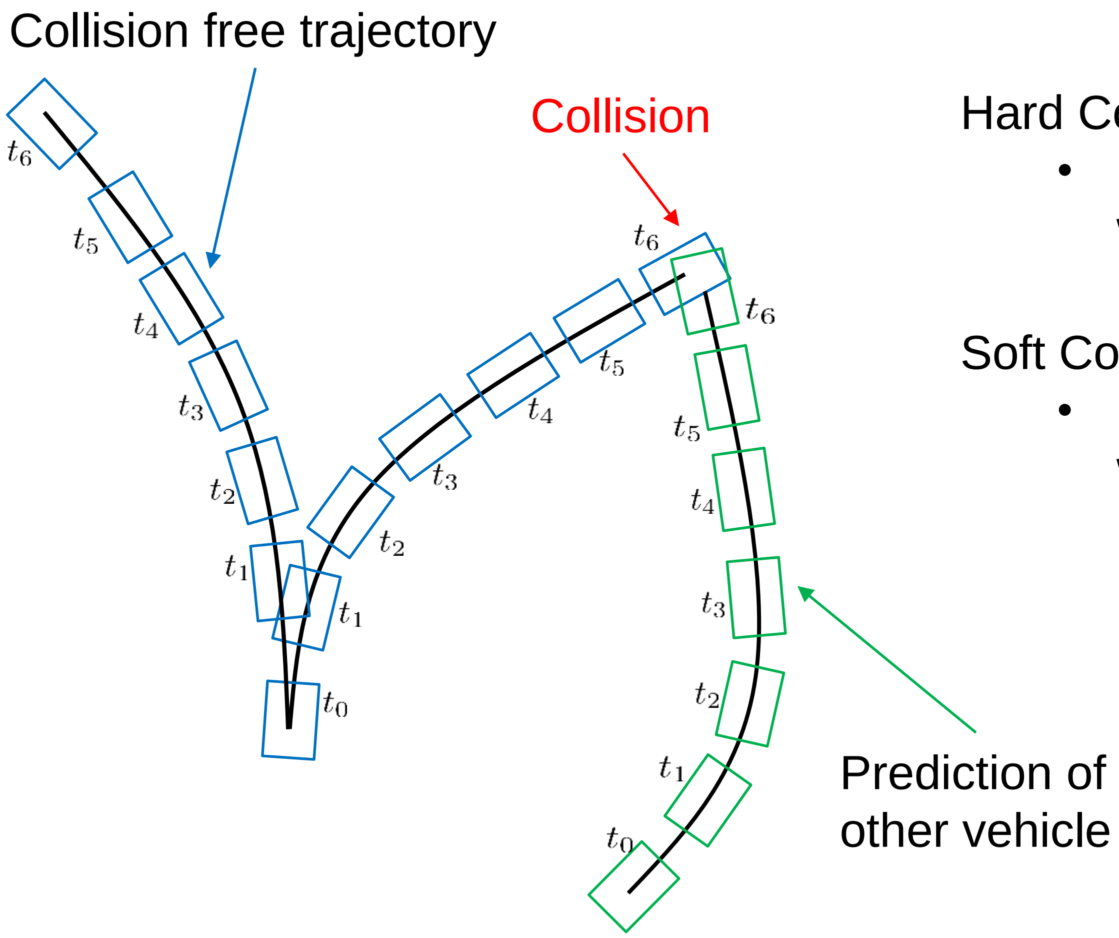
Additional Slides

Collision checks are often performed in a hierarchical manner. Before computational expensive exact collision checks it is often beneficial to first apply a fast check with a simple bounding volume like minimum bound circles. If this first check is positive, it is required to perform an exact check during a narrow phase. If the simple first check in the broad phase however is negative, we can skip the expensive narrow phase, since it is guaranteed that the two objects to not overlap.

The same principle can be applied to collision checks in the spatio-temporal domain. To check if two trajectories collide, one should first check if the paths intersect at all e.g. with OBBs. Only if this test is positive, it is necessary to include the time information. Alternatively, the time can be interpreted as a dimension of the bounding volume. The bounding volume of the entire trajectory can be separated into smaller bounding volumes of different time intervals.

Sampling at different time steps on a trajectory and pairwise collision checks with the second trajectory can lead to *tunneling* of objects. A countermeasure is to decrease the sampling time step size.

Collision Checking
Hard vs. Soft Collision Checking



Hard Collision Check

- Remove all trajectories with collisions

Soft Collision Check

- Increase cost of trajectories with collisions

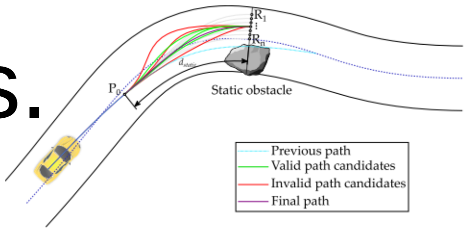
Local Planning
Prof. Dr. Markus Lienkamp

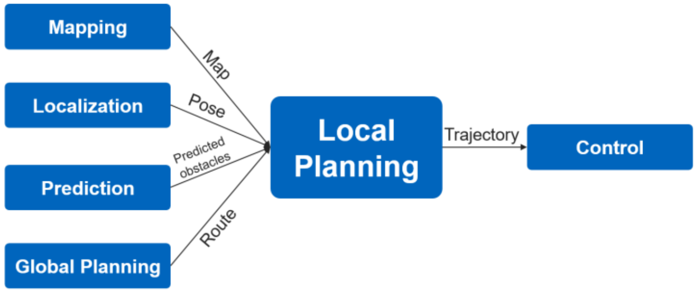
Levent Ögretmen, M. Sc.

Agenda

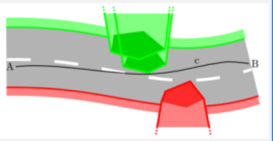
Summary – What did we learn today

VS.





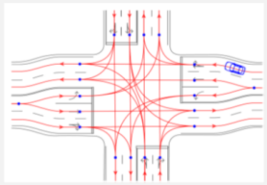
Variational Methods



$$\min J(\boldsymbol{x}(t))$$

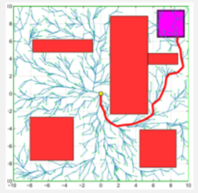
$$J(\boldsymbol{x}(t)) = \int_{t_0}^{t_0+T} L(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \dddot{\boldsymbol{x}})dt$$

Graph-Based Methods

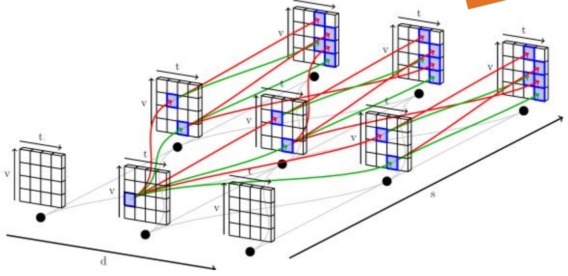


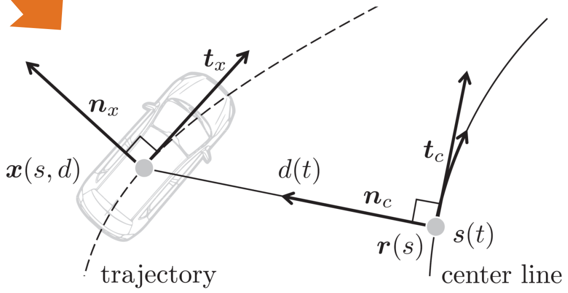
G=(V, E)

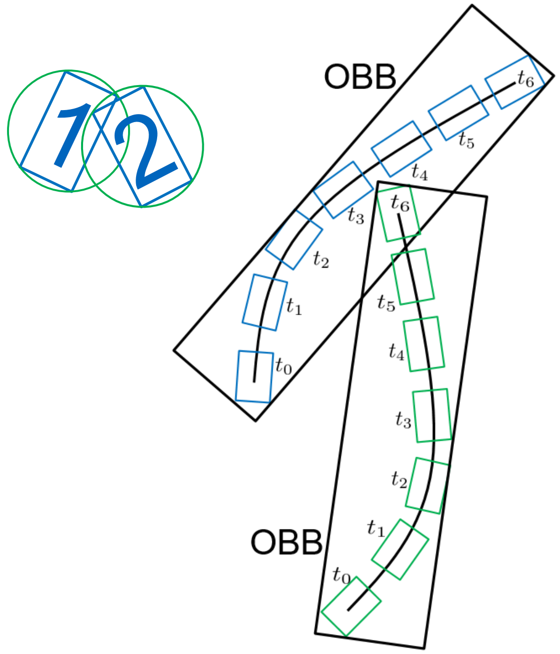
Incremental Methods



```
$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $i \leftarrow 0$;
while $i < N$ do
    $G \leftarrow (V, E)$;
    $x_{\text{rand}} \leftarrow$ Sample($i$); $i \leftarrow i + 1$;
    $(V, E) \leftarrow$ Extend($G, x_{\text{rand}}$);
```







Additional Slides

What did we learn today?

- **Distinction** between global and local planning
- **Interface** and **requirements** of local planning
- Three different **methodologies** of local planning and comparison
  1. Variational methods
  2. Graph-based methods
  3. Incremental methods
- **Trajectory planning** approaches
  - Bucket based planning by McNaughton
  - Jerk minimal planning in Frenet coordinates by Werling
- **Collision Checking** in spatial and spatio-temporal domain