Control of Autonomous Vehicles II

TSFS12: Autonomous Vehicles –planning, control, and learning systems

Lecture 9: Jan Åslund <jan.aslund@liu.se>

LINKÖPINGS UNIVERSITET

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:

- Kinematic Model (lecture 3)
- Dubins Car (lecture 3)
- Rapidly-exploring random tree (lecture 4)
- Pure-Pursuit Control (lecture 6)

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)

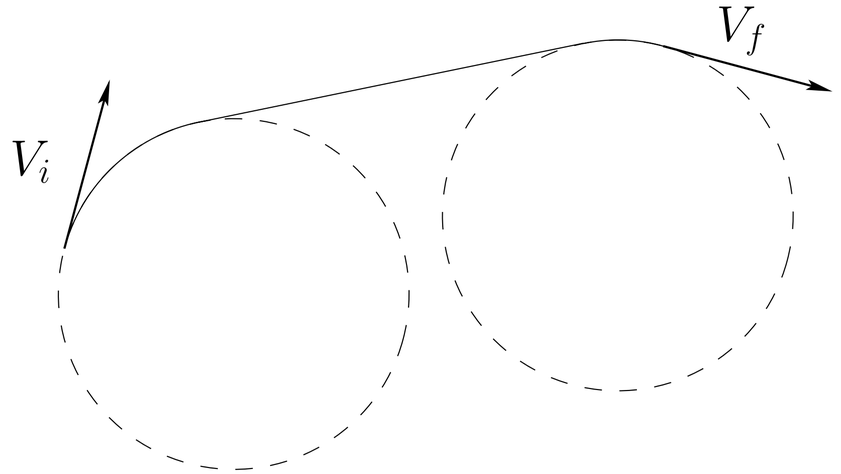
Example of a kinematic model:

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

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

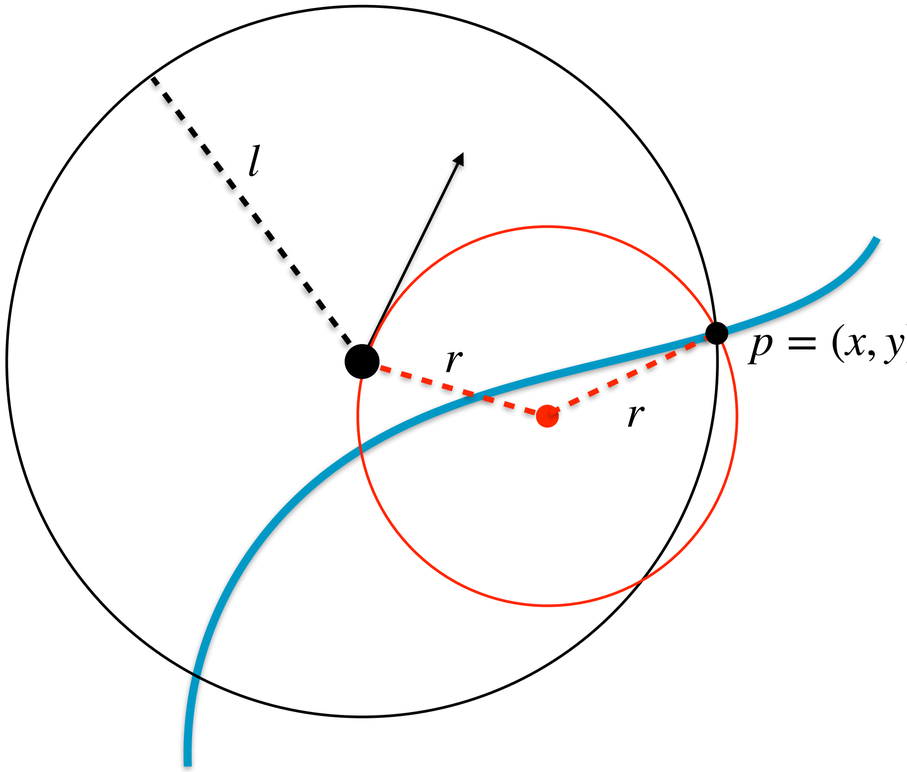


Solution: The optimal solution consists of segments with minimal radius R_{min} and straight lines.

Pure-pursuit control (Lecture 6)

- A simple control technique to compute the arc needed for a robot to get back on path
- 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

$\mathcal{V} \leftarrow \{q_I\}, \quad \mathcal{E} \leftarrow \emptyset;$
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 Research*, 20(5), 378-400, 2001.

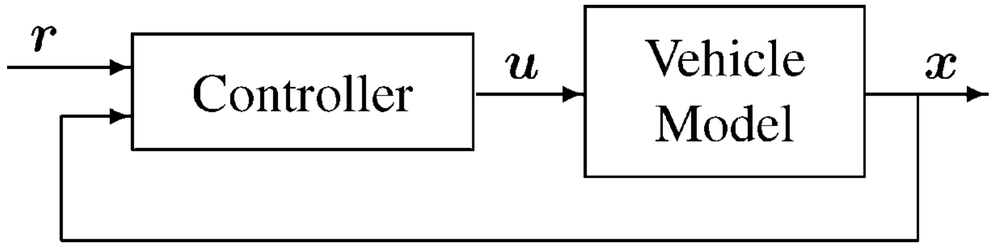
Basic Version of RRT (Lecture 4)

- **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)

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

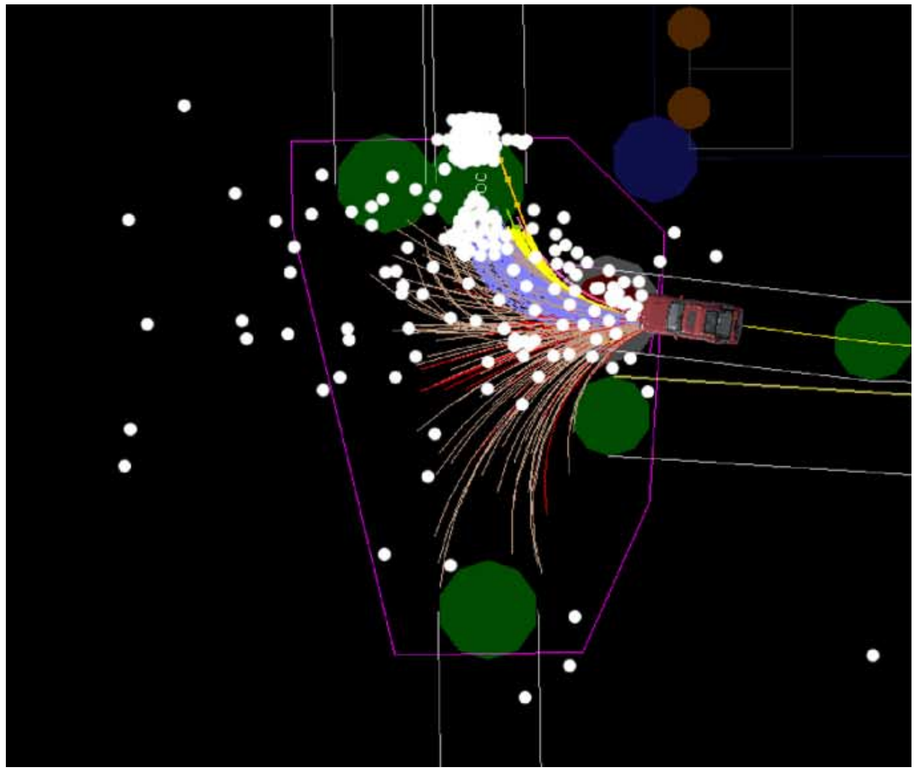
CL-RRT: Sampling strategies

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

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

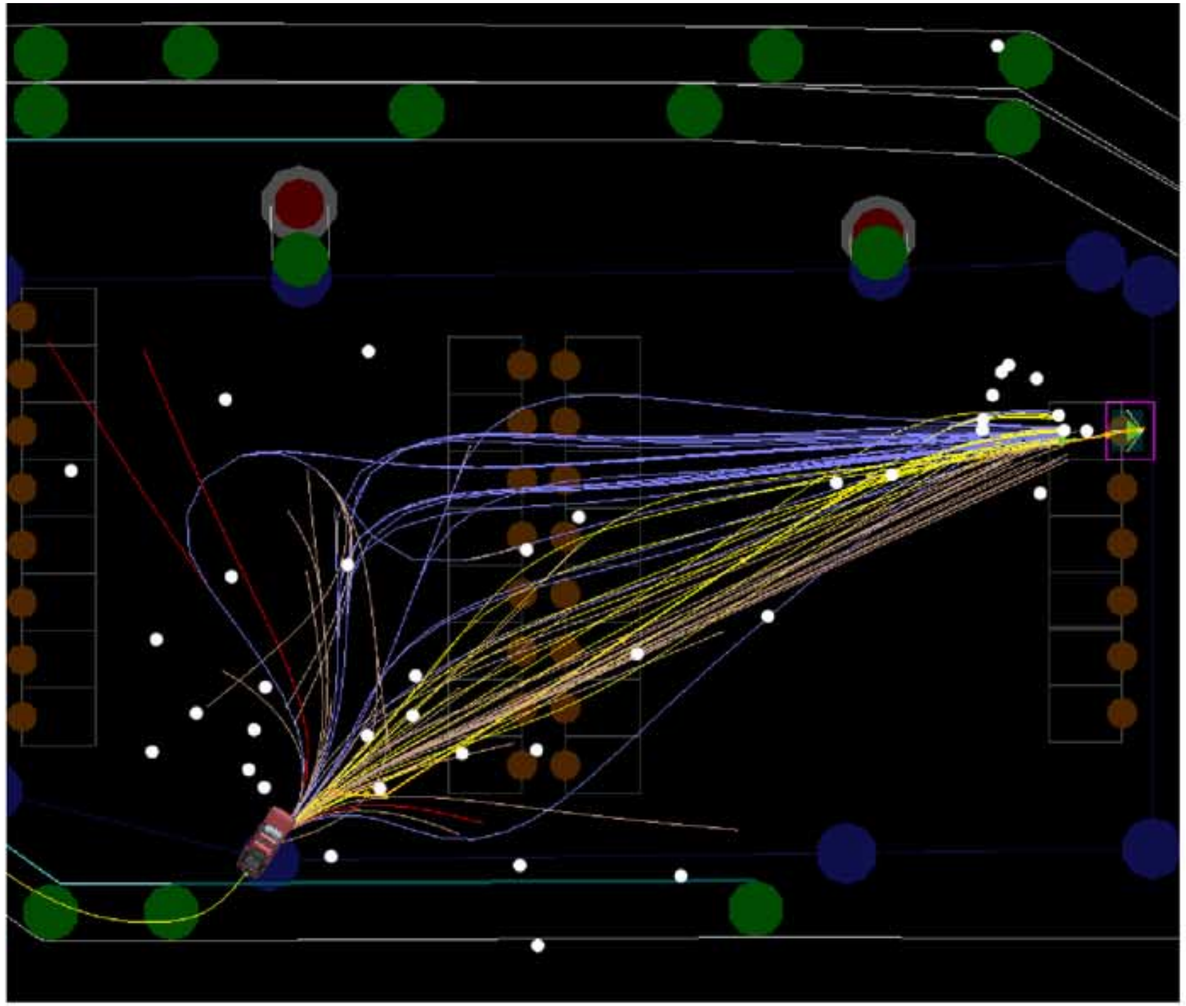
where n_r and n_θ are random variables with standard Gaussian distribution, σ_r is the standard deviation in the radial direction σ_θ is the standard deviation in the circumferential directions, and r_0 is an offset with respect to (x_0, y_0).

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.

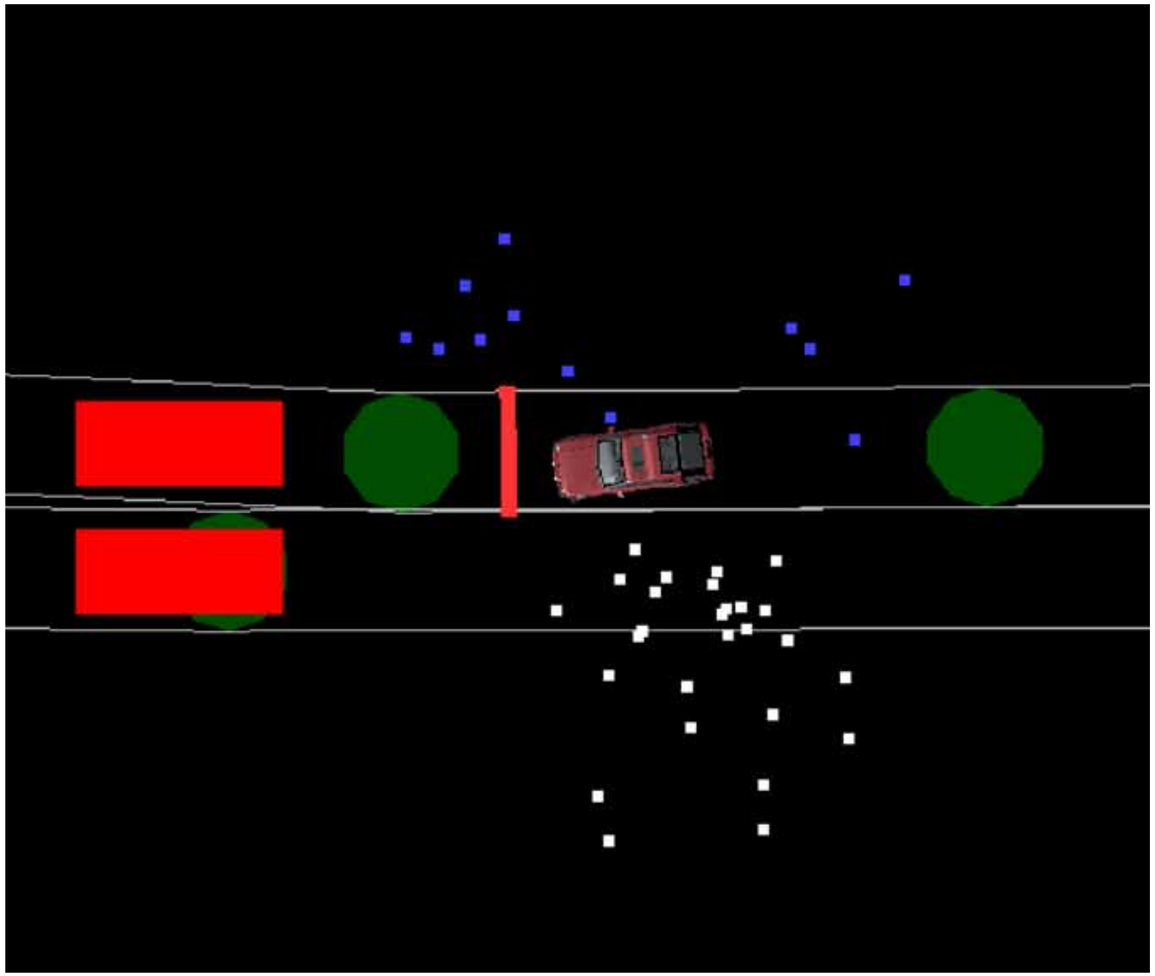
Example: Sampling at a parking lot



Sampling is taken both around the vehicle and around the parking spot.

Around the vehicle a wide and long distribution is used

Example: Sampling strategy for an U-turn

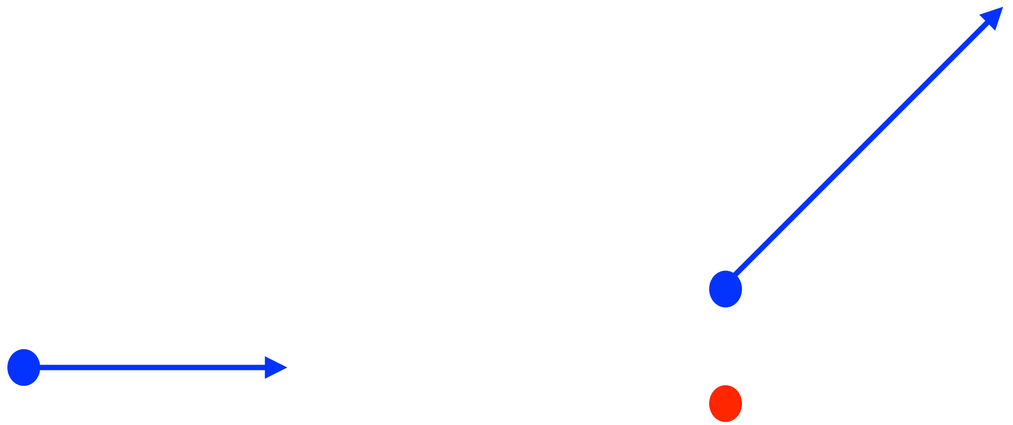


The vehicle is facing the road blockage (red).

Blue and white dots are reverse and forward manoeuvres, respectively.

CL-RRT: Heuristic

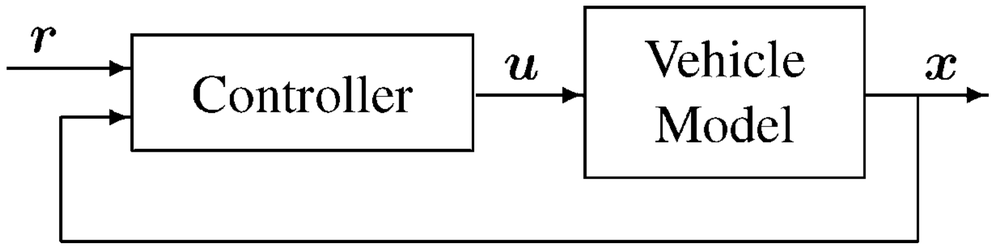
Which node (blue) is “closest” to the sample state (red)?

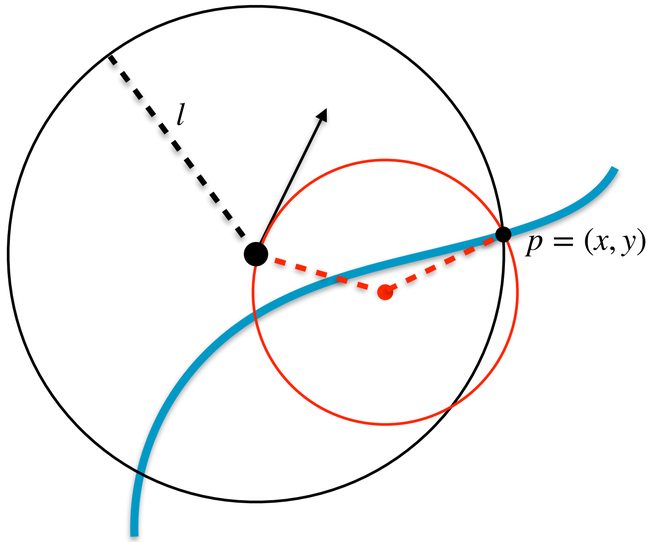


In RRT the distance was used, and the answer in this case the nearest node would be the one to the right.
In CL-RRT the Dubins distance is used, defined as the shortest path a with a minimal turning radius R_{min}. The nearest node would be the one to the left if R_{min} is sufficiently large. This is called the exploration heuristic.

CL-RRT: The controller

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



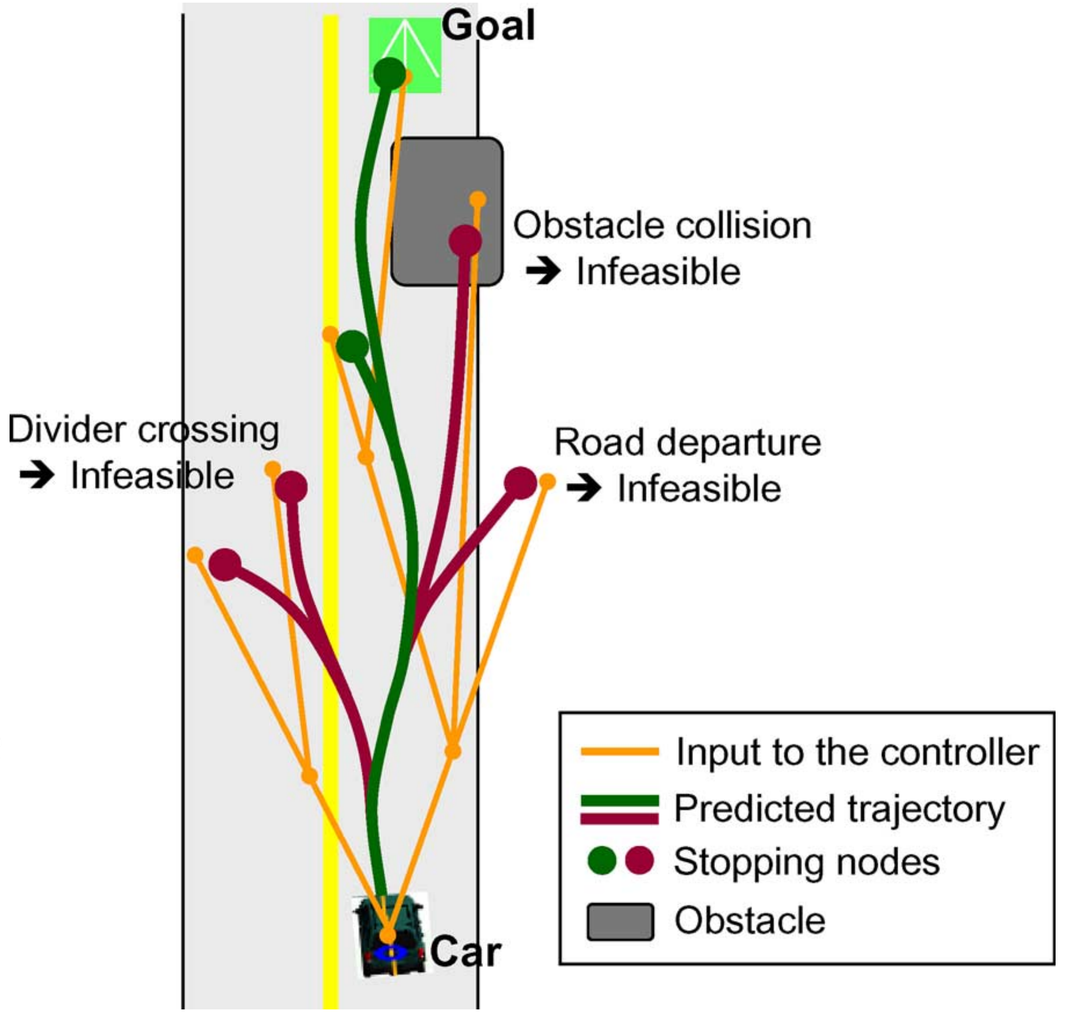


The next step is to describe how to construct reference paths for this controller.

CL-RRT: Expand the tree

Procedure to expand the (orange) tree

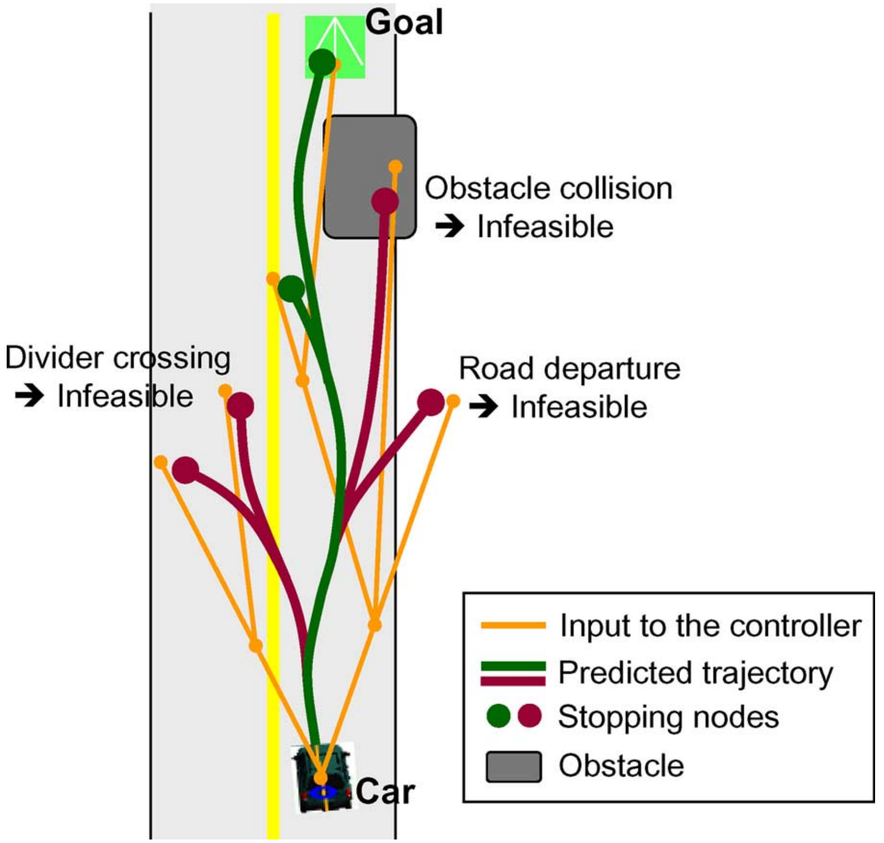
- Generate a sample position s.
- For each node q in the tree, in the order sorted by the Dubins distance $L_\rho(s)$, use the line segment between node q and sample s to extend the orange tree.
- Use the new reference path in the orange tree as input to the controller to generate a trajectory $\mathbf{x}(t)$, $t \in [t_1, t_2]$ 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 feasible, add them to the tree.

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

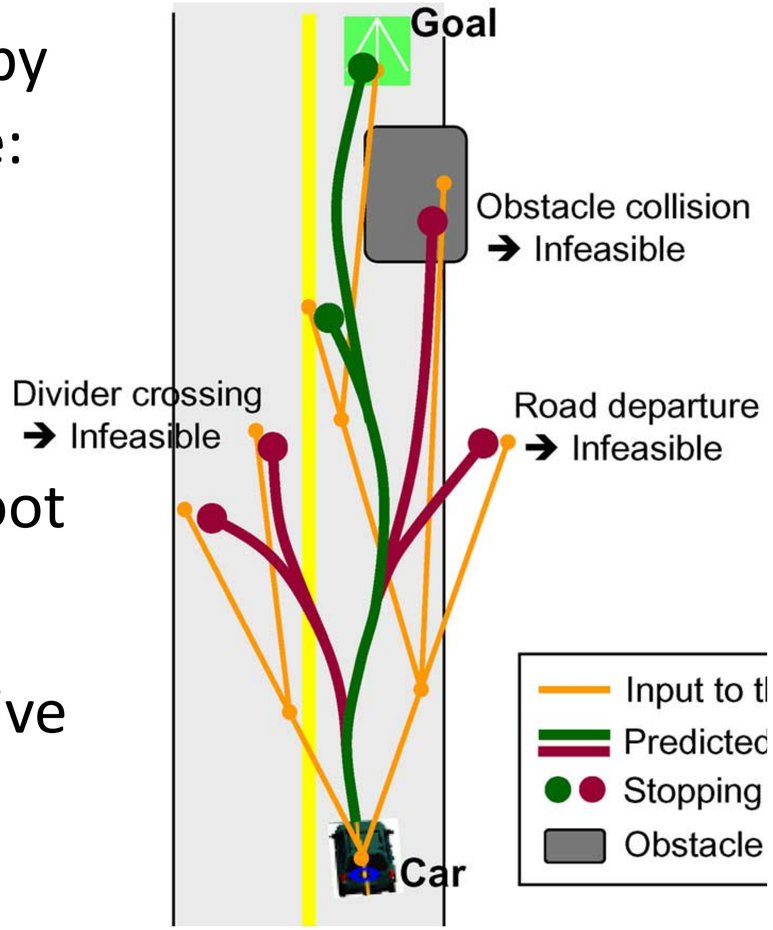


CL-RRT

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:

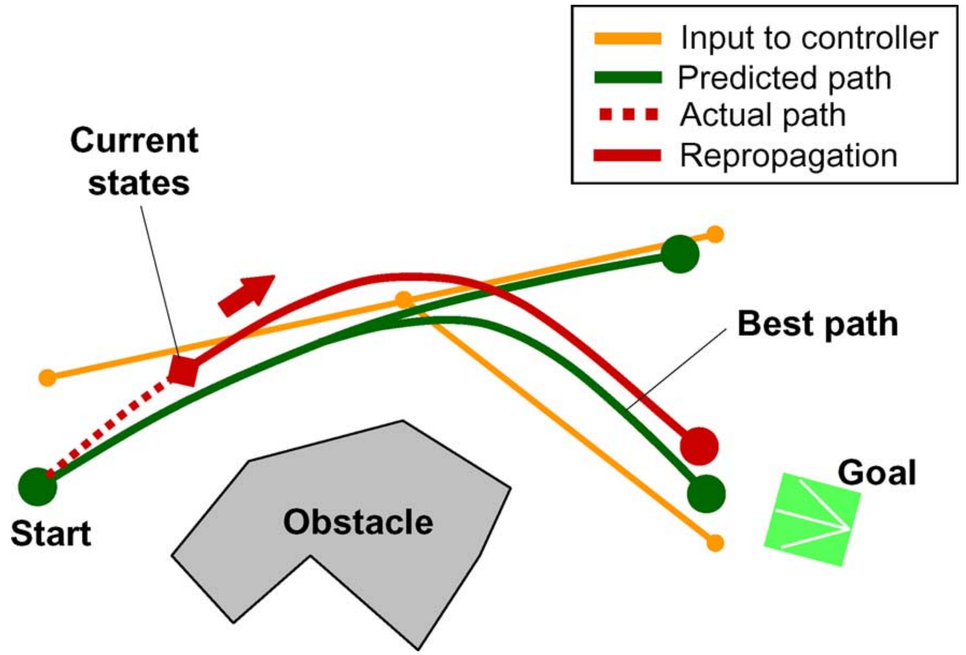
$$C_{total} = C_{cum}(q) + L_{\rho}(s)/v$$

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.



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