Ground Vehicle Motion Control II

LINKÖPING UNIVERSITY

Closed Loop Rapidly-Exploring Random Tree (CL-RRT)

Today I will present the method Closed Loop Rapidly-Exploring Random Tree. The presentation will be based on the following material from previous lectures:

- Rapidly-exploring random tree (lecture 4)
- Pure-Pursuit Control (lecture 7)
- Kinematic Model (lecture 3)
- Dubins Car (lecture 3)

The main reference is the paper:
Real-Time Motion Planning With Applications to Autonomous Urban Driving, Kuwata et.al.,IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, VOL. 17, NO. 5, SEPTEMBER 2009

Material from Previous Lectures

Kinematic Model (Lecture 3)

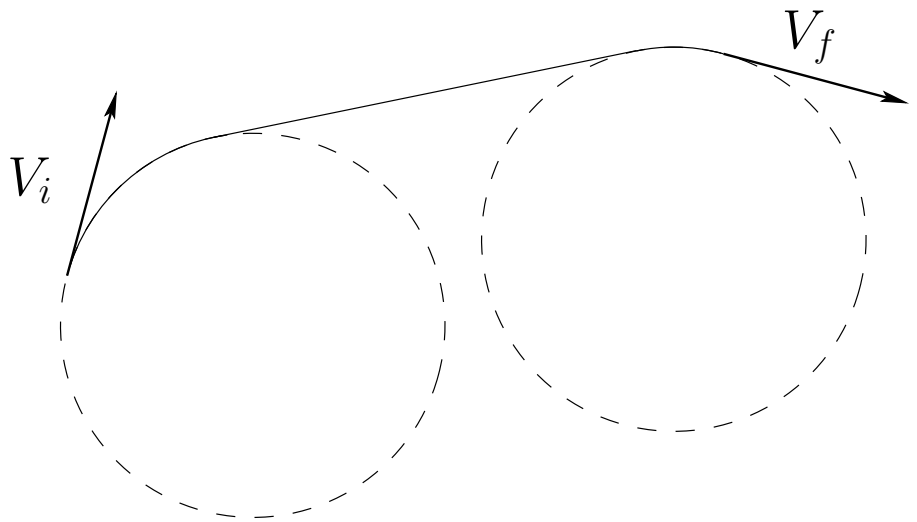
Example of a kinematic model:

$$
\begin{aligned}
\dot{x} &= v\cos\theta \\
\dot{y} &= v\sin\theta \\
\dot{\theta} &= v(\tan\delta)/l \\
\dot{\delta} &= u_1 \\
\dot{v} &= a \\
\dot{a} &= u_2 \\
\delta &\in [-\delta_{max}, \delta_{max}] \\
u_1 &\in [-\dot{\delta}_{max}, \dot{\delta}_{max}]
\end{aligned}
$$

Note that the steering angle and acceleration are states.
This gives a smoother trajectory.

Dubins Car (Lecture 3)

Problem: Find the shortest path between two points with the orientation specified at the initial and final point, and the turning radius limited from below

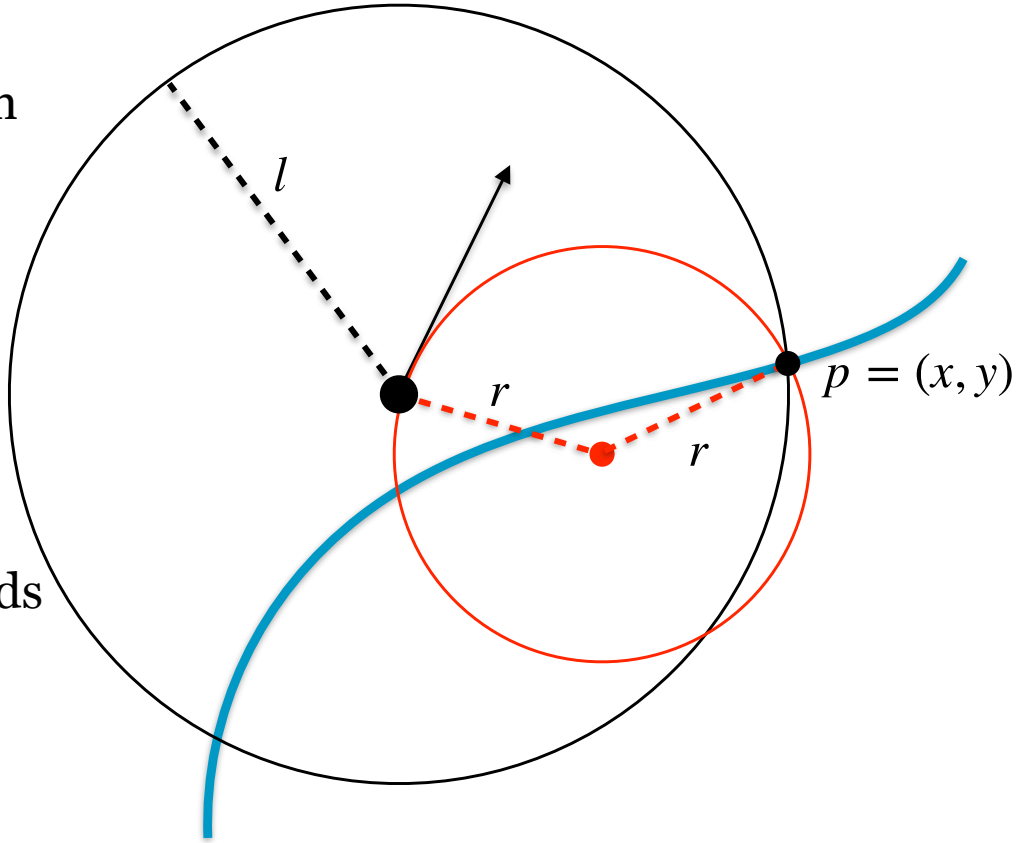


Solution: The optimal solution consists of segments with minimal curvature R_{min} and straight lines.

Pure-pursuit control (Lecture 7)

- A simple control technique to compute the arc needed for a robot to get back on path
- Derived early 80's
- With a look-ahead horizon, l, find point $p = (x, y)$ on the path to aim for
- Compute the turning radius r to get there
- For a single-track robot, this corresponds to a steering angle

$$\frac{1}{L}\tan\delta = \frac{1}{r}$$



Basic Version of RRT (Lecture 4)

Algorithm 1: RRT w/o. Differential Constraints

$\mathcal{V} \leftarrow \{q_I\}$, $\mathcal{E} \leftarrow \emptyset$;
for $i = 1, \ldots, N$ **do**:
    $q_{\text{rand}} \leftarrow \text{Sample}$;
    $q_{\text{nearest}} \leftarrow \text{Nearest}(\mathcal{G} = (\mathcal{V}, \mathcal{E}), q_{\text{rand}})$;
    $q_{\text{new}} \leftarrow \text{Steer}(q_{\text{nearest}}, q_{\text{rand}})$;
    if $\text{ObstacleFree}(q_{\text{nearest}}, q_{\text{new}})$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{q_{\text{new}}\}$;
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(q_{\text{nearest}}, q_{\text{new}})\}$;
return $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;

LaValle, S. M., & J. J. Kuffner Jr.: "Randomized kinodynamic planning". *The International Journal of Robotics Res*

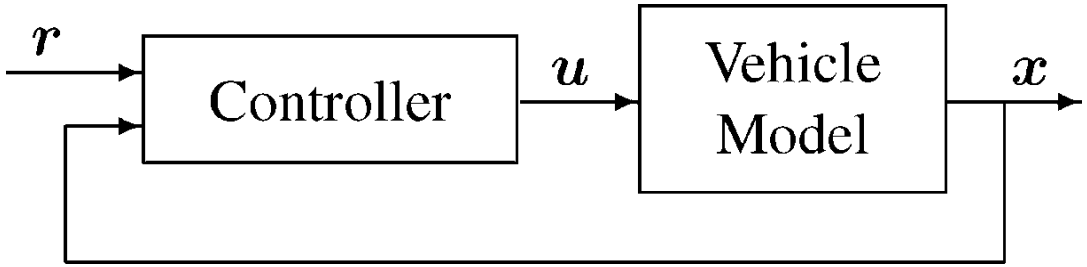
Basic Version of RRT (Lecture 4)

- **Sample**: Gives a sample in the free state space.
- **Nearest**: Provides the vertex in the tree that is closest to the sampled state.
- **Steer**: In general this is a so called two-point boundary value problem (TPBV). Construct a path from the nearest vertex towards the sampled state, often with a maximum path length (alternative strategies exist).
- **ObstacleFree**: Checks whether the path from the closest vertex in the graph to the new state is collision free.

Closed Loop Rapidly-Exploring Random Tree (CL-RRT)

Closed Loop Rapidly-Exploring Random Tree (CL-RRT)

In contrast to the previous work, CL-RRT samples an input to the stable closed-loop system consisting of the vehicle and the controller.



- CL-RRT works for vehicles with unstable dynamics, such as cars and helicopters, by using a stabilizing controller.
- A single input to the closed-loop system can create a long trajectory (on the order of several seconds) while the controller provides a high-rate stabilizing feedback

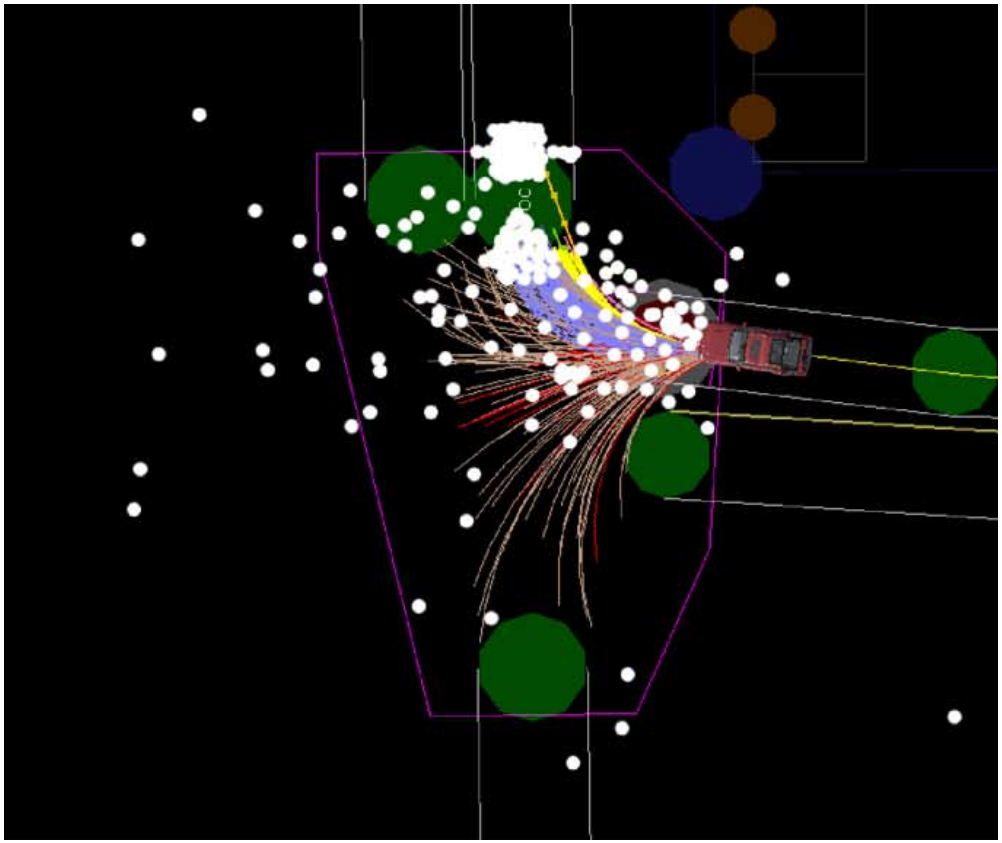
CL-RRT: Sampling strategies

Given a reference position and heading (x_0, y_0, θ_0) a sample point (s_x, s_y) is generated by:

$$\begin{bmatrix} s_x \\ s_y \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + r \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \text{ with } \begin{cases} r = \sigma_r |n_r| + r_0 \\ \theta = \sigma_\theta n_\theta + \theta_0 \end{cases}$$

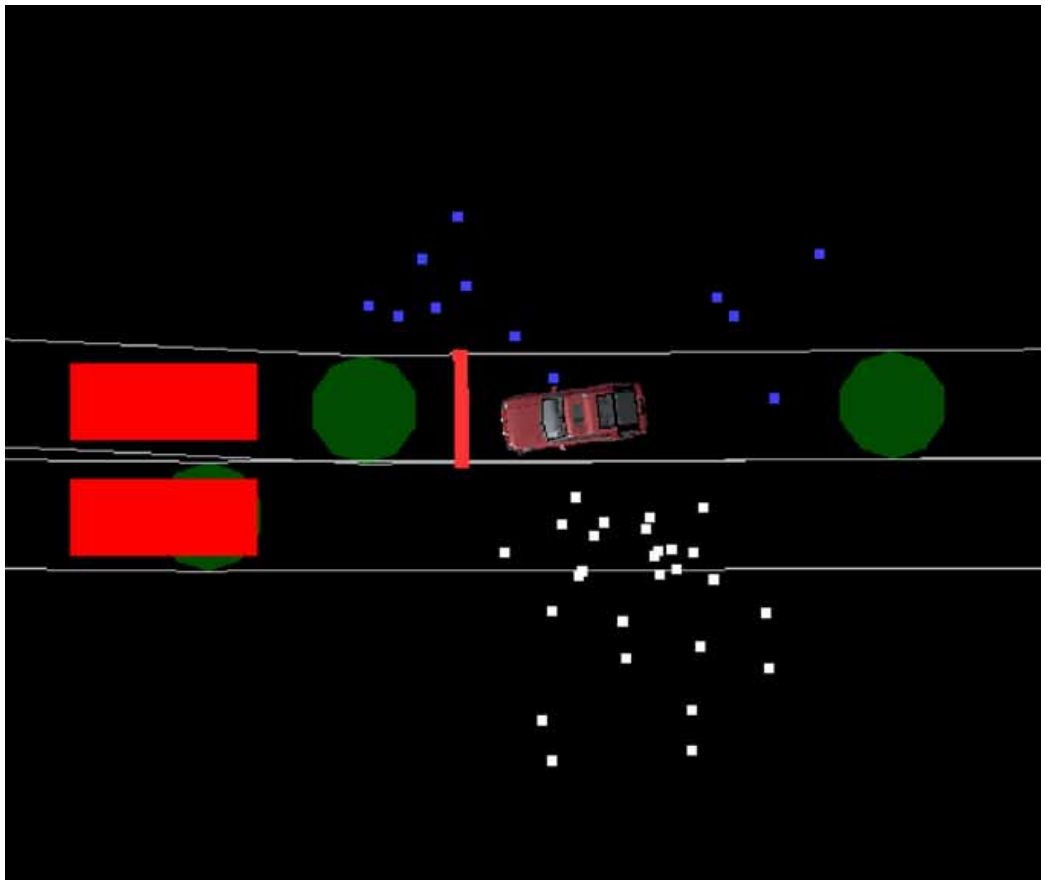
where n_r and n_θ are random variables with standard Gaussian distribution, σ_r is the standard deviation in the radial direction σ_θ is the standard deviation in the circumferential directions, and r_0 is an offset with respect to (x_0, y_0).

Example: Sampling at a right hand turn at an intersection



A wide and relatively short Gaussian distribution is used that covers the open space inside the intersection boundary. The value of σ_r is set to the distance to the goal and σ_θ is set at 0.4π.

Example: Sampling strategy for an U-turn



The vehicle is facing the road blockage (red).

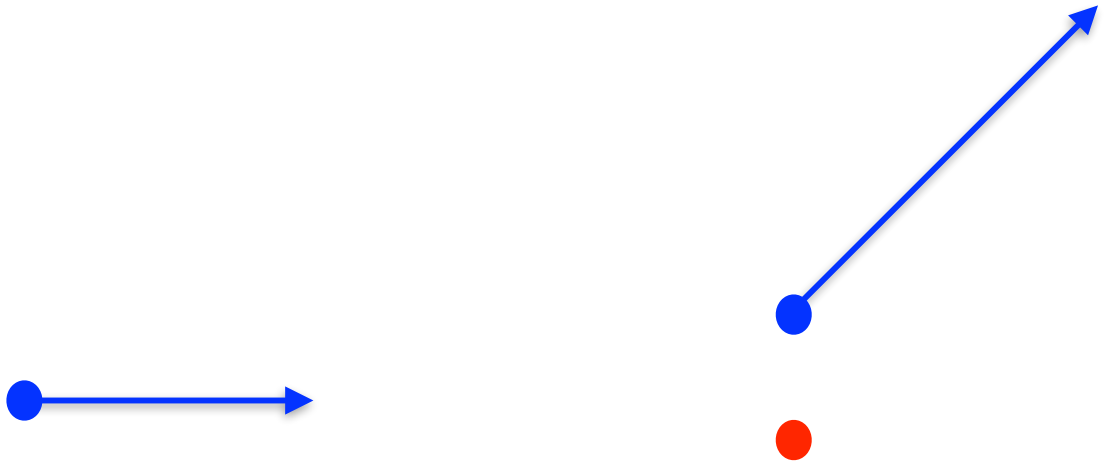
Blue and white dots are reversed forward manoeuvres, respectively.

Parameters used:

$$
\begin{aligned}
1^{\text{st}}: &\quad \sigma_r = 5, \quad \sigma_\theta = 0.1\pi, \quad r_0 = 3, \quad \theta_0 = 0.44\pi \\
2^{\text{nd}}: &\quad \sigma_r = 5, \quad \sigma_\theta = 0.2\pi, \quad r_0 = 3, \quad \theta_0 = -0.17\pi \\
3^{\text{rd}}: &\quad \sigma_r = 10, \quad \sigma_\theta = 0.25\pi, \quad r_0 = 3, \quad \theta_0 = 0.83\pi.
\end{aligned}
$$

CL-RRT: Heuristic

Which node (blue) is “closest” to the sample state (red)?

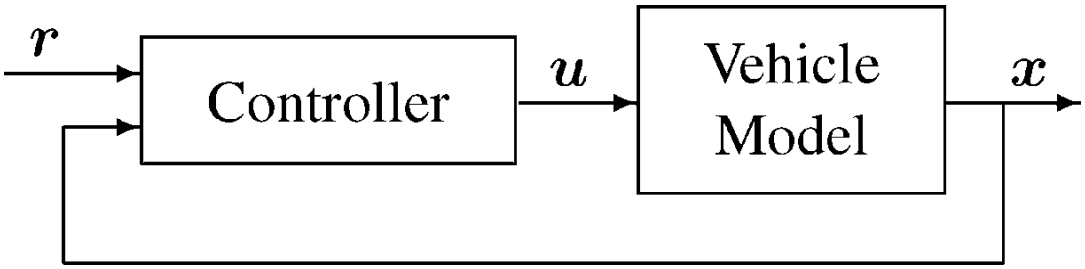


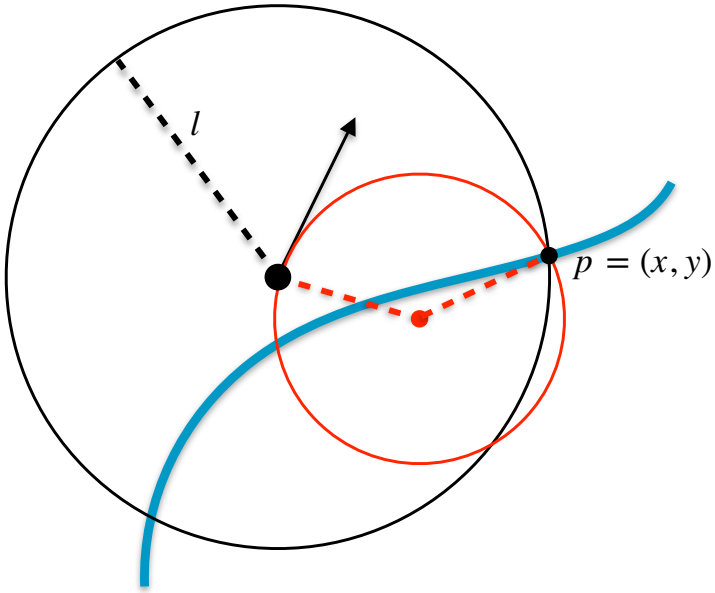
In RRT the distance was used, and the answer in this case the nearest node would be the one to the right.

In CL-RRT the Dubins distance is used, defined as the shortest path a with a minimal turning radius ρ. The nearest node would be the one to the left if ρ is sufficiently large. This is called the exploration heuristic.

CL-RRT: The controller

A pure-pursuit controller is used in the CL-RRT



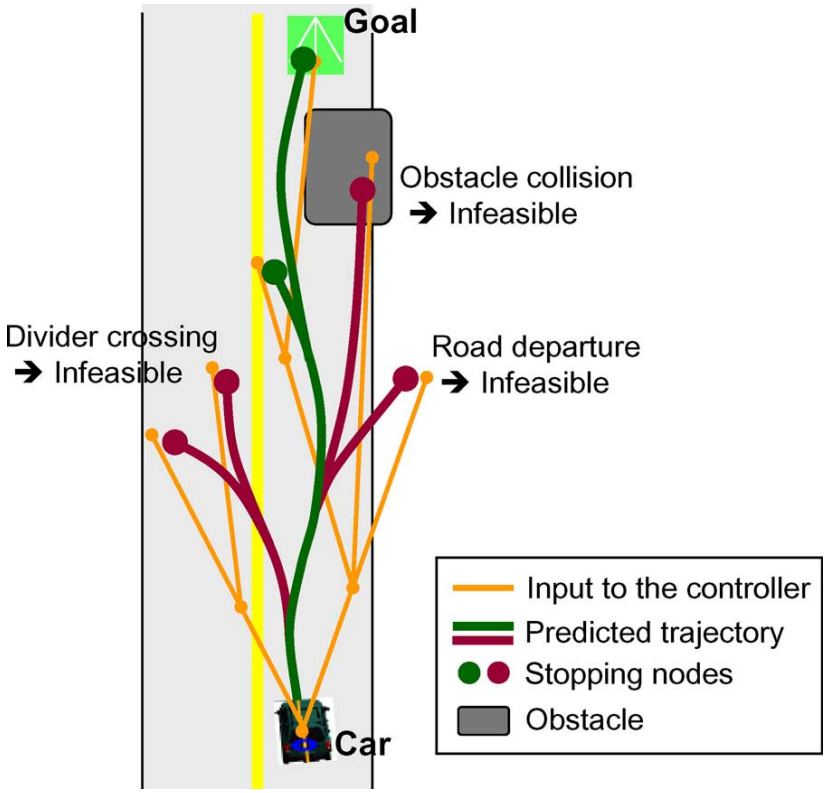


The next step is to describe how to construct reference paths for the controller.

CL-RRT: Expand the tree

Procedure to expand the (orange) tree

- Generate a sample position s.
- For each node q in the tree, in the order sorted by the heuristic, use the line segment between node q and sample s to extend the tree.
- Use the new reference path in the orange tree as input to the controller to generate a trajectory $\mathbf{x}(t)$, $t \in [t_1, t_2]$ until it stops (red and green curves).

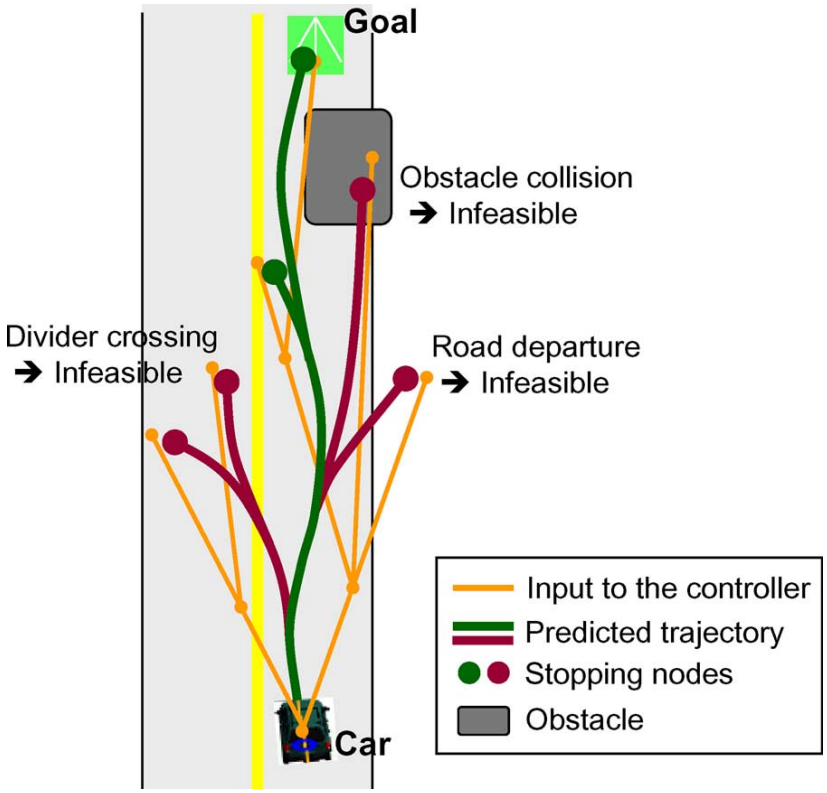


CL-RRT: Expand the tree

- If $\mathbf{x}(t) \in \mathcal{X}_{free}(t)$ for all $t \in [t_1, t_2]$, then add the sample s to the tree and also some intermediate nodes.
- Else if the intermediate nodes are all feasible, add them to the tree.

If no nodes were added above, then repeat the process with a new node q.

The nodes given by sample s will not be considered as potential connection points for new samples, unless it is the only node of the tree or the driving direction changes.



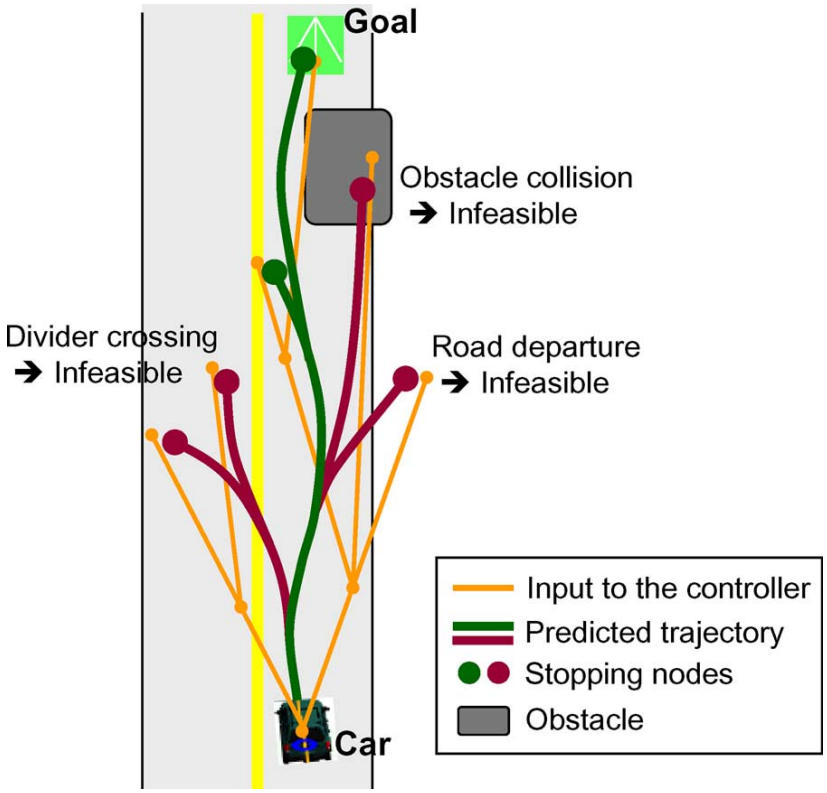
CL-RRT

The tree is expanded until the goal has been reached. After that the nodes are mainly sorted by ascending order of total cost to reach the sample:

$$C_{total} = C_{cum}(q) + L_{\rho}(s)/v$$

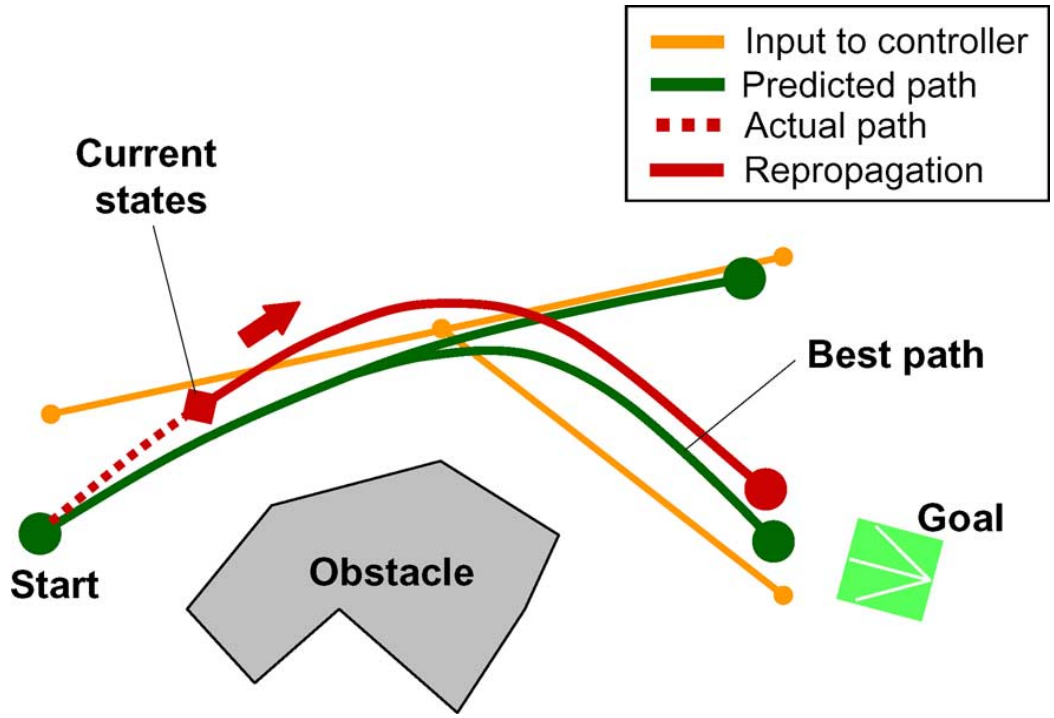
where $C_{cum}(q)$ is the cumulative cost from the root of the tree to a node q, $L_{\rho}(s)$ is the Dubins distance, and v is the sampled speed. The objective is to make the new trajectories approach the shortest path. This is called the optimisation heuristic.

Exploration heuristic and optimisation heuristic



Replanning

When the vehicle has moved forward a step, the expansion of the tree continues and obsolete parts are removed.

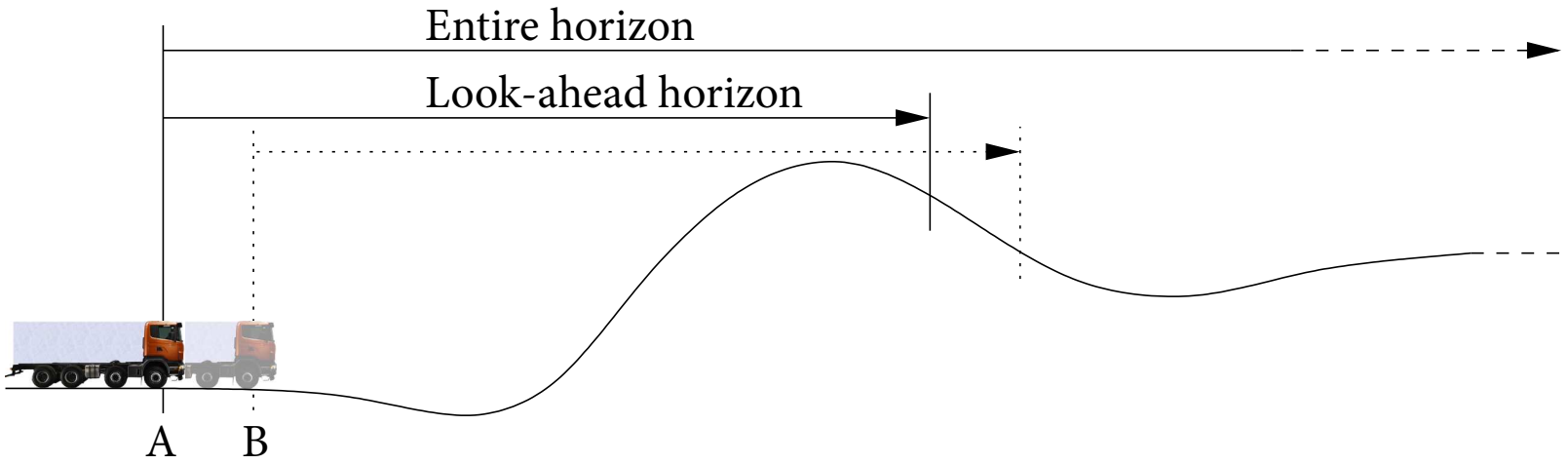


See the reference on the first slide for further details

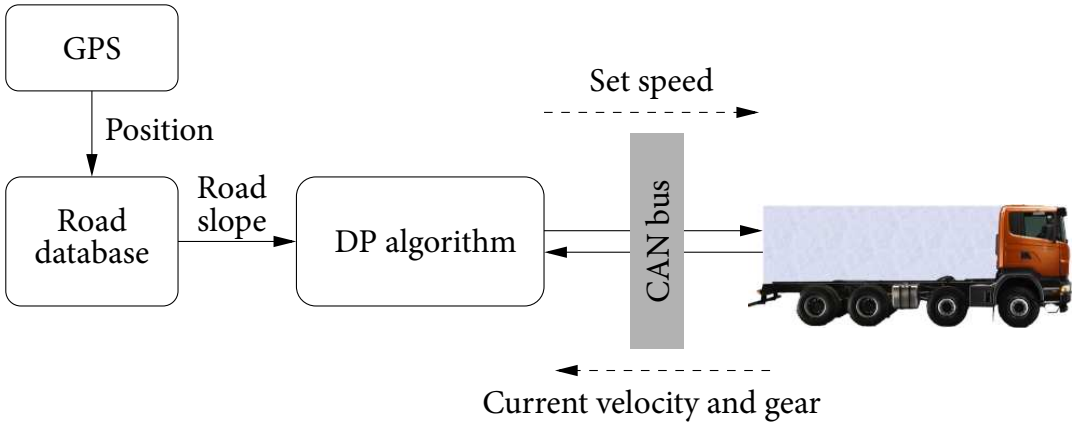
Model Predictive Control (MPC)

Example of a Model Predictive Control (MPC) application

Optimal cruise control with information about the road slope ahead of the vehicle.



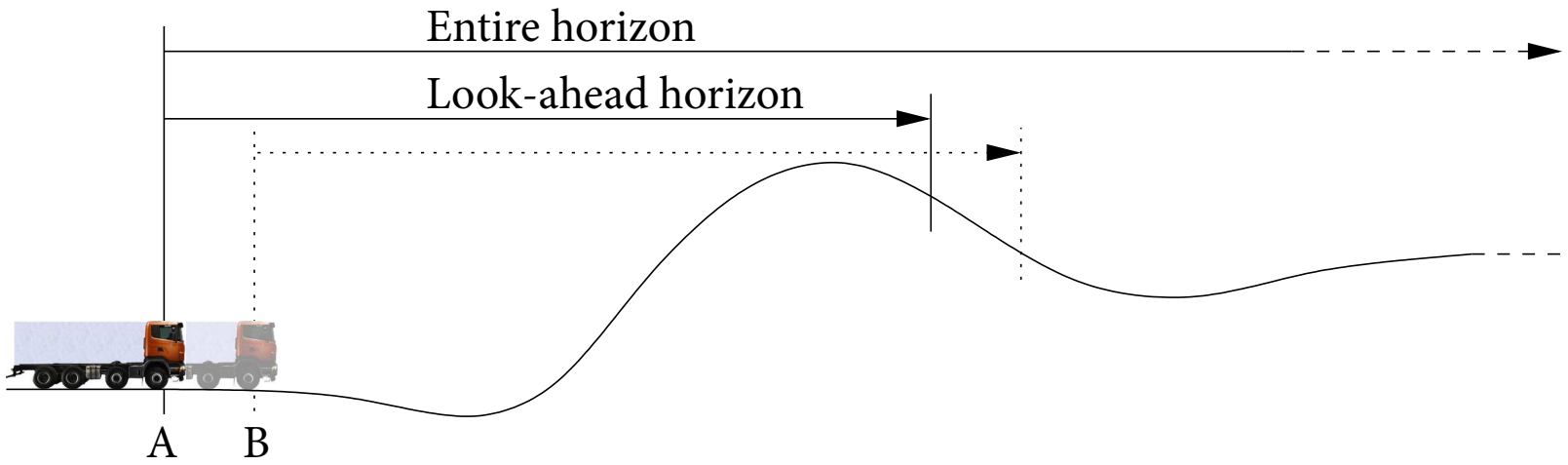
Information flow



- The GPS gives the position of the truck.
- A road data base gives the road slope ahead of the vehicle from the current position to limited distance ahead of the vehicle (typically ~2 km).
- A dynamic programming algorithm, together with a model of the vehicle, is used to determine the control signal that minimises the fuel consumption on this intervall.

Model Predictive Control (MPC)

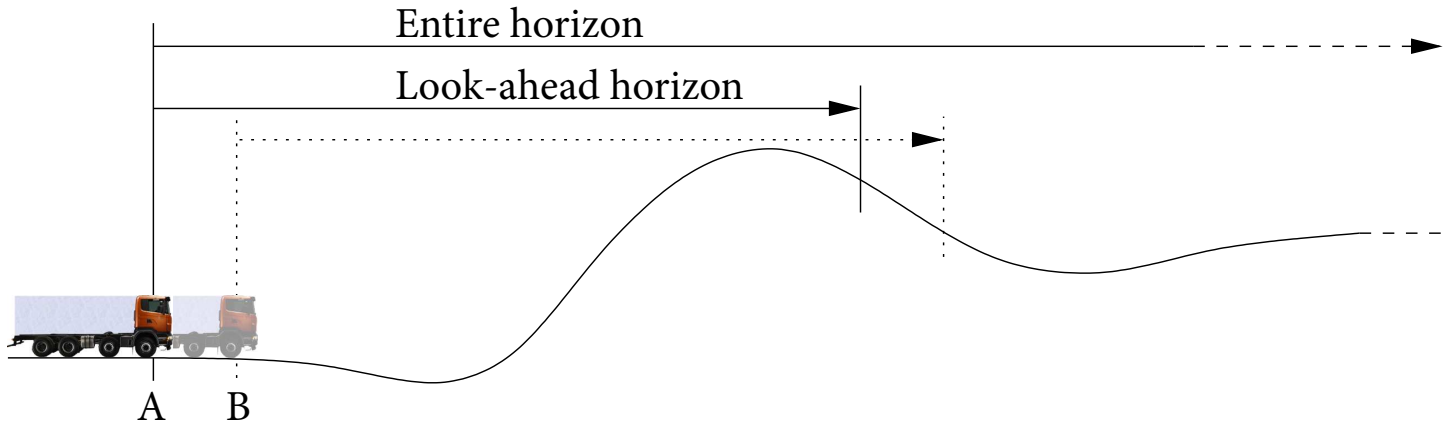
The basic idea in MPC is to use the first control signal at point A and use that until the vehicle reaches point B.



At point B the procedure on the previous slide is repeated and a new control signal is obtained that is applied at point B

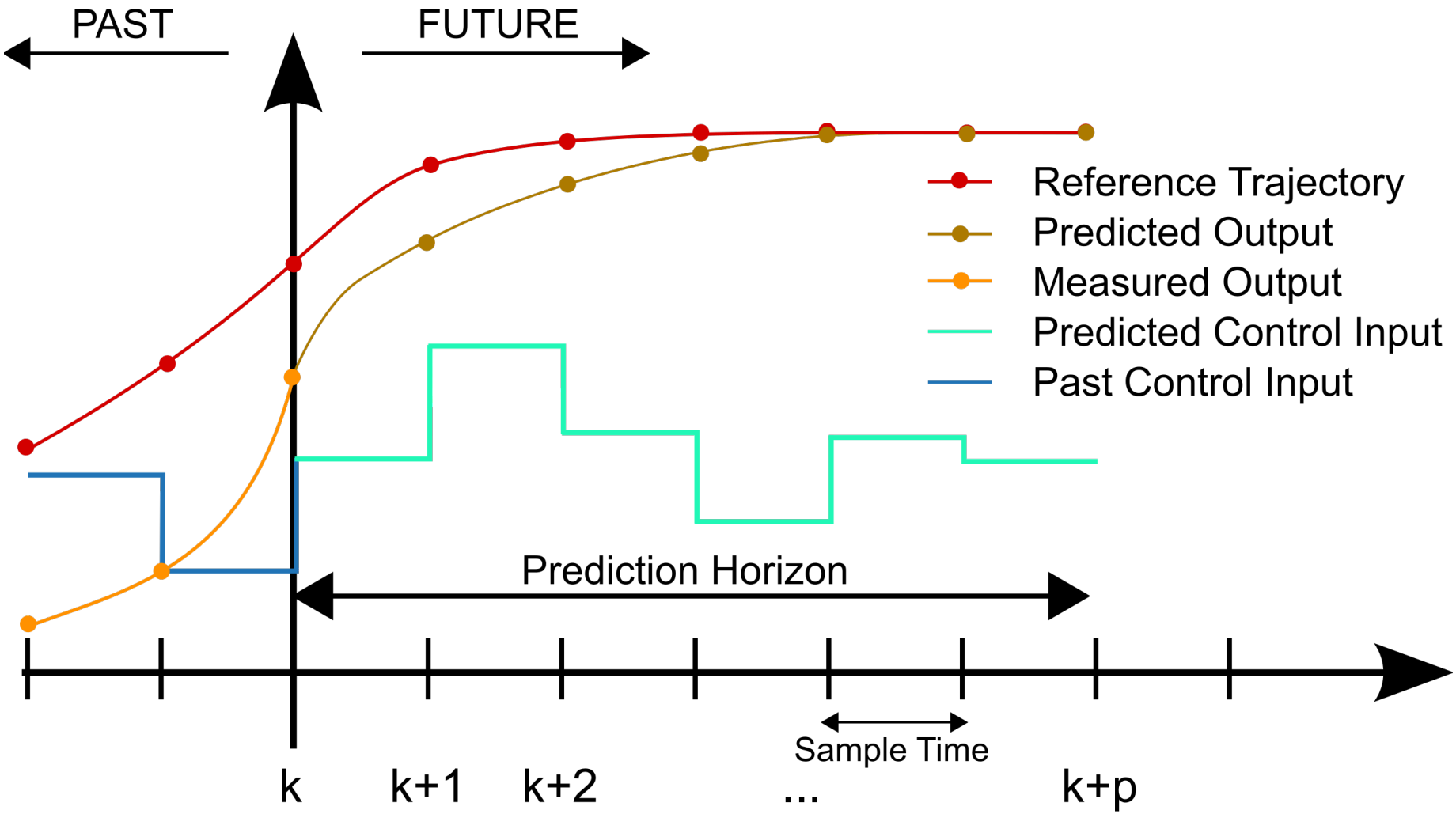
Model Predictive Control (MPC)

An advantage with the MPC approach is that it can into account limitations on control signals and states.



In this application the truck will accelerate before an uphill if it is necessary to avoid that the speed drops below a lower limit on speed due to limits on the propelling force.

Another advantage is that can handle changes driving conditions, e.g, new vehicles


PAST
FUTURE
Reference Trajectory
Predicted Output
Measured Output
Predicted Control Input
Past Control Input
Prediction Horizon
Sample Time
k
k+1
k+2
...
k+p

MPC for a discrete linear model

The system is described by the discrete linear model

$$\begin{aligned} x(k+1) &= Fx(k) + Gu(k) \\ y(k) &= Cx(k) \\ z(k) &= Mx(k) \end{aligned}$$

Assume that the current state is $x(k)$ is known (measured or estimated by an observer). Then a prediction of the next state is

$$x(k+1) = Fx(k) + Gu(k)$$

Using the model once again a two-step prediction is obtained

$$\begin{aligned} x(k+2) &= Fx(k+1) + Gu(k+1) \\ &= F^2x(k) + FGu(k) + Gu(k+1) \end{aligned}$$

Note that it is a linear function of $x(k), u(k), u(k+1)$. By continuing the procedure a state $x(k+n)$ can be written as a linear function of $x(k), u(k), u(k+1), \ldots, u(k+n-1)$

Predictions

Consider a horizon of length N. The predictions in this intervall are

$$\begin{aligned} x(k+1) &= Fx(k) + Gu(k) \\ x(k+2) &= Fx(k+1) + Gu(k+1) \\ &= F^2x(k) + FGu(k) + Gu(k+1) \\ &\vdots \\ x(k+N-1) &= \text{ a linear function of } x(k), u(k), \ldots, u(k+N-1) \end{aligned}$$

Let

$$U = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \end{bmatrix}$$

Conclusion: $x(k+n)$ and $z(k+n) = Mx(k+n)$, $n = 0,1,\ldots,N-1$ are linear functions of $x(k)$, U.

The objective function

The following objective function will be considered

$$J(x(k), U) = \sum_{j=0}^{N-1} \|z(k+j)\|_{Q_1}^2 + \|u(k+j)\|_{Q_2}^2$$

where Q_1 and Q_2 are positive definite matrices and the notation $\|x\|_Q^2 = x^T Q x$ is used. are linear functions of $x(k)$ and U. We used that $z(k+j)$ is linear function of $x(k)$ and U.

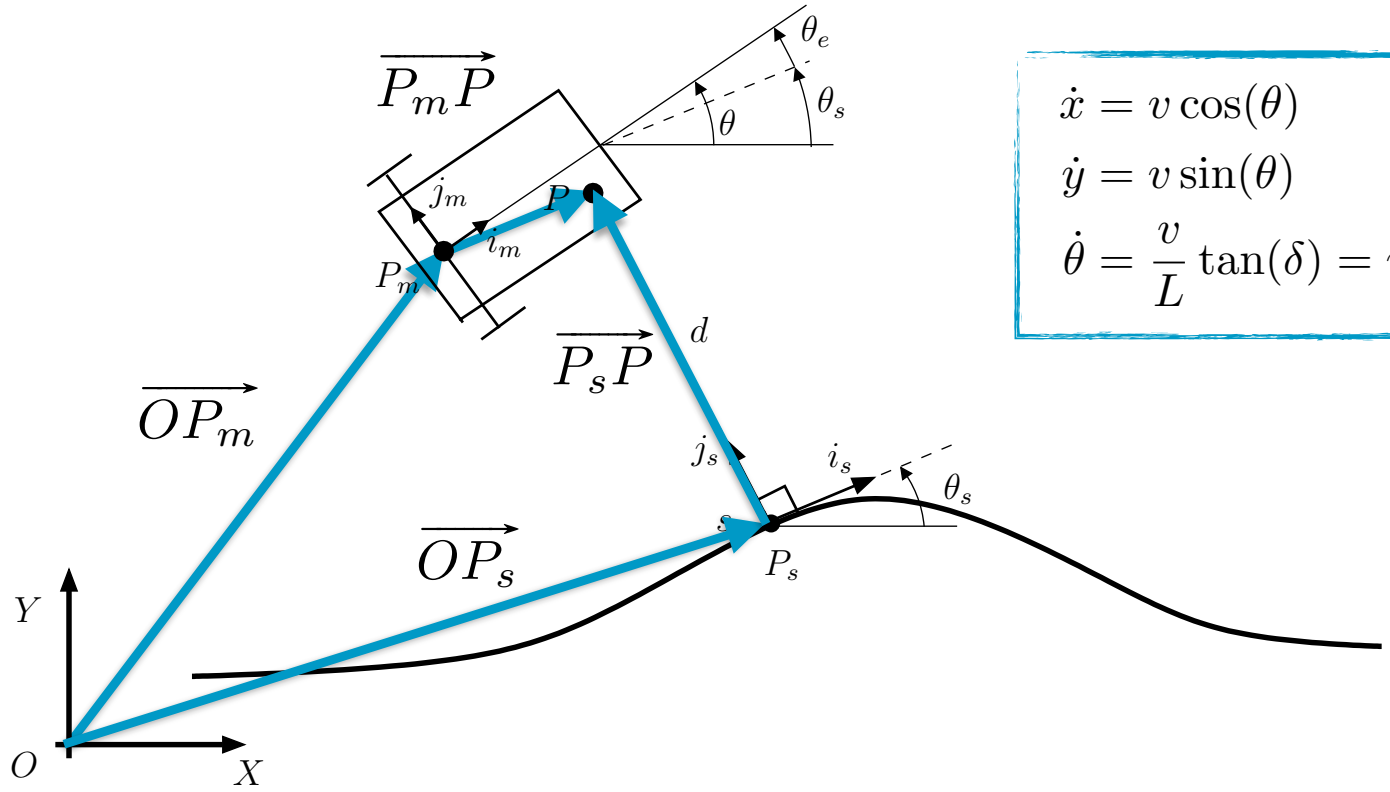
We want to minimise $J(x(k), U)$ for a given $x(k)$. It can be shown that the objective function can be written as

$$J(x(k), U) = \frac{1}{2} U^T H U + f^T U (+C)$$

where H is positive definite matrix, f is vector, and C is constant that can be omitted when want do determine U.

Application: Path tracking

Consider the path following problem from Lecture 7



$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = \frac{v}{L}\tan(\delta) = v\,u, \quad u \sim \text{curvature}$$

Basic idea:Linearise the model and choose the errors d and θ_e as your vector z.

Application: Trajectory tracking

The MPC formulation can easily be extend to include a reference signal $r(k)$

$$\sum_{j=0}^{N-1} \|z(k+j) - r(k+j)\|_{Q_1}^2 + \|u(k+j)\|_{Q_2}$$

This can be used for trajectory tracking if we choose $z(k)$ as position.

Limitations on the control signal

As an example, assume that there is one control signal and it is limited by the condition $|u(k+n)| \leq 1, n = 0,\ldots,N+1$, or equivalently by the two conditions $u(k+n) \leq 1$ and $u(k+n) \geq -1, n = 0,\ldots,N+1$

This can also be written in matrix form

$$\begin{bmatrix} I_N \\ -I_N \end{bmatrix} U \leq \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

or more generally constraint in the form

$$A_u U \leq b_u$$

can be considered.

Quadratic Programming QP

The optimisation problem, with conditions on the control signals, can be formulated as QP

$$\min_U \frac{1}{2} U^T H U + f^T U$$
$$s.t. A_U U \leq b_u$$

To summarise the discussion, the problem to minimise the function

$$J(x(k), U) = \sum_{j=0}^{N-1} \|z(k+j)\|^2_{Q_1} + \|u(k+j)\|^2_{Q_2}$$

with limitations on the control signal can be formulated as a QP.