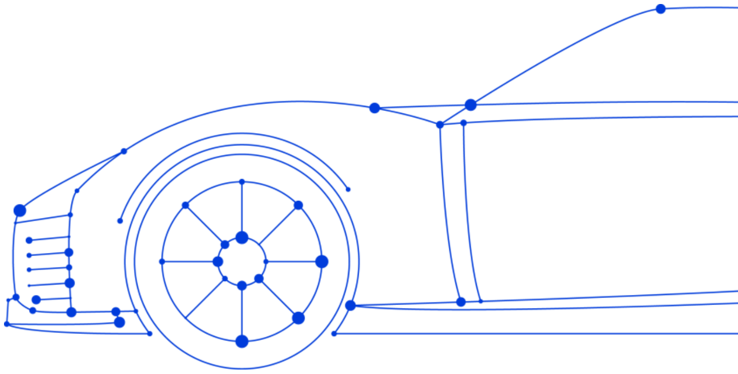
Robotics & Embedded Systems

Autonomes Fahren

VL Sommer 2019

Path Planning:

Kinematics, nonholonomic constraints



Aim

Lecture Series

- Goal of this lecture is to:
  - Learn the basic concepts and algorithm for path planning applied to autonomous cars

- Lecture 1: Kinematics, nonholonomic constraints
  - Understand the basic single track model
  - Obstacle free path planning
  - Basics of collision checking
- Lecture 2: Classical path planning algorithms

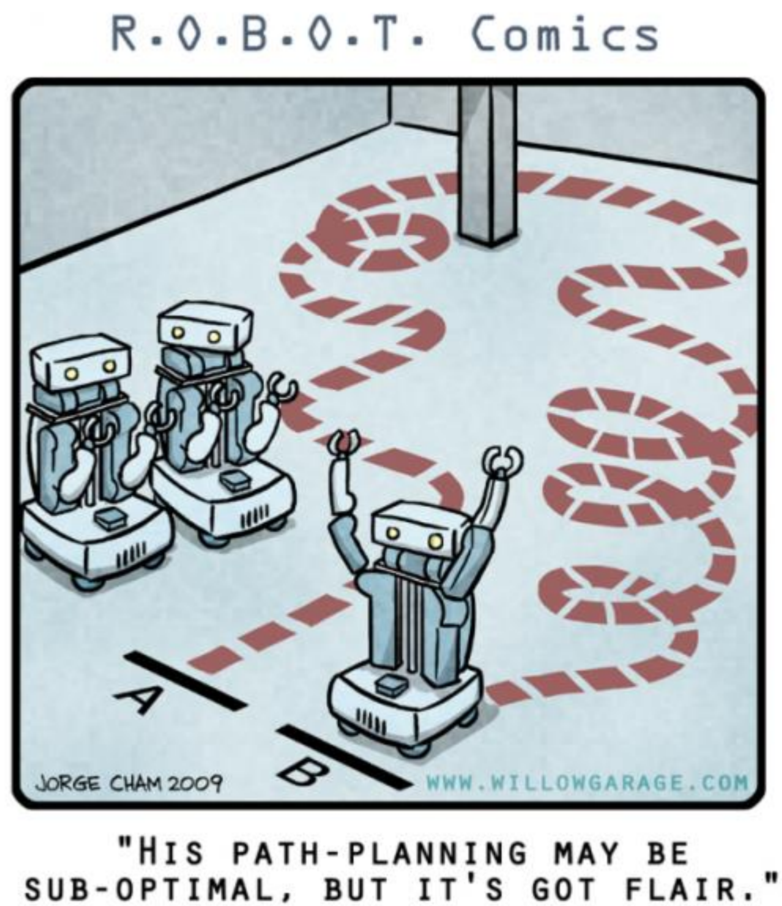
Content

- **Different planning tasks**
- Holonomic vs. nonholonomic constraints
- Single track model
- Dubins Car, metrics and cost functions
- Extending the single track model
- Coordinate Systems
- Collision Checking

Planning Tasks for Autonomous Cars

Overview

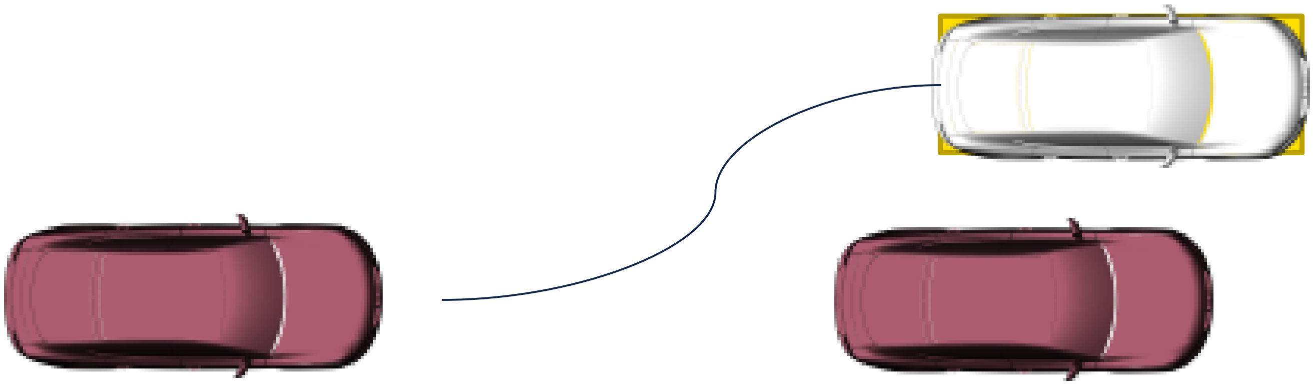
- Basic task:
  - Find path from a configuration A to a configuration B
  - Respect all imposed constraints. For example: nonholonomic constraints, continuous curvature, obstacles …



Planning Tasks for Autonomous Cars

Parking

- Maximal use of available space (minimal distance)
- Typically not time critical



Planning Tasks for Autonomous Cars

Unstructured environments

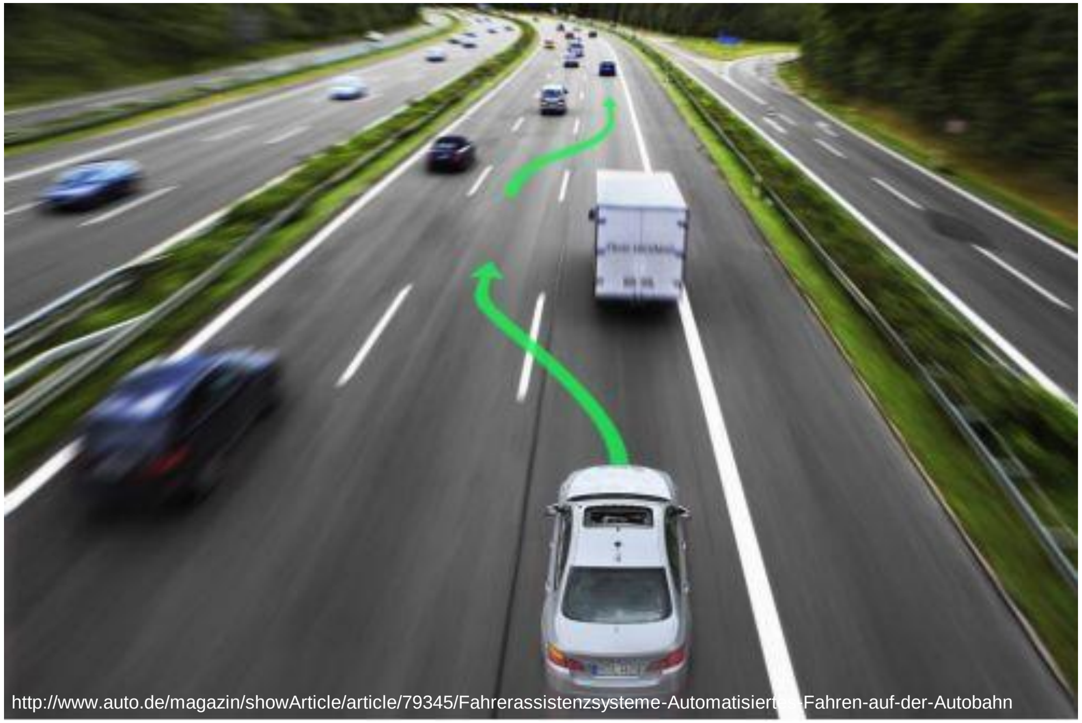
- Example: Parking lot without pre-defined paths
- Large search space of possible paths
- Mostly high distance to obstacles, but optimal path can lead through bottlenecks

[Google Maps: Parking Lot of TUM in Garching]

Planning Tasks for Autonomous Cars

Highway

- High speed
- Safety critical
- Different maneuvers
  - Lane switch
  - Following a car
  - Driving constant speed

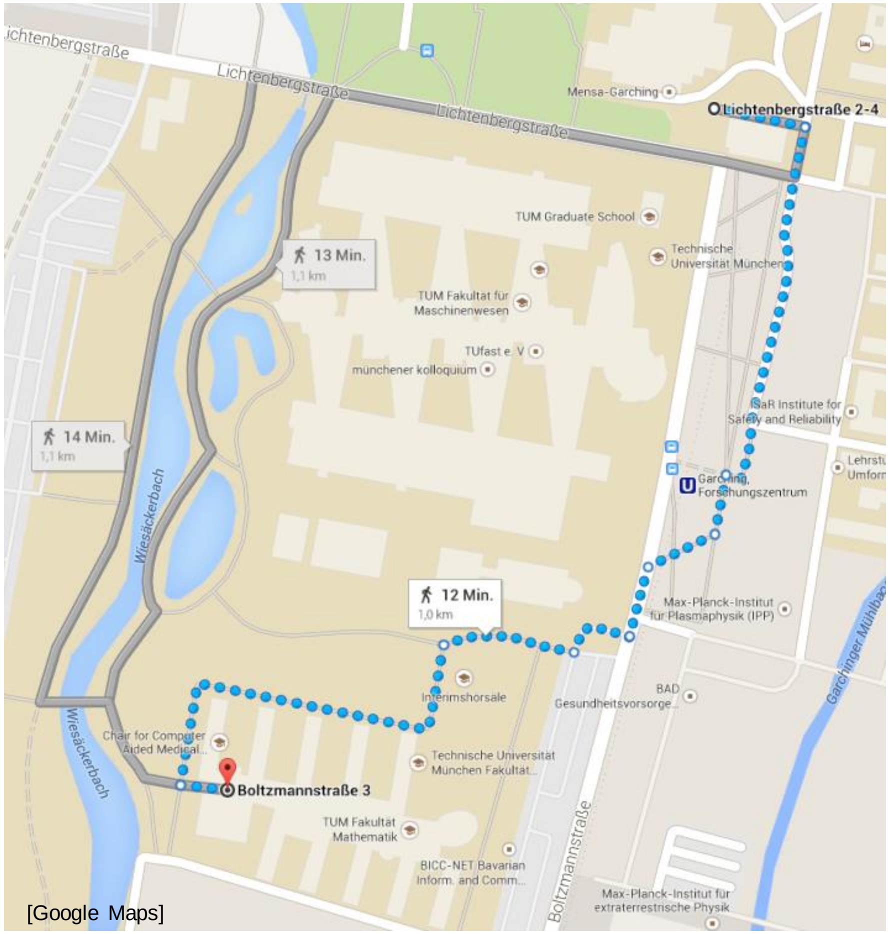


http://www.auto.de/magazin/showArticle/article/79345/Fahrerassistenzsysteme-Automatisiertes-Fahren-auf-der-Autobahn

Planning Tasks for Autonomous Cars

Route Planning

- Road abstraction
- Connecting Maneuvers:
  - Get from parking lot to driving lane
  - Driving through narrow gates
  - …



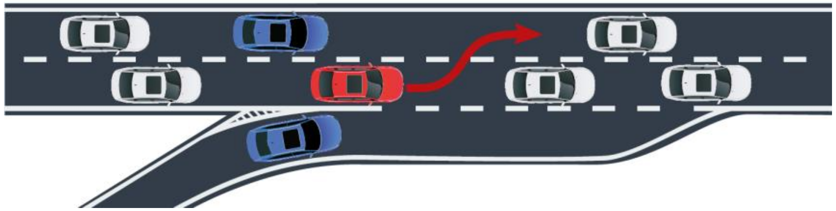
[Google Maps]

Uncertain Interactive Tactical Planning

From reactive to anticipative behavior

- In dense traffic interaction and collaboration is necessary
- Uncertainty about how other vehicles will react
- Actions of vehicle influences behavior of others

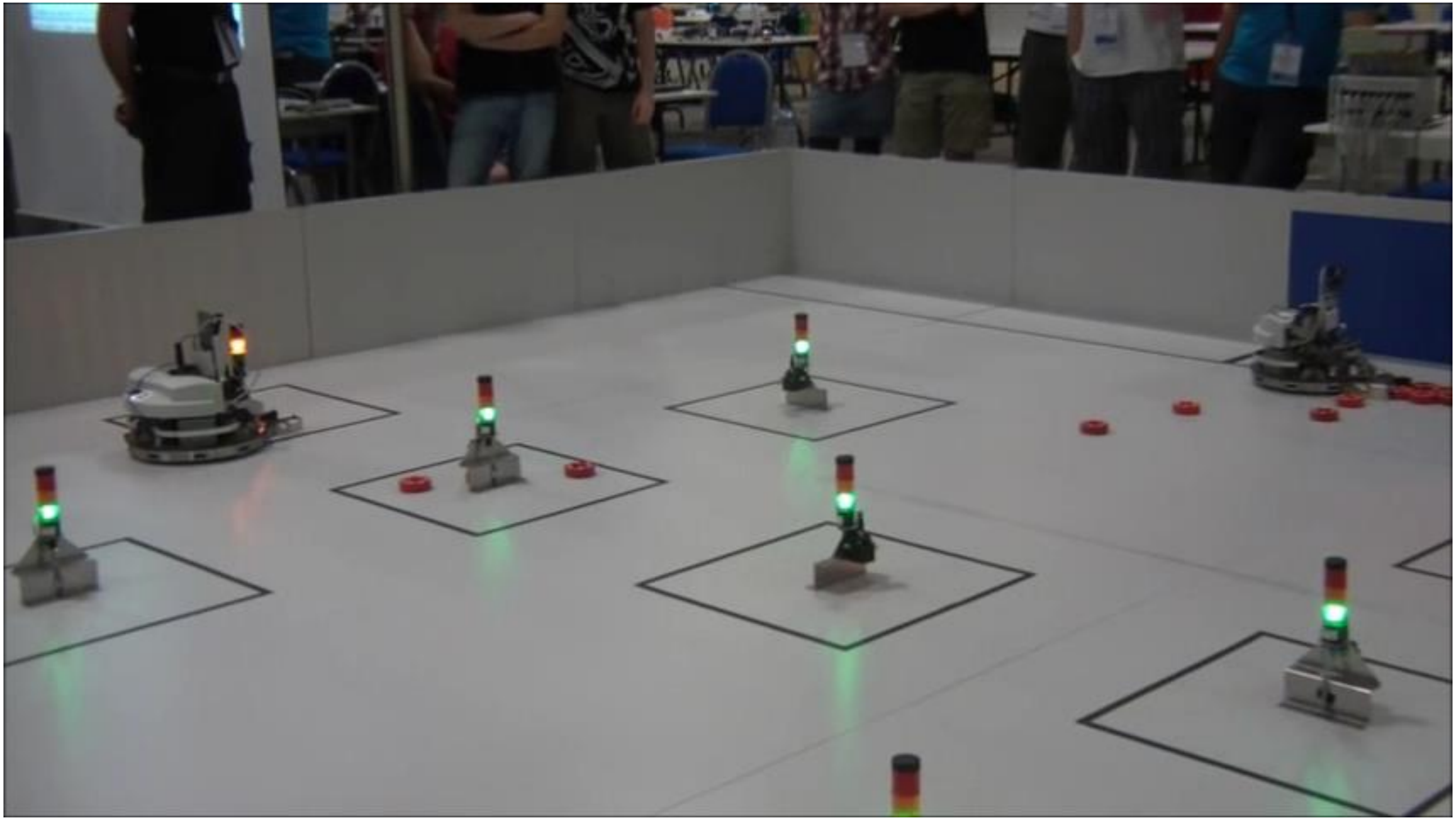
➔Goal: Find the best strategy to efficiently drive in uncertain and interactive scenarios



Content

- Different planning tasks
- **Holonomic vs. nonholonomic constraints**
- Single track model
- Dubins Car, metrics and cost functions
- Extending the single track model
- Coordinate Systems
- Collision Checking

Holonomic Robot System



[https://www.youtube.com/watch?v=WUA5CWBUy98]

Modelling a Robot System

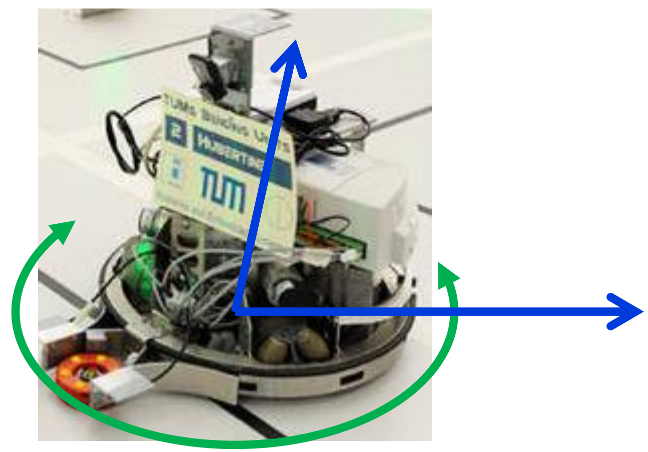
- Define which variables describe the state of the robot. For example:
  - x, y: Position of the robot
  - θ: Orientation of the robot
- Define the possible continuous transitions and the possible inputs of the robot system
  - For example:
    - $\dot{x} = u_x$
    - $\dot{y} = u_y$
    - $\dot{\theta} = u_\theta$
  - This would be a robot that can be steered in any direction

Holonomic or Nonholonomic Constraints

Holonomic Constraints

- Constraints limit the possible state transitions
- Examples for a holonomic constraint:
  - the robot can't leave the arena: $0 \leq x \leq 10 \ \wedge \ 0 \leq y \leq 10$
  - The robot can't leave the surface of a sphere: $x_1^2 + x_2^2 + x_3^2 = 1$
- The constraints can be written without using derivatives
- Any reachable configuration can be reached by a simple motion

➢ The robot can directly drive to a goal configuration



Parking with Nonholonomic constraints

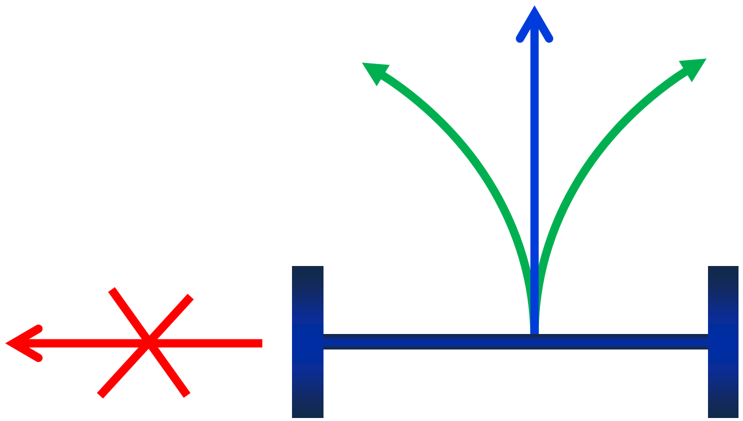
Parking without Nonholonomic constraints

Parking with Nonholonomic constraints

Holonomic or Nonholonomic Constraints

Nonholonomic Constraints

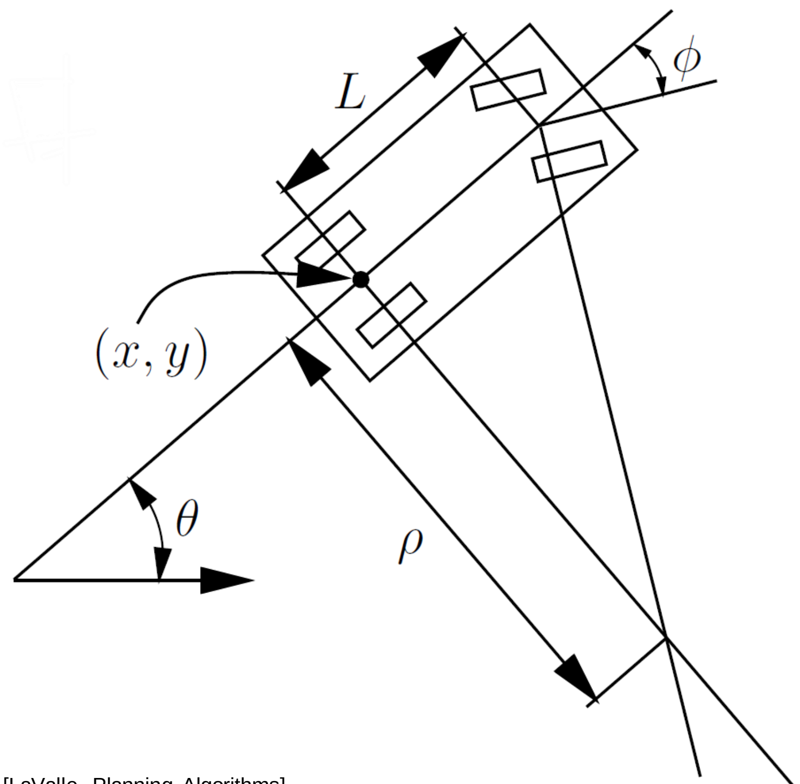
- Nonholonomic constraints may depend on the derivatives of state variables
  - A wheel may only move in one direction, for example: $\dot{y} = \sin(\theta)\,, \dot{x} = \cos(\theta)\,, \dot{\theta} = u$
  - Can’t be integrated to a representation without derivatives
  - Nonholonomic systems can reach states, using a combination of motions, which they can’t reach using simple motions



Content

- Different planning tasks
- Holonomic vs. nonholonomic constraints
- **Single track model**
- Dubins Car, metrics and cost functions
- Extending the single track model
- Coordinate Systems
- Collision Checking

Simple Single Track Model



[LaValle, Planning Algorithms]

- The simple single track model is a good approximation for low speed scenarios like parking
- Configuration: $(\mathrm{x}, \mathrm{y}, \theta)$
  - x, y: center of the rear axle
  - θ: orientation of the car
- Turning radius ρ depends on steering angle Φ and the distance between front and rear axle L:
  - ➢ $\rho = L/\tan(\Phi)$

Simple Single Track Model

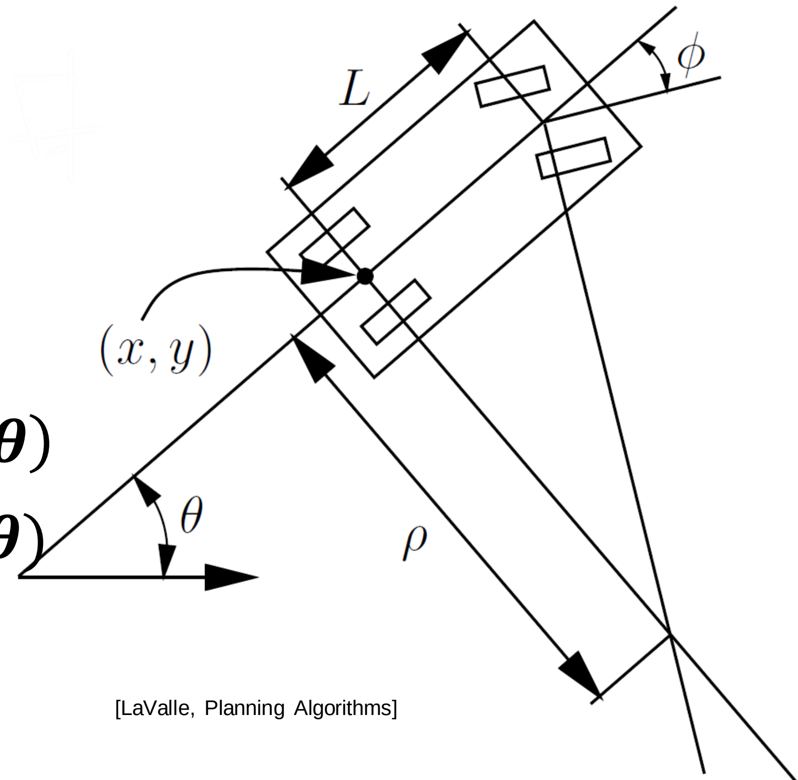
Deriving $\dot{x}$ and $\dot{y}$

- For a small Δt, the car moves approximately in the direction the rear wheels are pointing:
  - $\dfrac{dy}{dx} = \tan(\theta)$
  - $\dfrac{\dot{y}}{\dot{x}} = \dfrac{\sin(\theta)}{\cos(\theta)}$
  - Possible solution:

$$\dot{\boldsymbol{x}} = \boldsymbol{v} \cdot \mathbf{cos}(\boldsymbol{\theta})$$

$$\dot{\boldsymbol{y}} = \boldsymbol{v} \cdot \mathbf{sin}(\boldsymbol{\theta})$$

- v is the velocity of the car

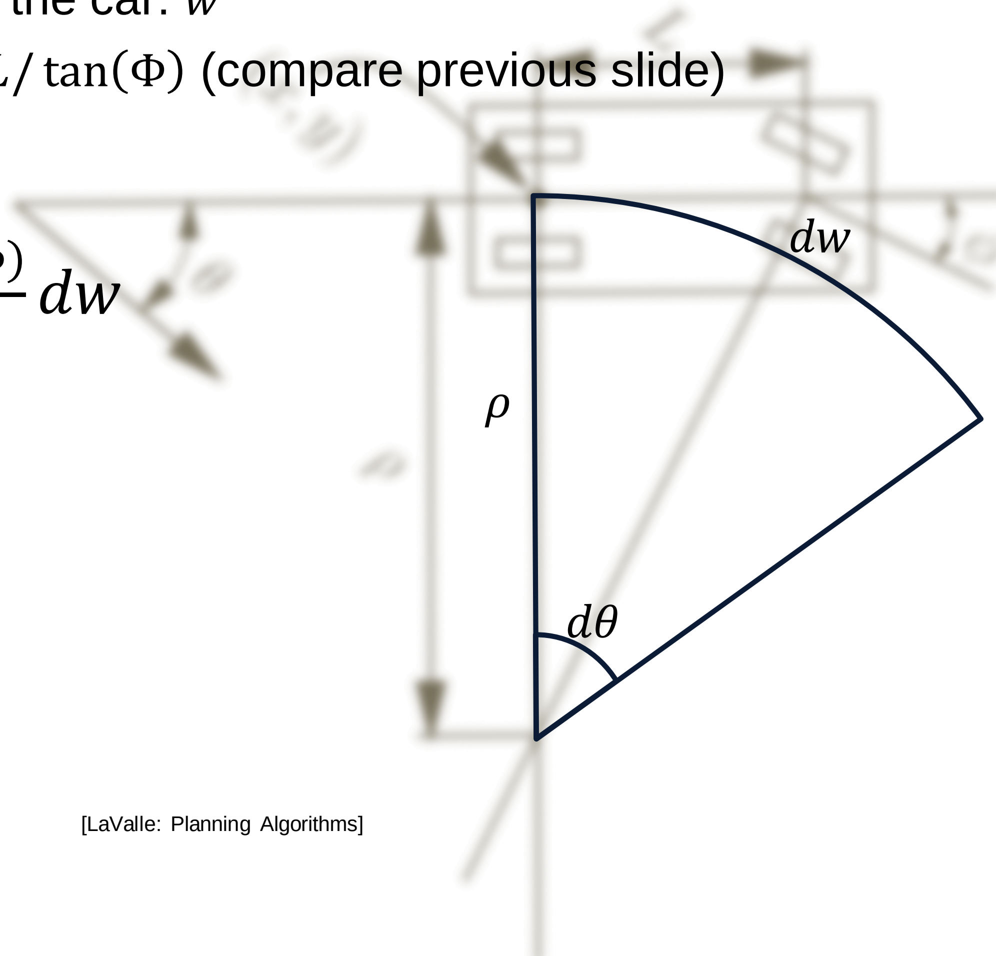


[LaValle, Planning Algorithms]

Simple Single Track Model

Deriving $\dot{\theta}$

- The orientation changes according to the circle segment covered
  - Distance traveled by the car: w
  - Turning radius: $\rho = L/\tan(\Phi)$ (compare previous slide)
  - $dw = \rho d\theta$
  - $d\theta = \frac{dw}{\rho} = \frac{\tan(\Phi)}{L} dw$
  - $\frac{d\theta}{dt} = \frac{\tan(\Phi)}{L} \frac{dw}{dt}$
  - $\boldsymbol{\dot{\theta} = \frac{tan(\Phi)}{L} v}$



[LaValle: Planning Algorithms]

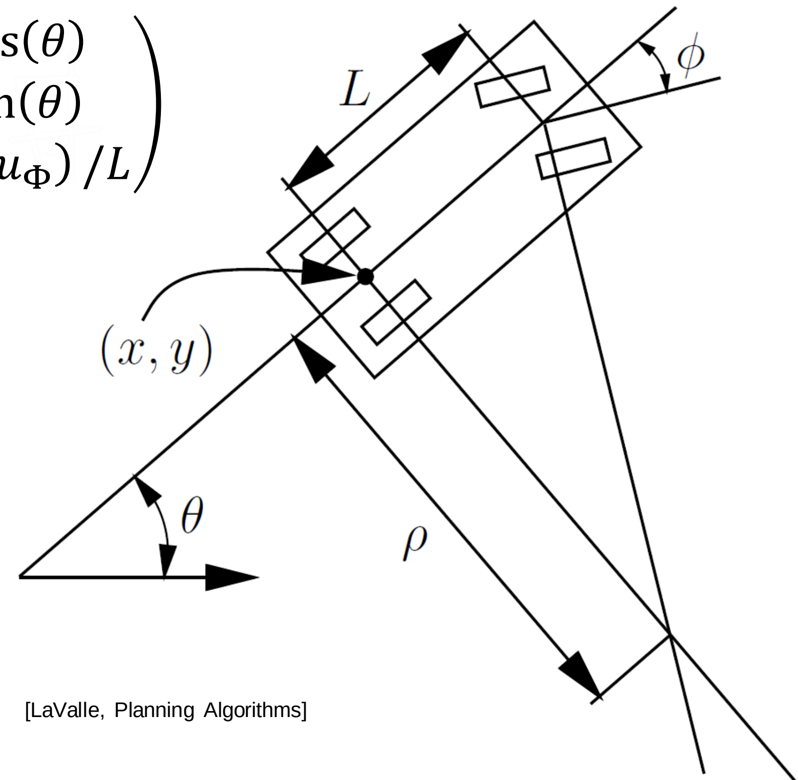
Simple Single Track Model

Specifying action variables

- Allow the car to set the velocity and steering angle directly:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} u_v \cdot \cos(\theta) \\ u_v \cdot \sin(\theta) \\ u_v \cdot \tan(u_\Phi)\,/L \end{pmatrix}$$

- u_v and u_Φ are the action variables



[LaValle, Planning Algorithms]

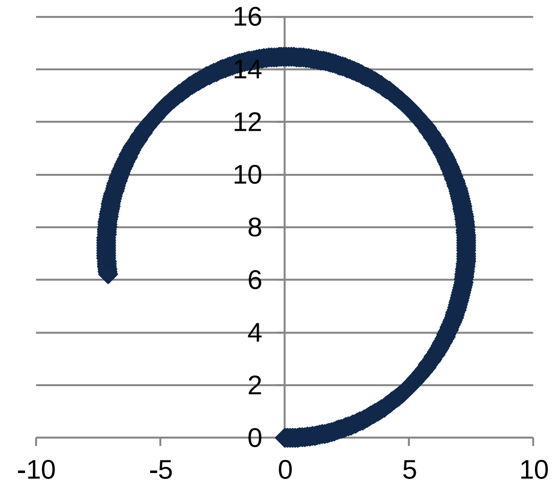
Simple Single Track Model

Example: Simulation

The Simple car equations can be used for a simple simulation program using the approximation: $\overrightarrow{x_{i+1}} = t_{step} \cdot \dot{\overrightarrow{x_i}}$ with $t_{step} = 0.1$ and $L = 3$

t	0.0	0.1	0.2	0.3	…	5	10
x	0	0,092	0,185	0,277	…	4,62	9,24
y	0	0	0,001	0,004	…	1,62	5,83
θ	0	0,014	0,028	0,041	…	0,69	1,38
u_v	1	1	1	1	…	1	1
u_Φ	π/8	π/8	π/8	π/8	…	π/8	π/8

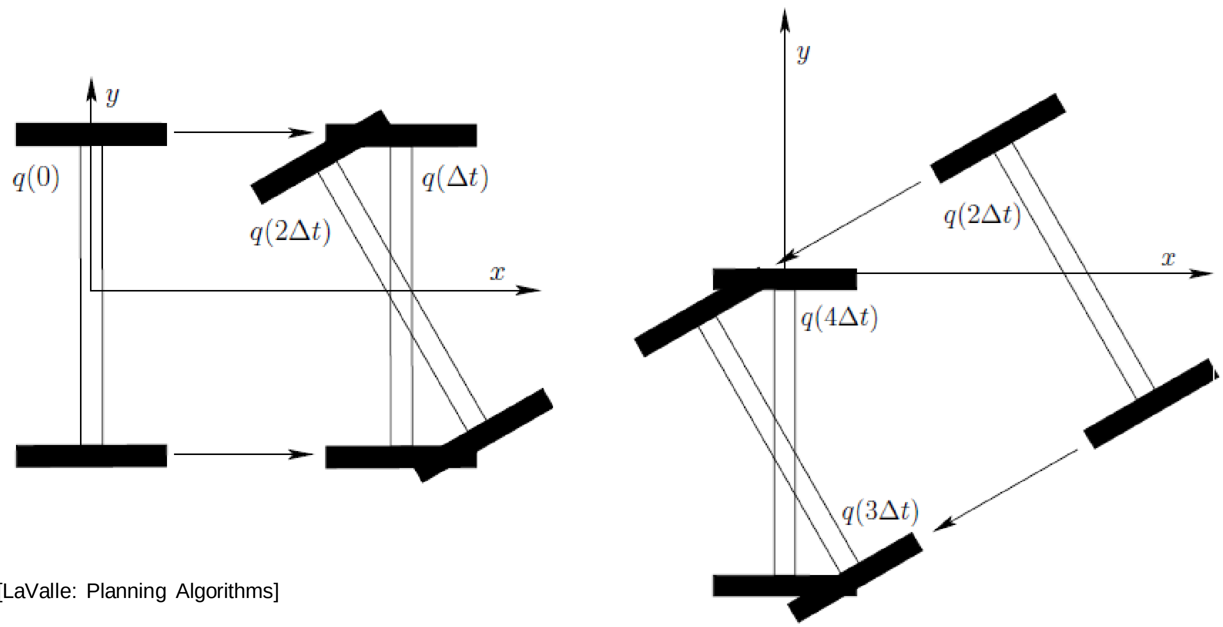
Values computed used simple car equations



Plot of x and y in Excel

Small Time Local Controllability

- A wheel can only rotate and move in the direction it is pointing
- However, using a combination of motions it can move sideways
- These motions can be arbitrarily short
  - ➢ The system is Small Time Locally Controllable



[LaValle: Planning Algorithms]

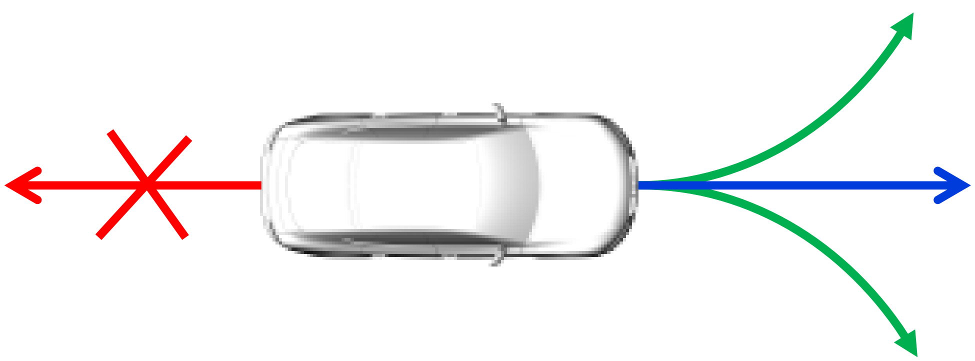
Content

- Different planning tasks
- Holonomic vs. nonholonomic constraints
- Single track model
- **Dubins Car, metrics and cost functions**
- Extending the single track model
- Coordinate Systems
- Collision Checking

Dubins Car

Optimal Path Planning

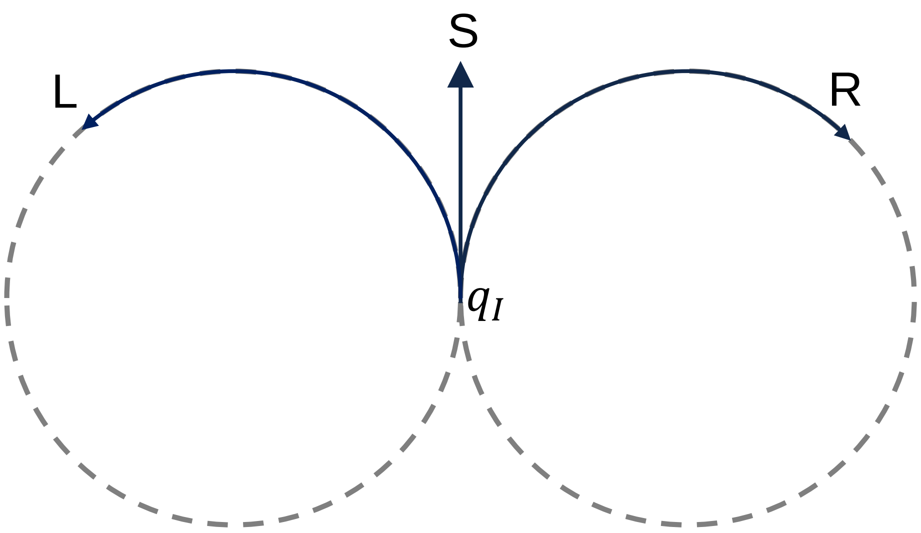
- For the Dubins Car, the action variable u_v is restricted to $u_v = 1$
  - The car can only drive forward with a constant speed
  - Only u_Φ can be changed
- Task: find the shortest path from an initial configuration $q_I = (x_I, y_I, \theta_I)$ to a goal configuration $q_G = (x_G, y_G, \theta_G)$
- The shortest path uses only $u_\Phi \in \{-\Phi_{max}; 0; \Phi_{max}\}$



Dubins Car

Three Possible Primitive Motions

- $u_\Phi \in \{-\Phi_{max}; 0; \Phi_{max}\}$ leaves only three primitive motions:
  - L:= Turn Left, R:= Turn Right, and S:= Drive Straight
- A combination of primitive motions is called word
- The shortest path can be expressed by one of the following 6 words:
  - {LRL; RLR; LSL; LSR; RSL; RSR}

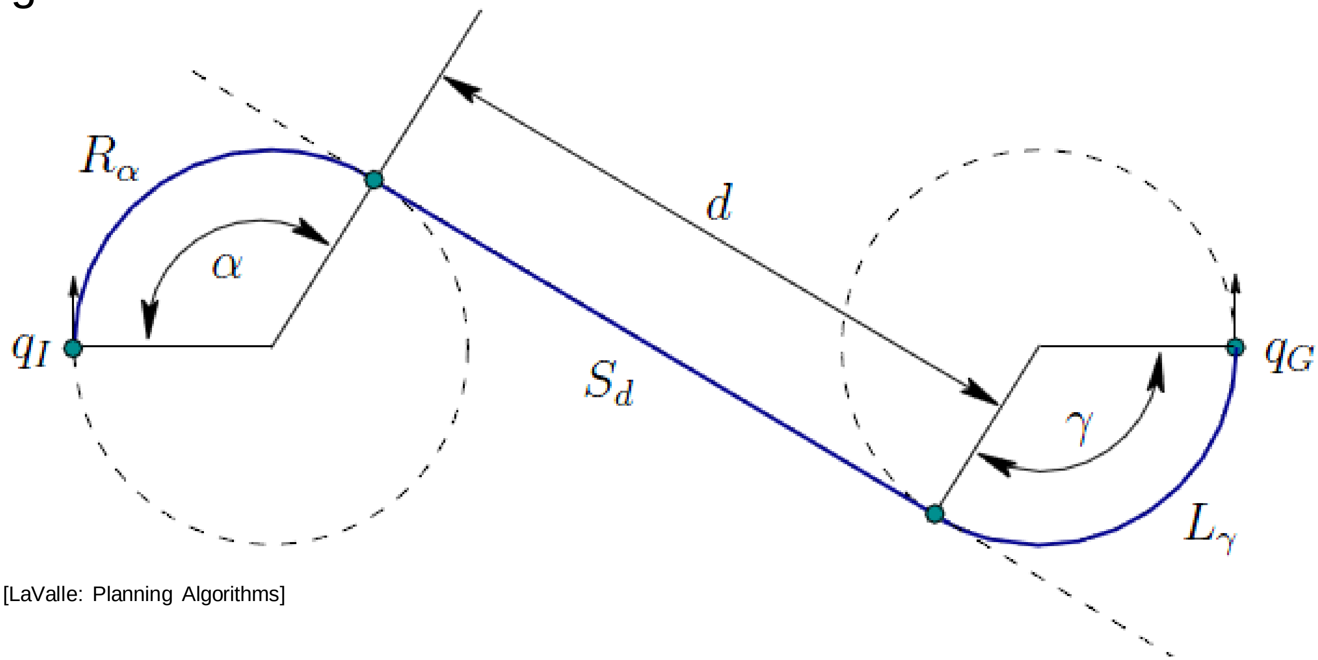


Dubins Car

Optimal Path planning

Example: Computing the path corresponding to the word RSL:

- Start with R-circle of the starting configuration q_I and L-circle of the goal configuration q_G
- Connect the circles by the S-tangent that passes both circles in the right direction



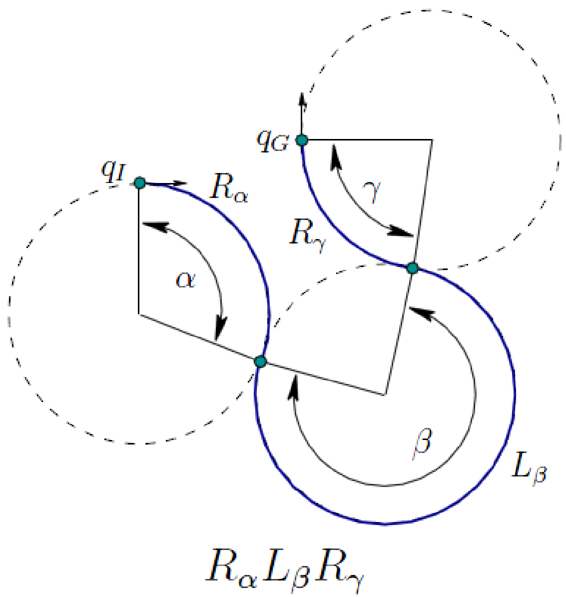
[LaValle: Planning Algorithms]

Dubins Car

Optimal Path Planning

- Example: Computing the path corresponding to the word RLR:
  - Start with R-circle of the starting configuration q_I and R-circle of the goal configuration q_G
  - Draw 2 cricles with the radius 2ρ around the centers of both R-circles. The center of the L-circle is the intersecting point of the two circles that leads to minimal length of the R-circle segments.

- Comparing all six words of primitive motions delivers the optimal path



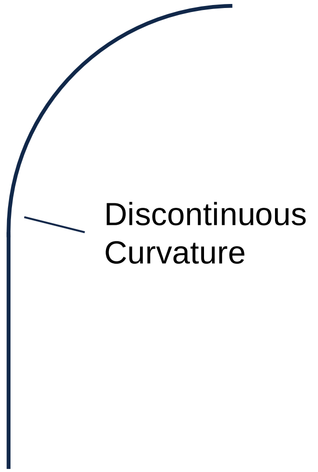
[LaValle: Planning Algorithms]

Robotics & Embedded Systems

From Dubins to Reeds and Shepp

- The Reeds and Shepp car model may additionally drive backward:
  - $u_v \in \{-1; 1\}$
- The optimal path is one of 48 different words of primitive motions

- Disadvantages of Reeds and Shepp curves:
  - No continuous curvature
  - Small position changes can lead to large differences concerning path length
    - Such position changes can be the result of sensor and actuator inaccuracies
    - It might be better to accept longer paths in order to avoid discontinuities



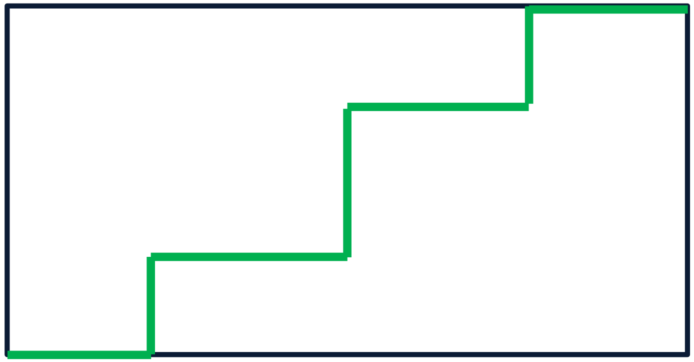
Metrics

Measuring Distance

Several planning tasks require measuring the distance of two configurations

- Decide, which path is shorter/better
- Estimate the remaining distance between two configurations

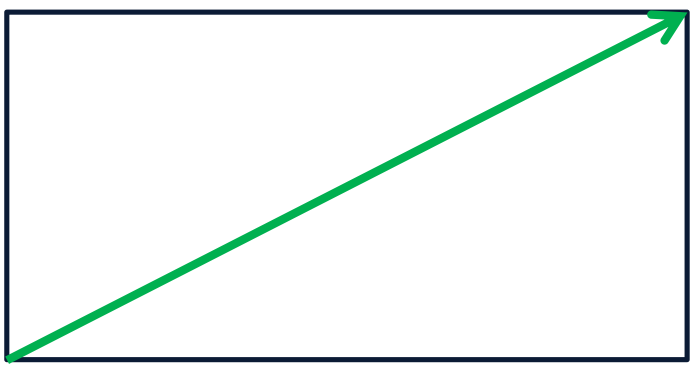
- Example: Manhatten Distance: Sum of distances for each dimension:



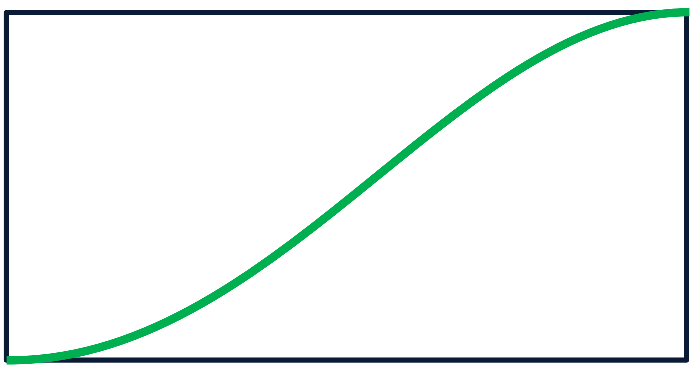
Metrics

Measuring Distance

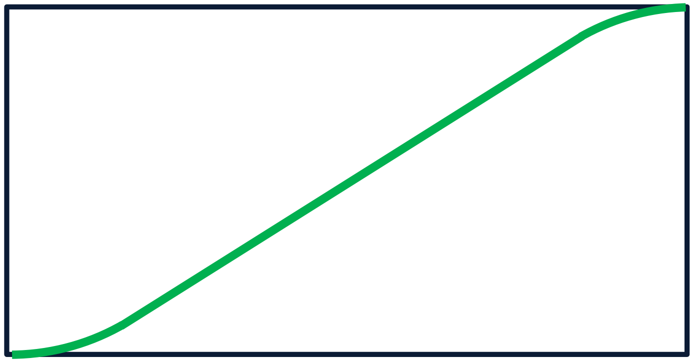
Euclidean Distance



Continuous Curvature Distance



Reeds-Shepp Distance



Cost Functions

- Finding an optimal path requires to define optimality
- A cost function maps a path or a part of a path to a usually scalar cost value
- Different cost functions can be combined to a common cost function
- Possible cost functions:
  - Distance metrics (compare previous slide)
  - Amount of steering necessary
  - Change of direction
  - Integral of longitudinal or lateral acceleration
  - Distance to obstacles
  - …

Content

- Different planning tasks
- Holonomic vs. nonholonomic constraints
- Single track model
- Dubins Car, metrics and cost functions
- **Extending the single track model**
- Coordinate Systems
- Collision Checking

Extending the Single Track Model

Dynamic Driving Situations

Extending the Single Track Model

More precise state description

- A more complex car model is necessary for planning and controlling
  - ➢ Simple single track model

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} u_v \cdot \cos(\theta) \\ u_v \cdot \sin(\theta) \\ u_v \cdot \tan(u_\Phi)\,/L \end{pmatrix}$$

- Extend the single track model:
  - Instead of the velocity we control the acceleration
  - Instead of the steering angle, we control the change of the steering angle
  - Resulting Model:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\Phi} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v \cdot \cos(\theta) \\ v \cdot \sin(\theta) \\ v \cdot \tan(\Phi)\,/L \\ u_{\dot{\Phi}} \\ u_a \end{pmatrix}$$

Extending the Single Track Model

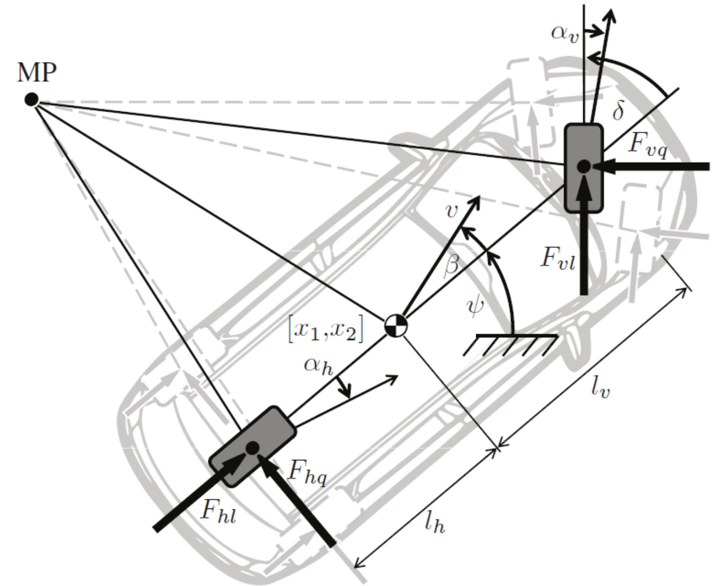
Dynamic Single Track Model

- Extending the single track model by additional dynamic information
  - Additional states: yaw rate r, slip angle β
  - Additional parameters: mass m, inertial torque J
- For planning and controlling, it should be possible to measure all state variables of the model using sensors available in the car
- For simulation purposes it can be reasonable to include additional state variables

Dynamic Single Track Model

- Complex models for medium to high velocities, can include further complex effects.
- Important is the selection of a suitable model for the specifics tasks.

$$
\boldsymbol{f} = \begin{bmatrix} v\cos(\psi+\beta) \\ v\sin(\psi+\beta) \\ r \\ \frac{-F_{hq}(\boldsymbol{x},\boldsymbol{u})l_h + F_{vq}(\boldsymbol{x},\boldsymbol{u})l_v\cos\delta + F_{vl}l_v\sin\delta}{J} \\ -r + \frac{F_{vq}(\boldsymbol{x},\boldsymbol{u})\cos(\delta-\beta) + F_{vl}\sin(\delta-\beta) + F_{hq}(\boldsymbol{x},\boldsymbol{u})\cos\beta - F_{hl}\sin\beta}{mv} \\ \frac{-F_{vq}(\boldsymbol{x},\boldsymbol{u})\sin(\delta-\beta) + F_{vl}\cos(\delta-\beta) + F_{hq}(\boldsymbol{x},\boldsymbol{u})\sin\beta + F_{hl}\cos\beta}{m} \end{bmatrix}
$$



Werling, Dissertation: Ein neues Konzept für die Trajektoriengenerierung und –stabilisierung in zeitkritischen Verkehrsszenarien, 2010

Fun with sophisticated dynamics

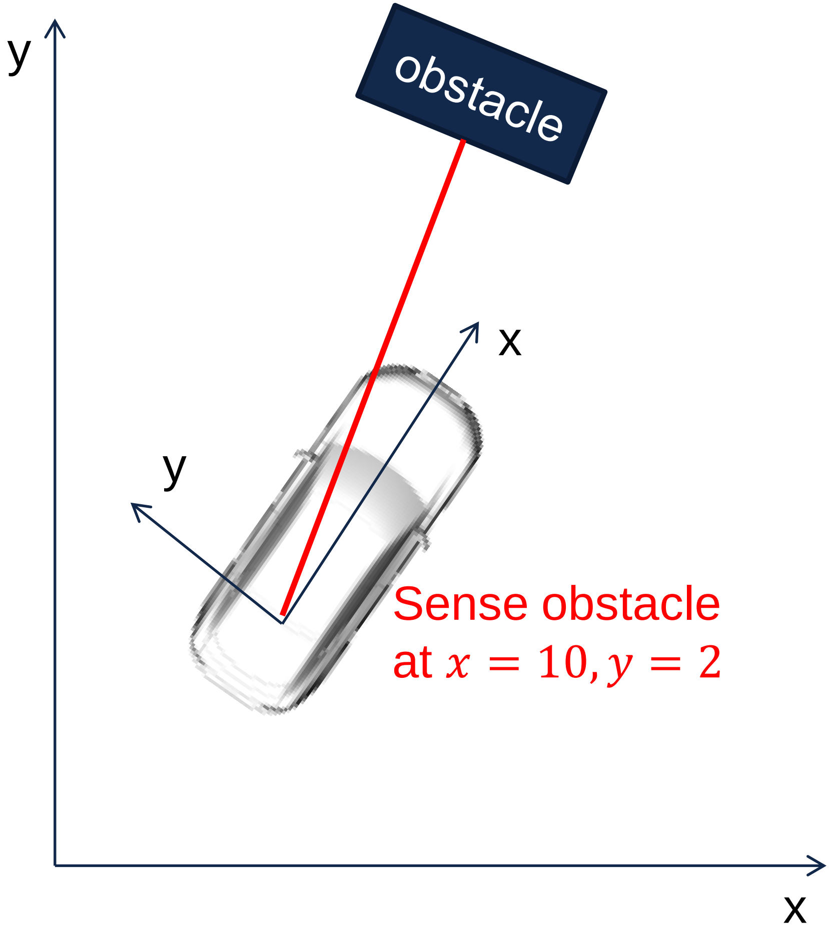
https://www.youtube.com/watch?v=gzI54rm9m1Q – Autonomous Slide Parking, Thrun et al.

Content

- Different planning tasks
- Holonomic vs. nonholonomic constraints
- Single track model
- Dubins Car, metrics and cost functions
- Extending the single track model
- **Coordinate Systems**
- Collision Checking

Coordinate Systems

- Coordinate frames should be right hand
- Information can be given in different coordinate systems
  - Planning is usually done in a stationary coordinate frame
  - Environment information is often available in a car centered coordinate frame
  - ➢ We need to transform information from one coordinate frame to another



Coordinate Systems

Affine Transformation

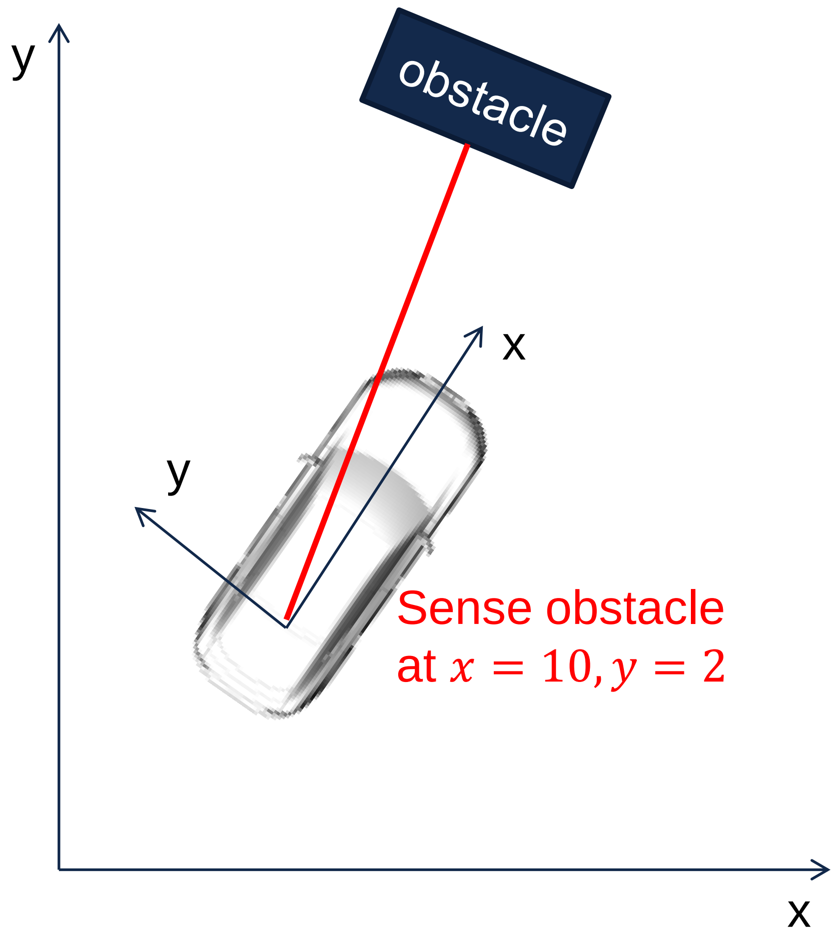
- Rotation matrix A:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

- Affine transformation, rotate by θ and translate by $\vec{t}$:

$$\overrightarrow{x'} = R_\theta \cdot \vec{x} + \vec{t}$$

- We want to rotate and translate with a single matrix T



Coordinate Systems

Homogenous Coordinates

- Affine Transformation:

$$\overrightarrow{x'} = R_\theta \cdot \vec{x} + \vec{t}$$

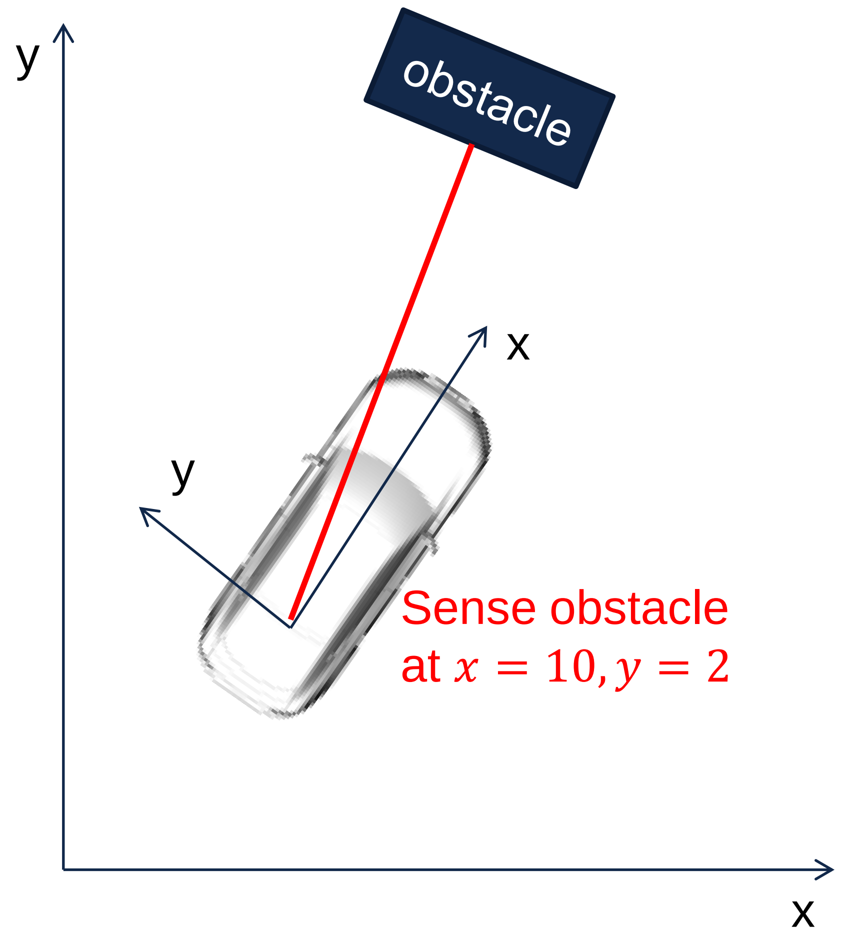
- Homogenous coordinates:

$$\overrightarrow{\mathrm{x_h}} = \begin{pmatrix} \vec{x} \\ 1 \end{pmatrix}$$

- Affine Transformation Matrix:

$$\mathrm{T} = \begin{pmatrix} R & \vec{t} \\ 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\theta & -\sin\theta & t_1 \\ \sin\theta & \cos\theta & t_2 \\ 0 & 0 & 1 \end{pmatrix}$$



Coordinate Systems

Affine Transformation

- Multiplication of the transformation matrix shows that the effect equals a rotation plus a translation:

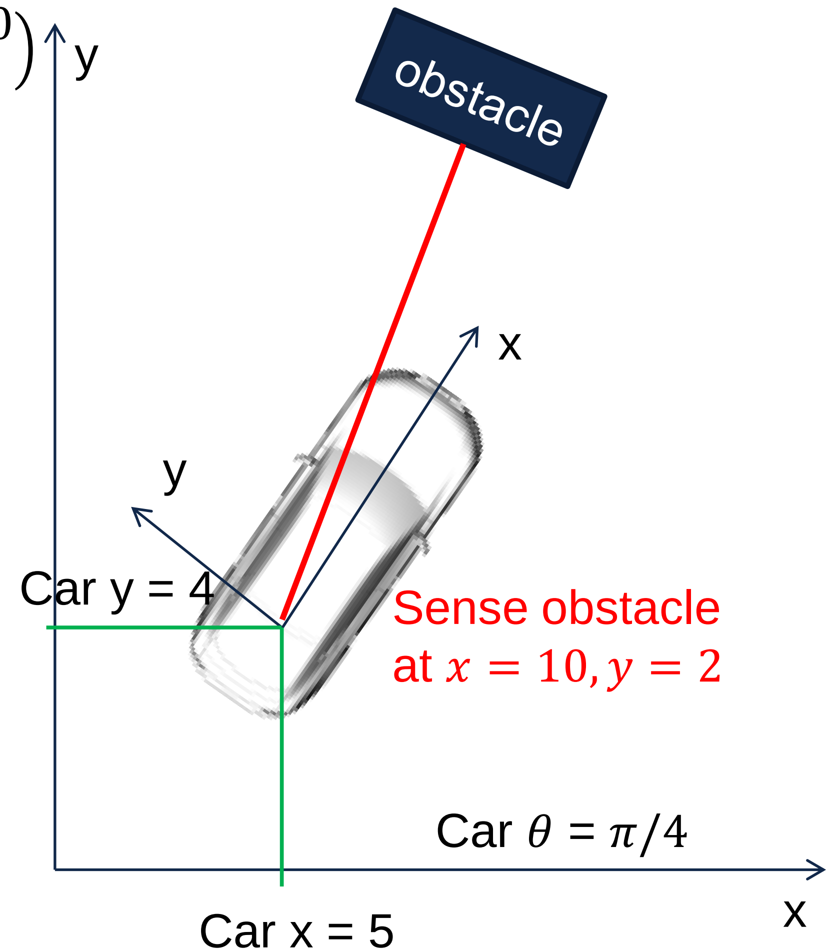
$$\overrightarrow{x_h'} = T \cdot \overrightarrow{x_h} = \begin{pmatrix} \cos\theta & -\sin\theta & t_1 \\ \sin\theta & \cos\theta & t_2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot x - \sin\theta \cdot y + t_1 \\ \sin\theta \cdot x + \cos\theta \cdot y + t_2 \\ 1 \end{pmatrix}$$

- Multiple Affine Transformations can be chained: $\overrightarrow{x_h'} = T_1 \cdot T_2 \cdot \overrightarrow{x_h}$
- For example: Transform from a coordinate system given relative to the car to the world coordinate system

Coordinate Systems
Example Affine Transformation

- Transform obstacle coordinates $\mathrm{obs} = \begin{pmatrix}10\\2\end{pmatrix}$ to world coordinate frame:
  - Rotate by $\theta = \pi/4$
  - Translate by $\vec{t} = \begin{pmatrix}5\\4\end{pmatrix}$
  - $T = \begin{pmatrix}\cos\theta & -\sin\theta & t_1\\ \sin\theta & \cos\theta & t_2\\ 0 & 0 & 1\end{pmatrix}$
  - $T \cdot obs_h = \begin{pmatrix}\cos\frac{\pi}{4} & -\sin\frac{\pi}{4} & 5\\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 4\\ 0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}10\\2\\1\end{pmatrix} =$

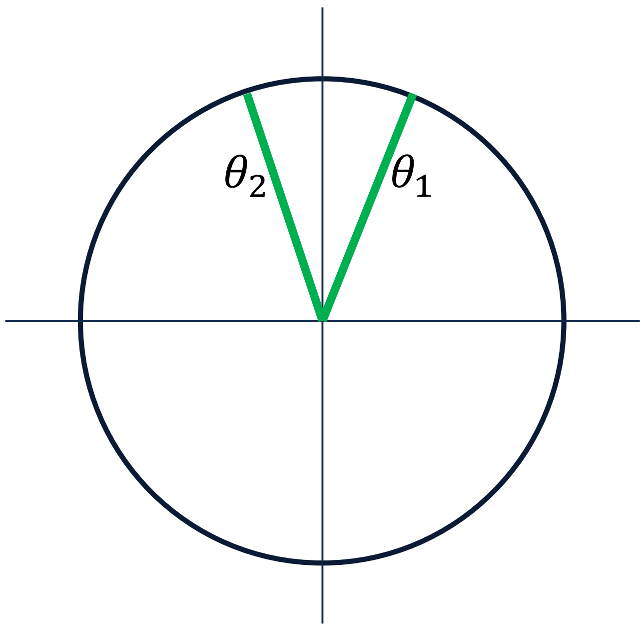
$$\begin{pmatrix}10.6569\\12.4853\\1\end{pmatrix}$$



Coordinate Systems

Representing angles

- Interpolation between to angles θ_1 and θ_2 can't be done using $\theta_{int} = \frac{\theta_1+\theta_2}{2}$, as for $\theta_1 = 0.1$ and $\theta_2 = 2\pi - 0.1$ the result would be $\theta_{int} = \pi$



Content

- Different planning tasks
- Holonomic vs. nonholonomic constraints
- Single track model
- Dubins Car, metrics and cost functions
- Extending the single track model
- Coordinate Systems
- **Collision Checking**

Collision Checking

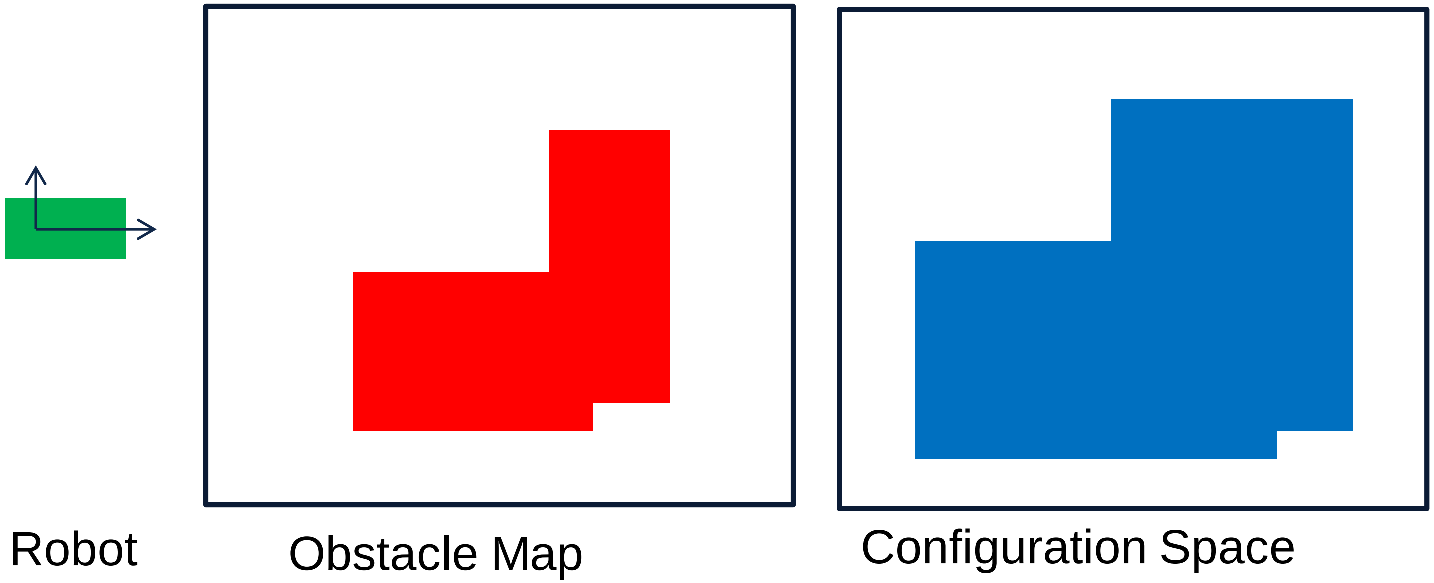
Overview

- Possible tasks:
  - Check, whether a robot in a configuration q intersects with an obstacle
  - Find the distance from a robot in a configuration q to the closest obstacle
- The obstacles and the robot can for example be represented as
  - Geometric primitives such as polygons or circles
    - Environment is given as a list of separate objects
    - Each object can be represented as a polygon
  - A grid map

[Source: http://www.carinsurancecomparison.com/when-should-i-drop-my-collision-coverage-from-my-car]

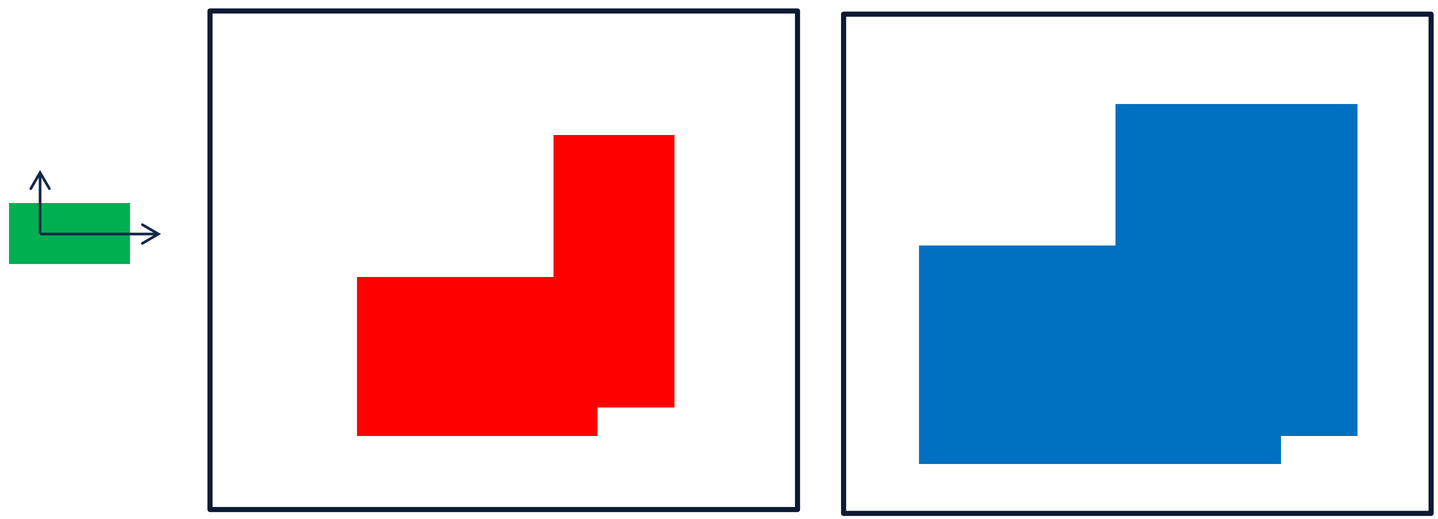
Configuration Space

- The configuration space contains all possible configurations of the robot
- Obstacles can be represented in the configuration space
  – Here constructed with **Minkowski-Sum**: $A + B \coloneqq \{a + b | a \in A, b \in B\}$
  – Example: Configuration space of a holonomic robot that can move in x and y direction



Configuration Space

- Collision check between the robot and the obstacles equals check, if configuration space is occupied at a configuration
- The configuration space becomes more complex when adding θ as a third dimension
- Path planning includes exploring the configuration space

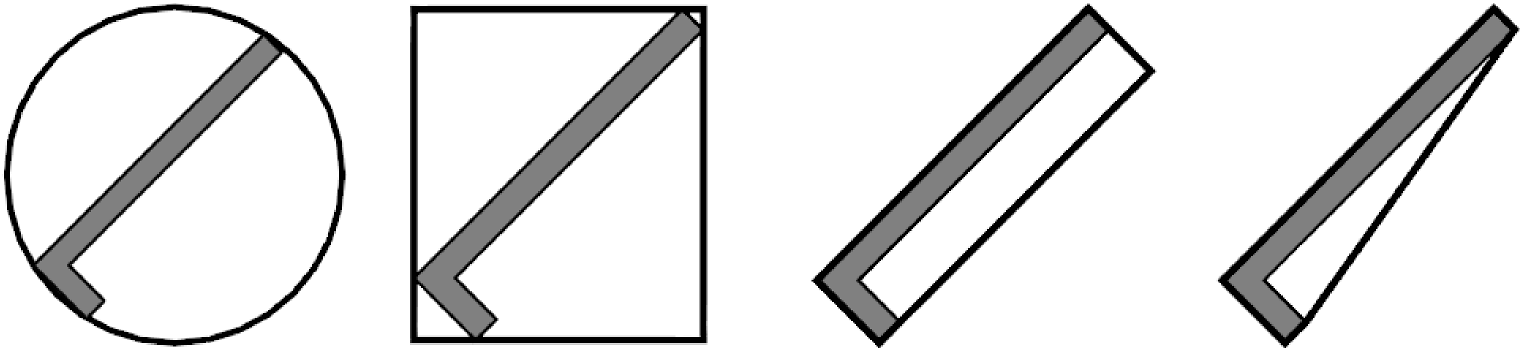


Robot   Obstacle Map   Configuration Space

Collision Checking

Bounding Regions

- Bounding regions can speed up collision checks by reducing the number of edges that have to be compared
  1. Check if the bounding regions collide
  2. If the bounding regions collide, check if the polygon collides
- Bounding regions should allow fast collision checking
- Examples of bounding regions (left to right):
  - Bounding Sphere, Axis-aligned bounding box (AABB), Oriented bounding box (OBB), Convex hull

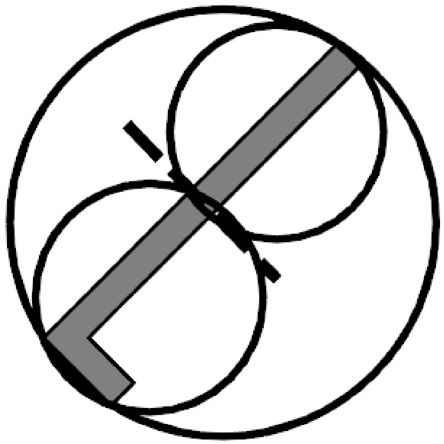


[LaValle: Planning Algorithms]

Collision Checking

Hierarchical methods

- A bounding region can contain several child bounding regions
- Resulting Algorithm:
  1. Check if root bounding regions collide
  2. If yes, replace one bounding region with its child bounding regions and repeat step 1 for each resulting pair until only leaf bounding regions remain
  3. Check the actual polygons for collision



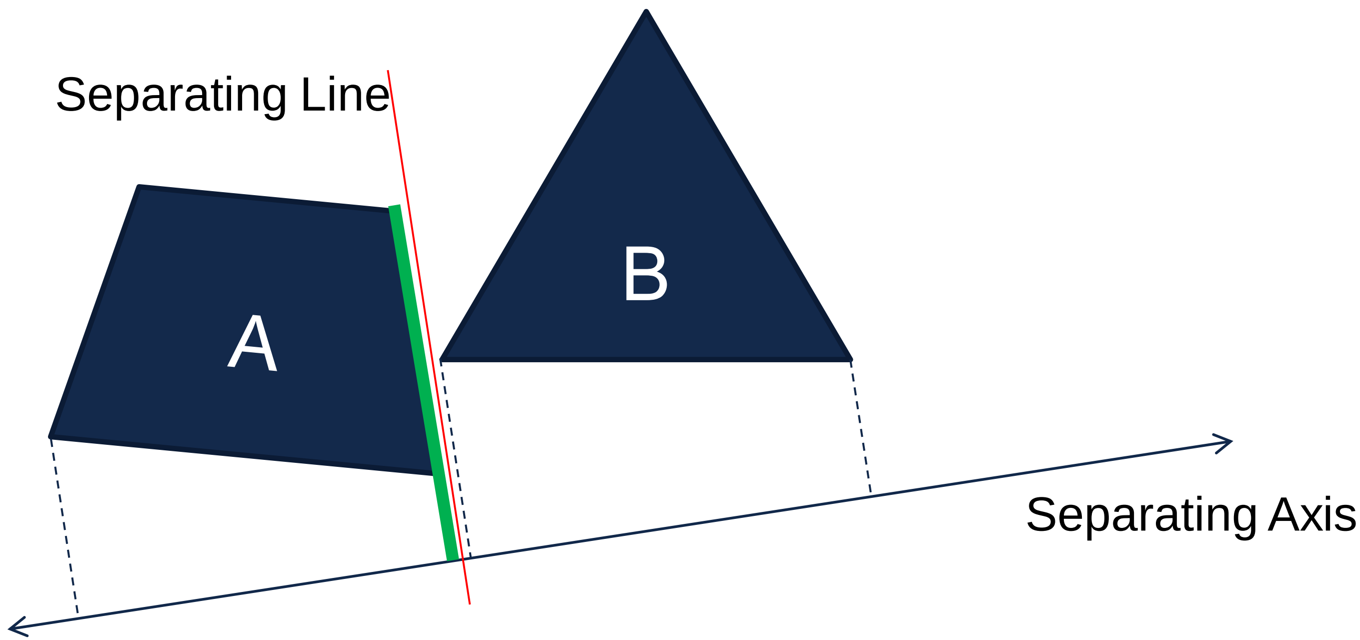
[LaValle: Planning Algorithms]

Collision Checking

Separating Axis Theorem

Method for testing collision check between two convex polygons:

- Two convex polygons collide iff there is no Line separating them
- This line is parallel to one of the polygons’ edges
- The separating axis is orthogonal to this polygon edge => Test all edges
- The projection the polygons to the separating axis does not overlap

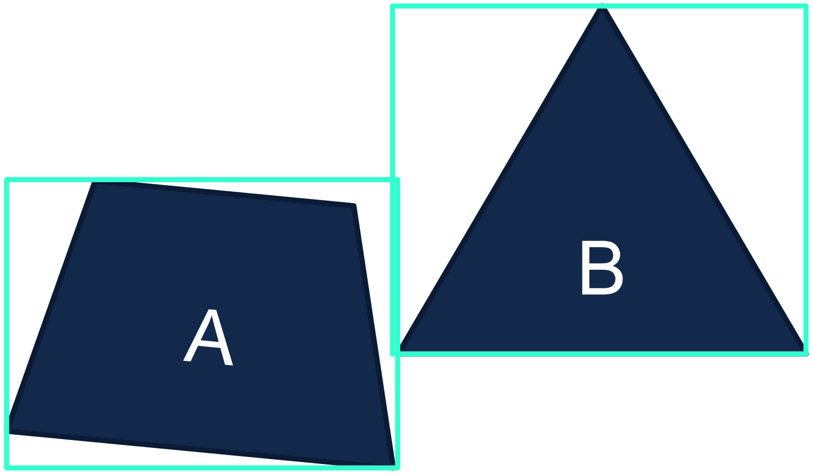


Collision Checking

Separating Axis Theorem: Example

Check Polygons A and B for collision

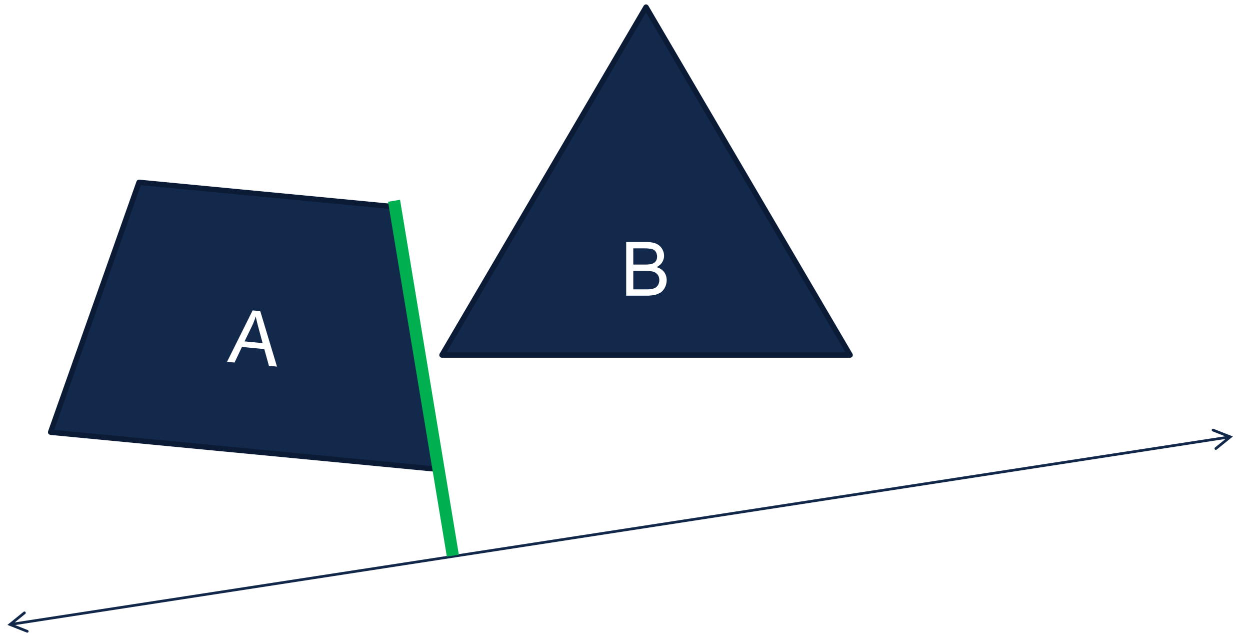
➢ Bounding Boxes do collide



Collision Checking

Separating Axis Theorem

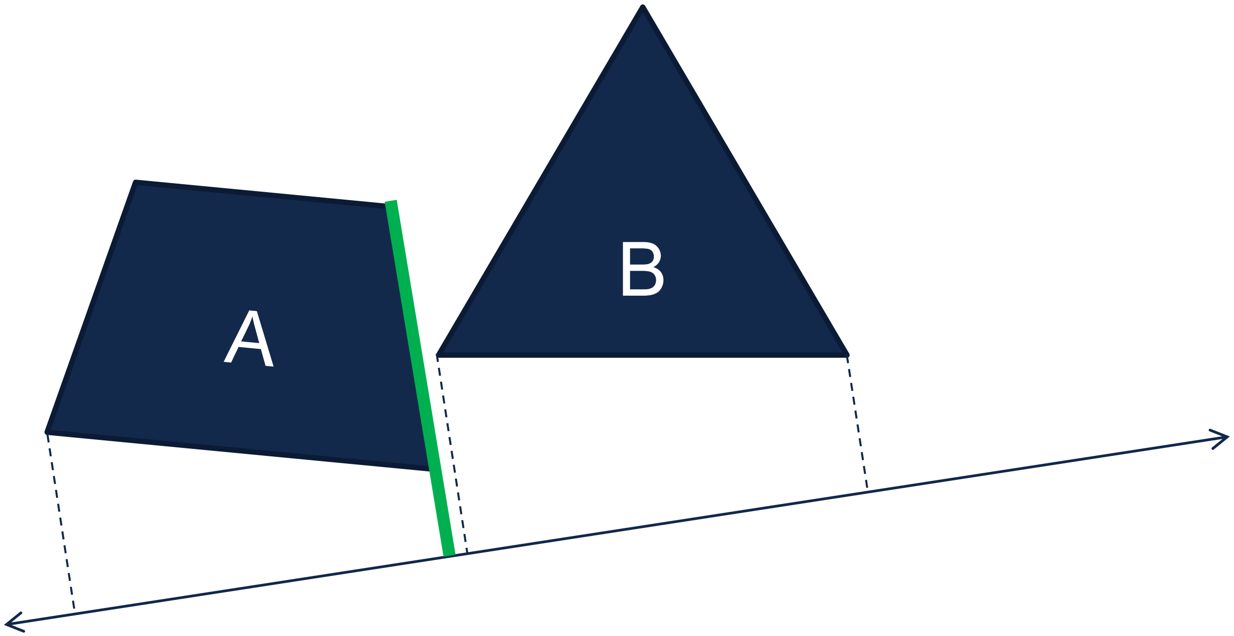
- Iterate over all edges
- For each edge, consider an axis that is orthogonal to the edge



Collision Checking

Separating Axis Theorem

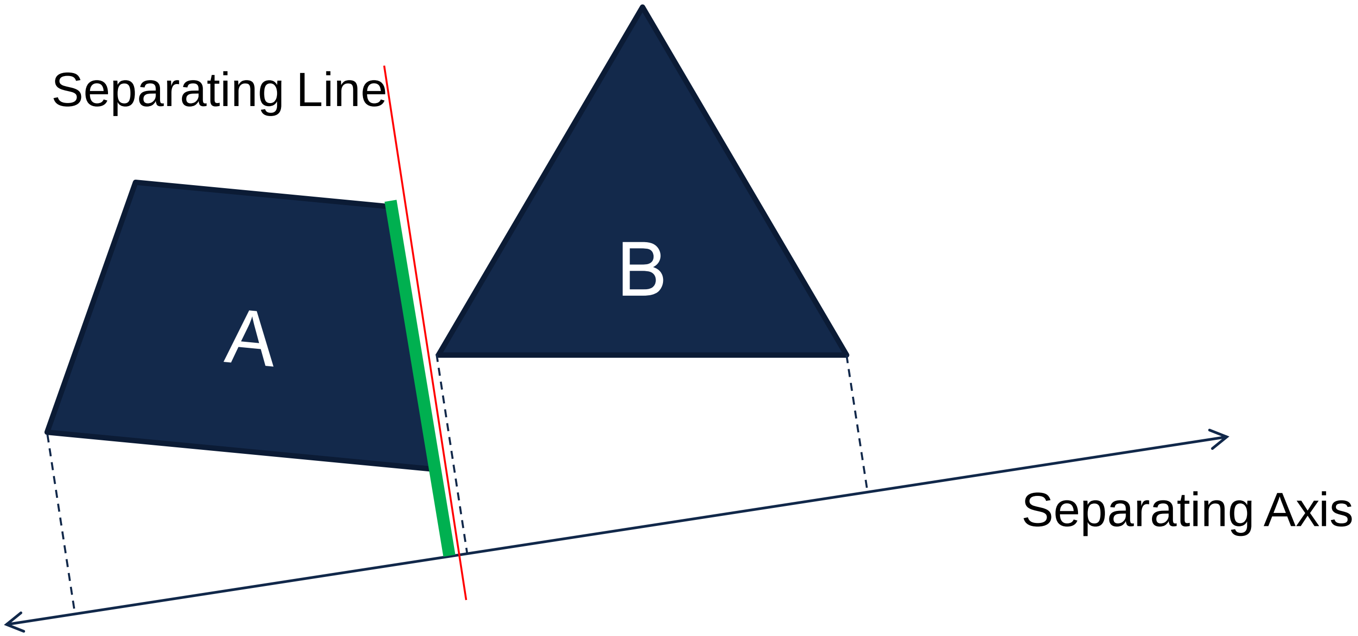
- Iterate over all edges
- For each edge, consider an axis that is orthogonal to the edge
- Project the polygons to this axis and check whether the intervals overlap



Collision Checking

Separating Axis Theorem

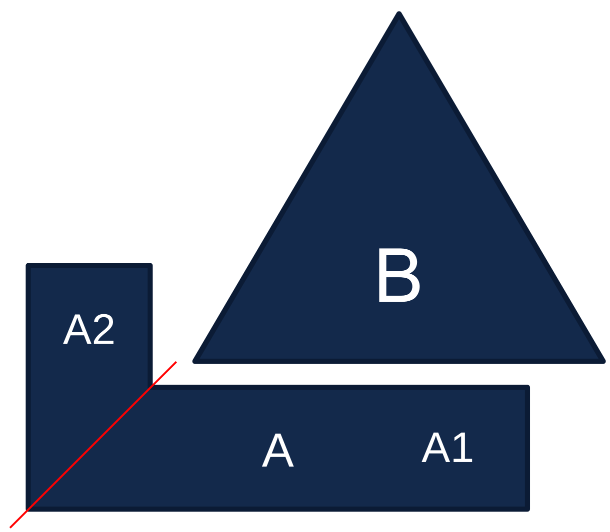
- Iterate over all edges
- For each edge, consider an axis that is orthogonal to the edge
- Project the polygons to this axis and check whether the intervals overlap
- If the intervals do not overlap, the axis is called separating axis and any line between the two polygons is called separating line



Collision Checking

Check collision of concave polygons

- Separating Axis Theorem is not applicable for concave polygons
  - ➢ Test a convex decomposition or check each edge of both polygons for collision
- The red line indicates a possible convex decomposition of polygon A into polygons A1 and A2



Collision Checking

Grid based methods

- Instead of polygons, the robot (left picture) and its environment (middle picture) can be represented as a grid map
- Collision checking can be performed by checking, if the Minkowski Sum (right picture) contains the robot position
- Problem: Different Minkowski sum for each orientation of the robot

[Ziegler, Stiller: Fast collision checking for intelligent vehicle motion planning]

Collision Checking

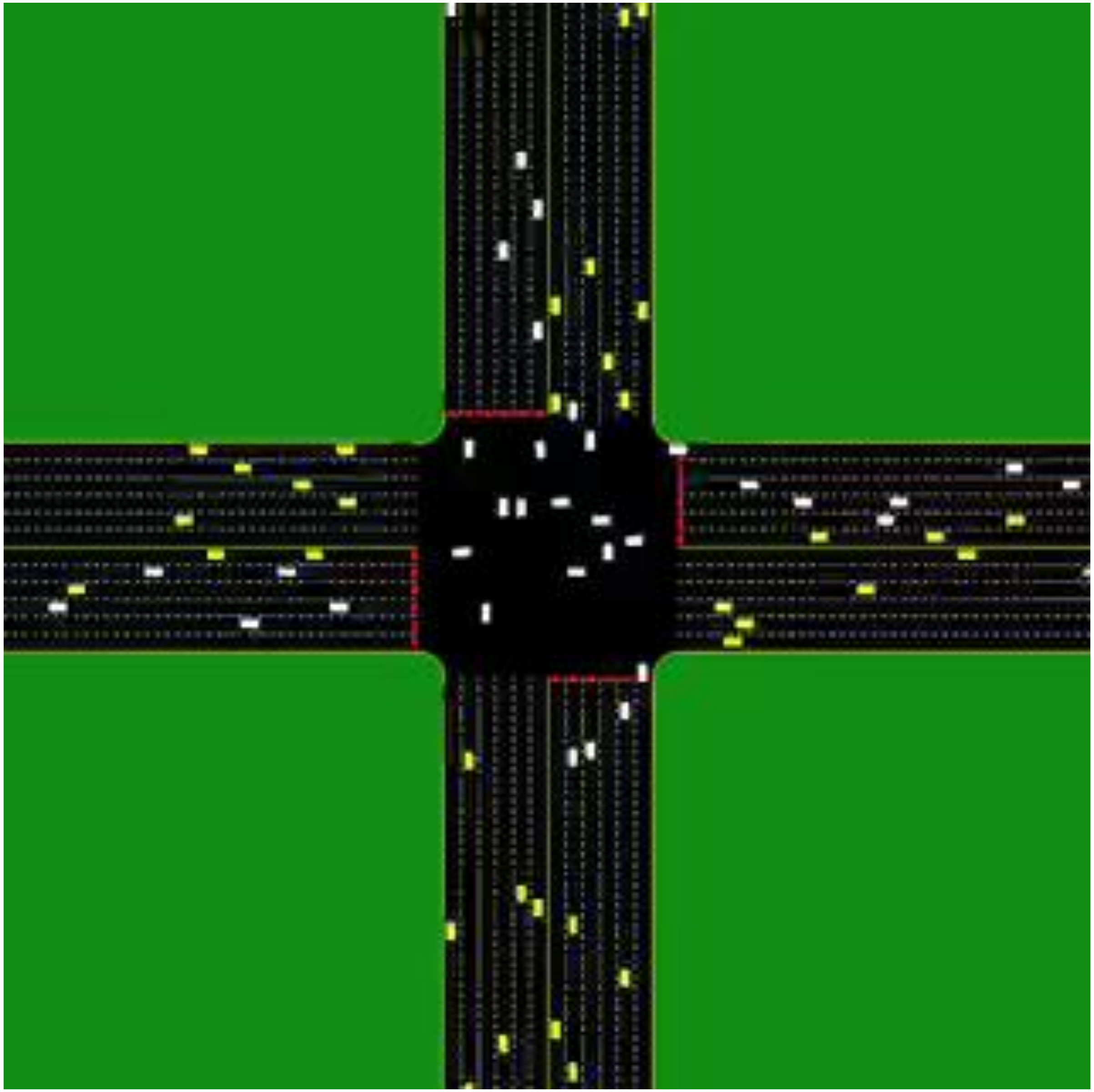
Summary

- There are polygon based methods and grid based methods for collision checking
- For polygon based methods can be accelerated by using bounding regions
- Convex Polygons can be checked for collisions using the Separating Axes Theorem
- For an obstacle grid, the Minkowski sum can be used for efficient collision checking

Conclusions

What have we learned?

- Non-holonomic and holonomic constraints
- Simple Single Track Model
  - The Simple Single Track model is Small Time Locally Controllable
- Dubins and the Reeds-Shepp car
- Affine Transformation matrix and homogenous coordinates
- Collision Detection
  - Bounding regions
  - Separating Axis Theorem



http://spectrum.ieee.org/cars-that-think/transportation/self-driving/the-scary-efficiency-of-autonomous-intersections

http://spectrum.ieee.org/cars-that-think/transportation/self-driving/the-scary-efficiency-of-autonomous-intersections

Gereon Hinz

STTech GmbH

Floriansmühlstraße 8 - 80939 München

Tel: 089/905499430

gereon.hinz@sttech.de

www.sttech.de