Parking for SVEA

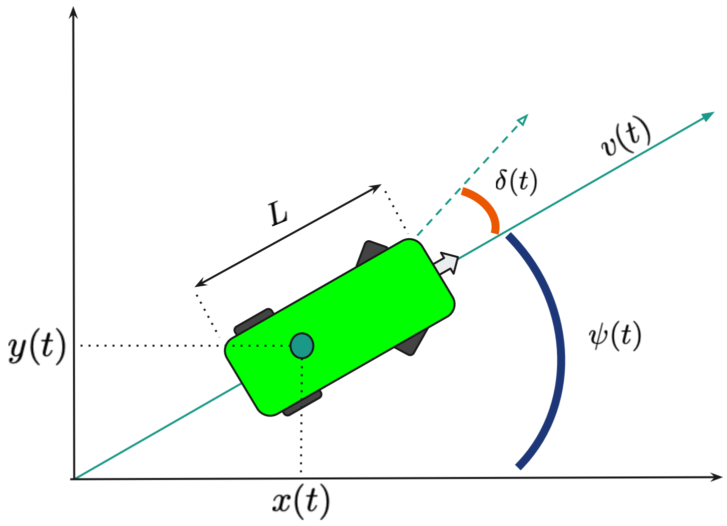
Team 3 - Christoffer, Devrat, Gustav and Sofia

Vehicle Modeling

Kinematic Bicycle Model

$$\dot{x} = v\cos\psi \qquad \dot{\psi} = \frac{v}{L}\tan\delta$$

$$\dot{y} = v\sin\psi \qquad \dot{v} = a$$



Why?

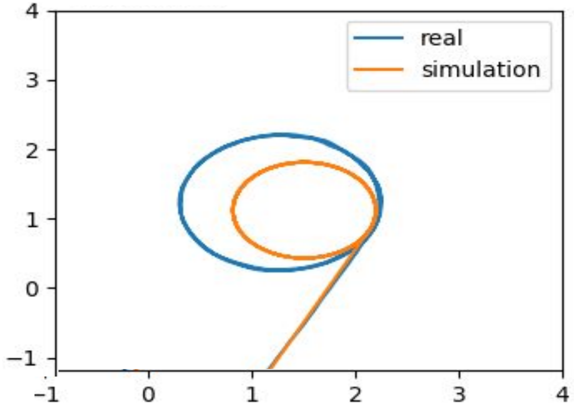
- Simplified car model
- Describes vehicle motion well
- Contains all states/inputs relevant for the task:
  - **States**: x, y, yaw, velocity
  - **Control inputs**: steering, acceleration(velocity)

Vehicle Modeling

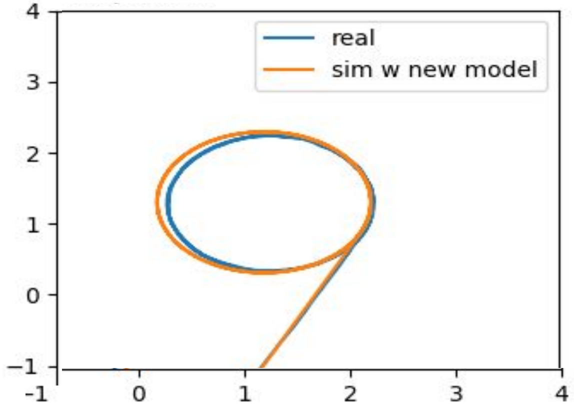
Tested accuracy of model by comparing simulation to real tests on SVEA

- Circle test
  - → true steering < steering input
  - → steering depends on speed
- Created function that takes difference in steering into account
- Function did more harm than good in the end, overcompensates steering when we should drive straight → didn't use it

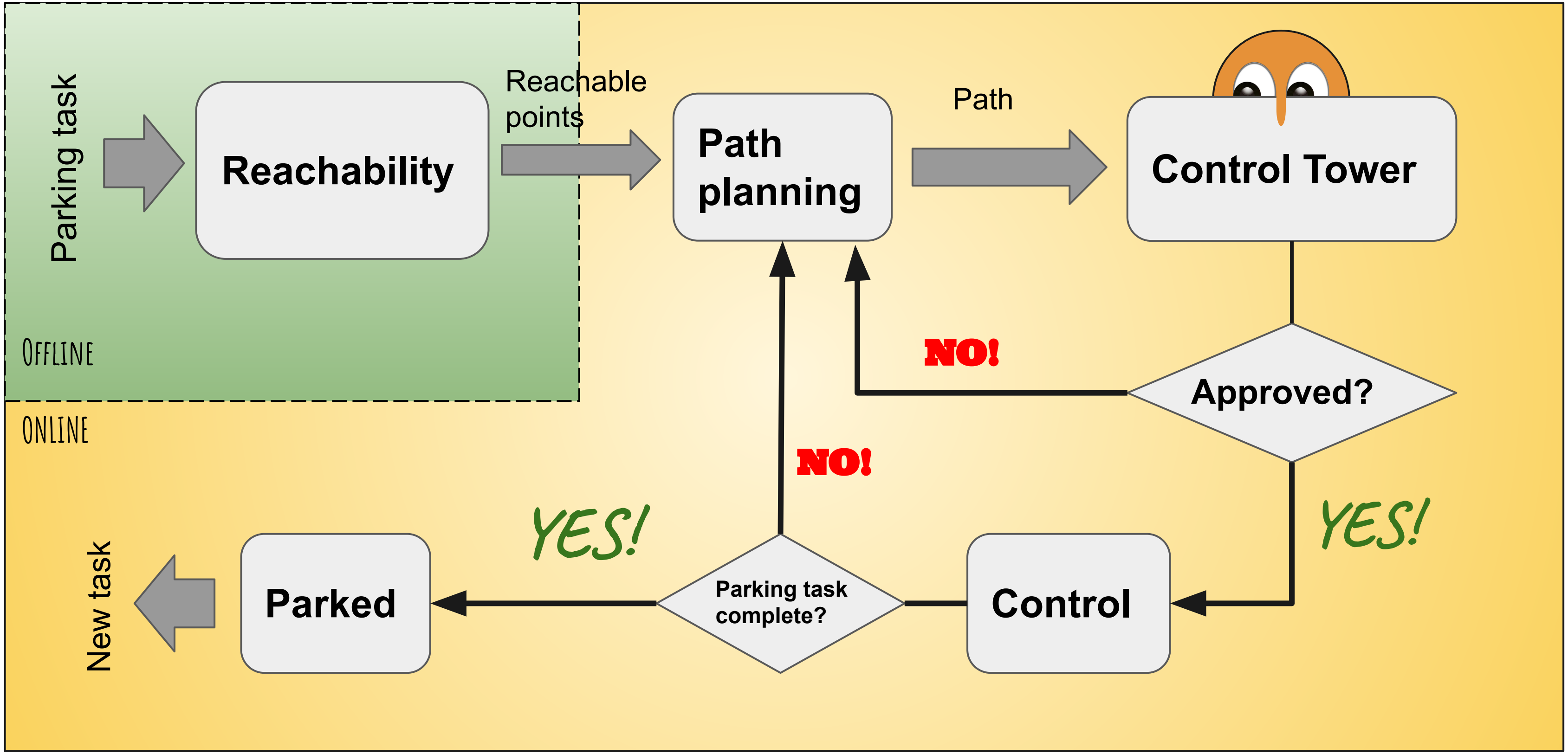
Before:



With function:



Information flow!



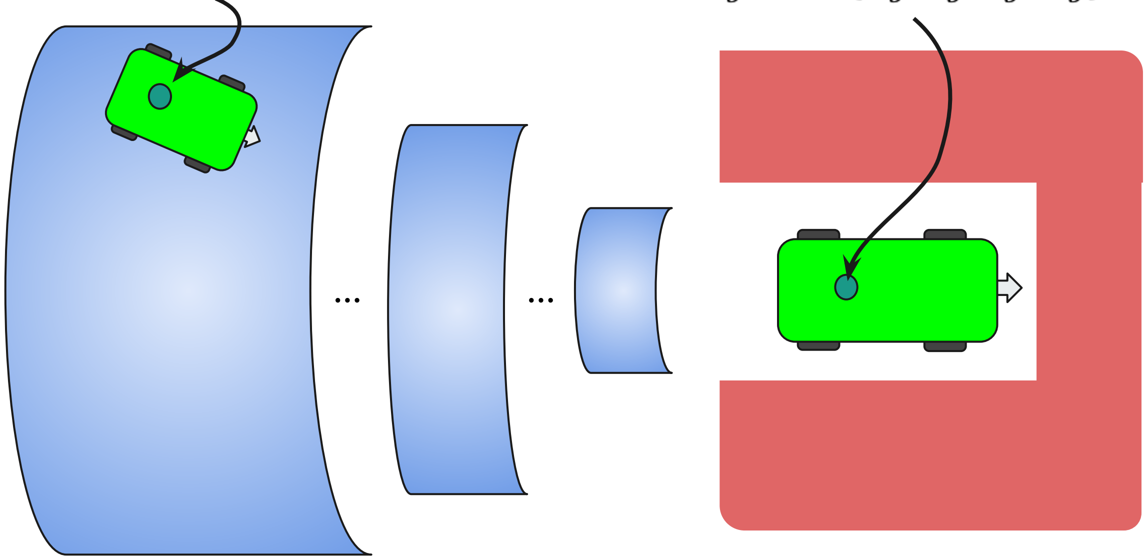
Parking task → **Reachability** → Goal point

Module: Reachability

Question:
Given a goal parking configuration, is there a way to obtain information about the starting configurations?
Answer:
Yes, with reachability analysis.

$$\mathbb{X}_{start} = \{x_s, v_s, v_s, \psi_s\}$$

$$\mathbb{X}_{goal} = \{x_g, y_g, v_g, \psi_g\}$$

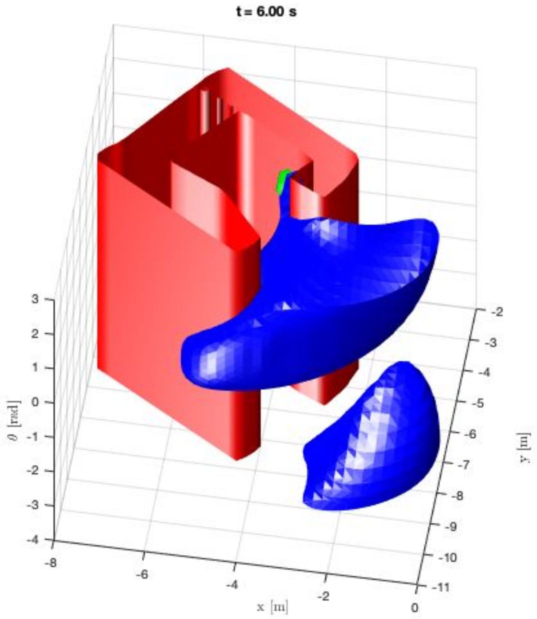


Module: Reachability

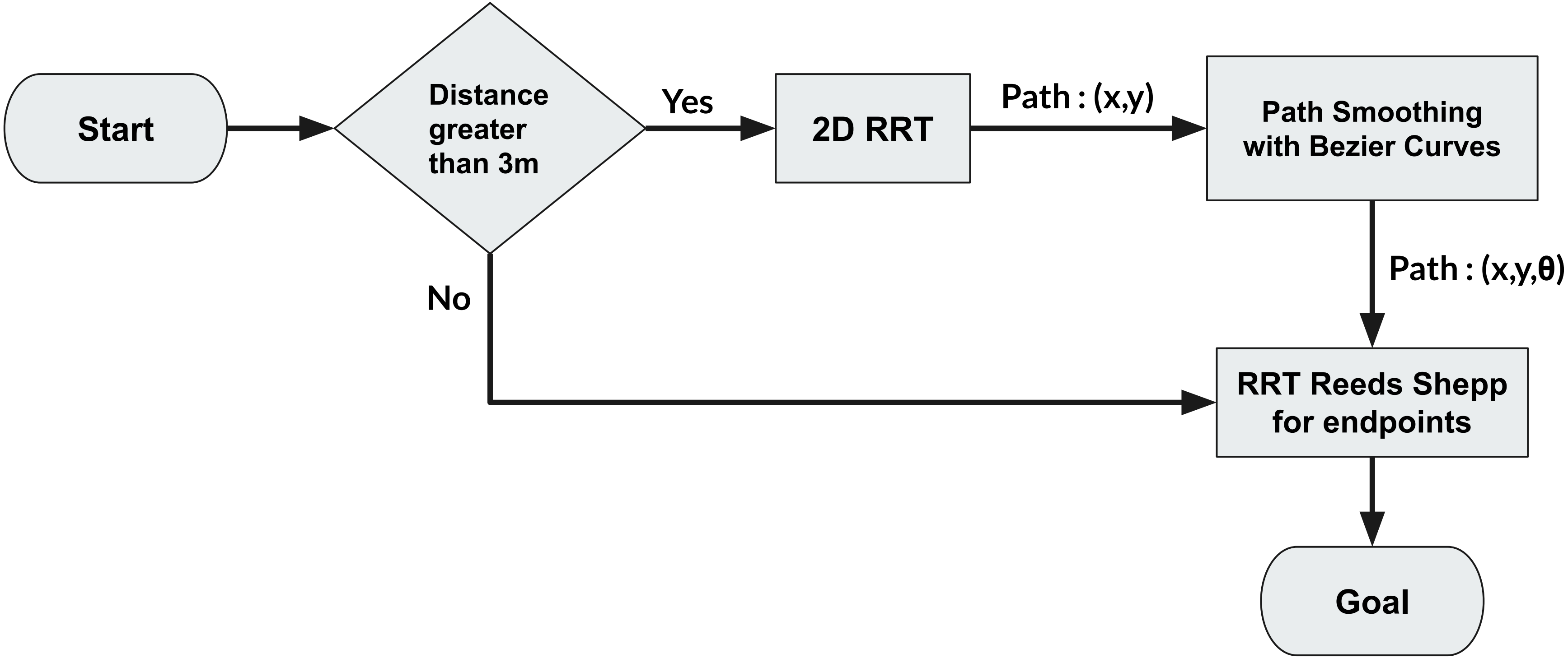
Blue blob: From what position and angle can I start?

Green blob: The parking spot configuration.

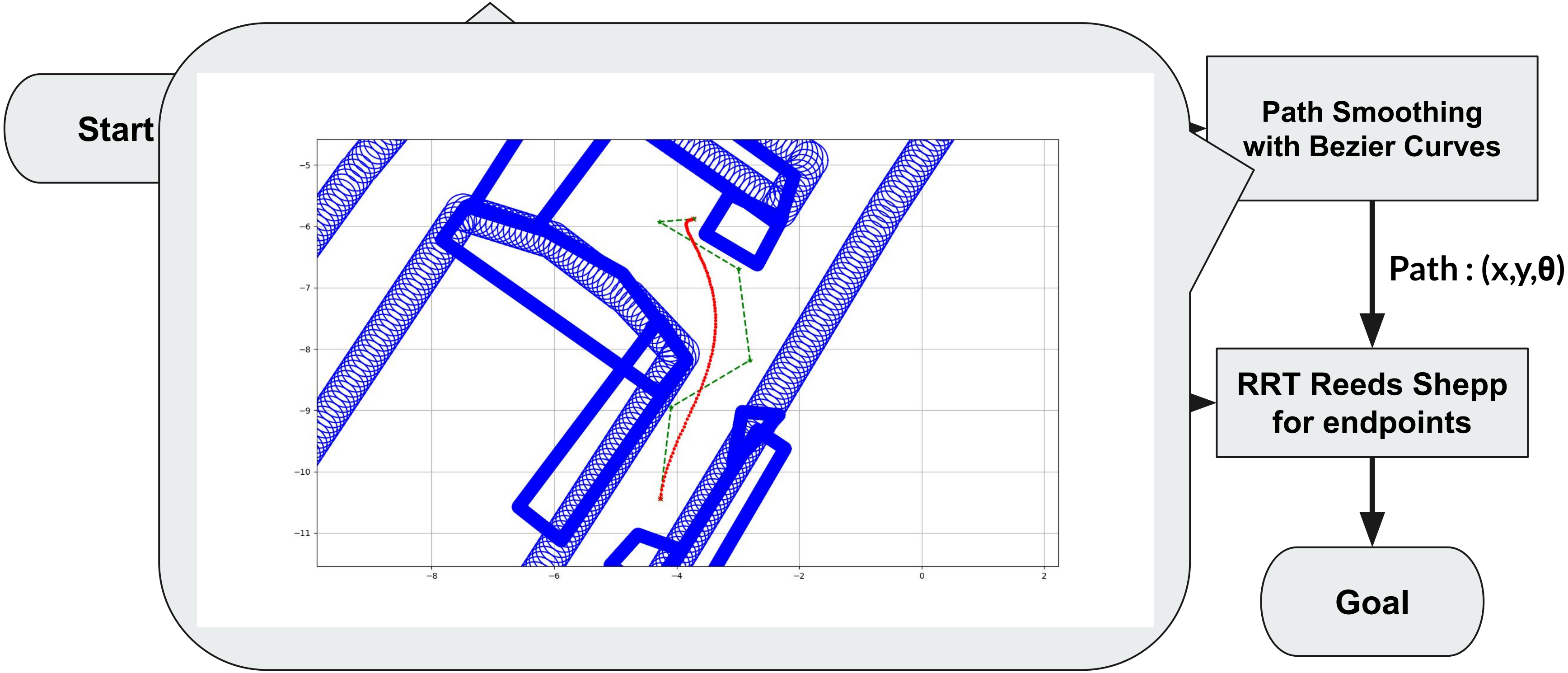
Red wall: Obstacles – The configurations we should not reach.



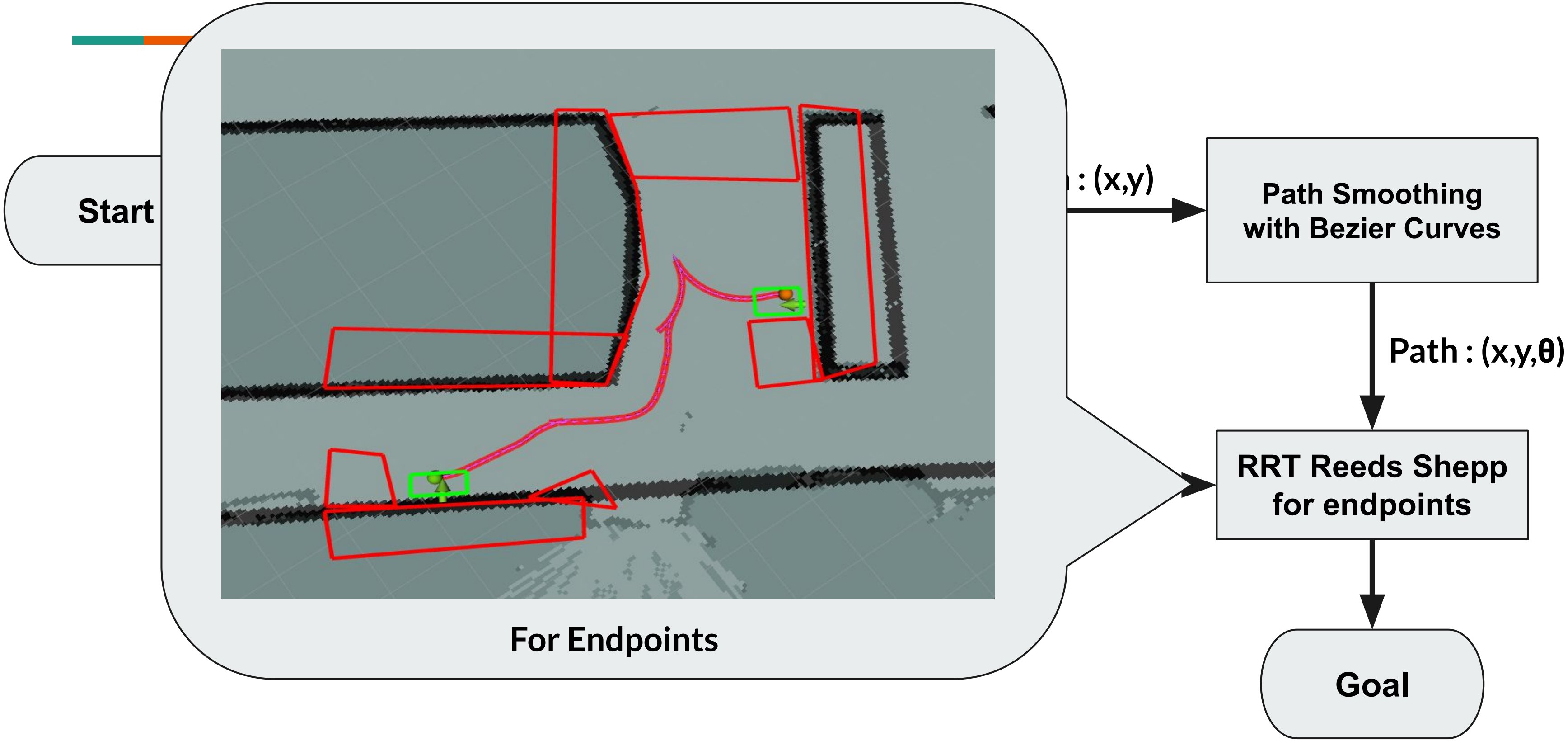
Module: Planning



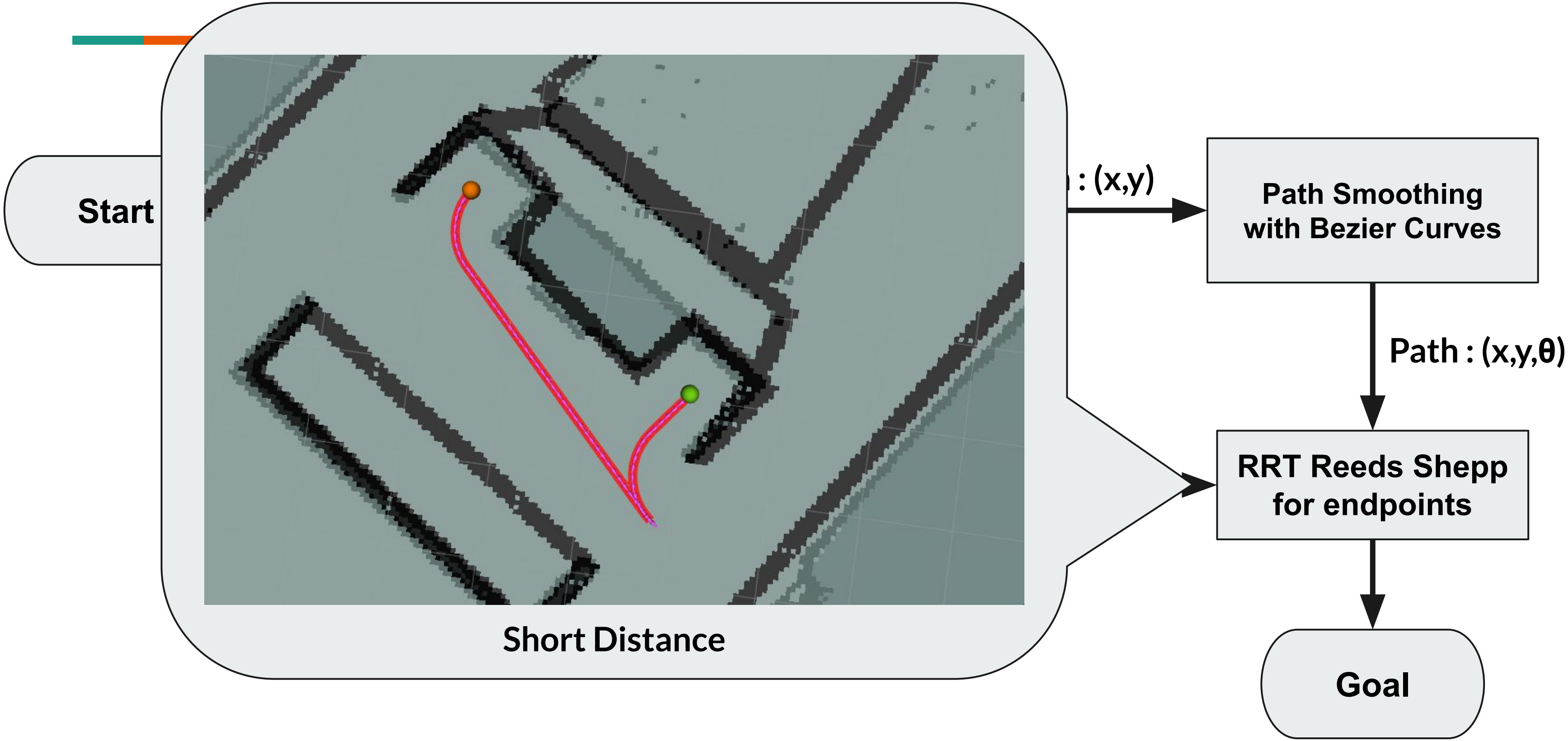
Module: Path Planning



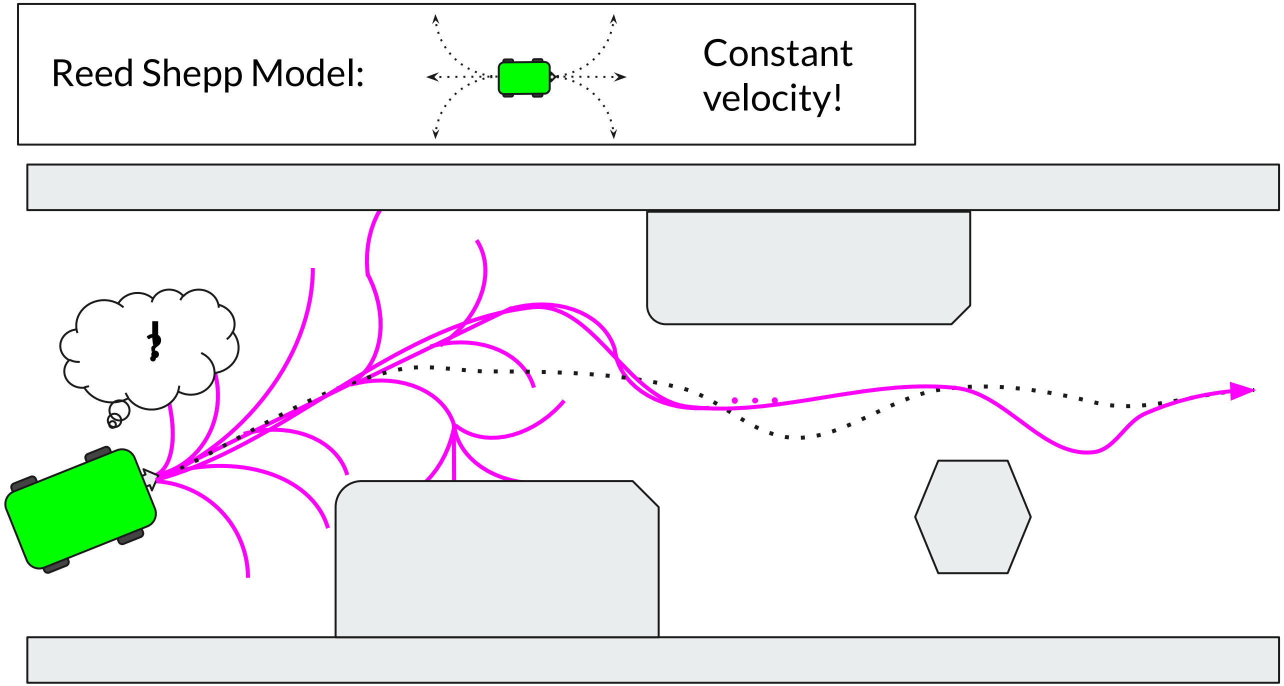
Module: Path Planning



Module: Path Planning



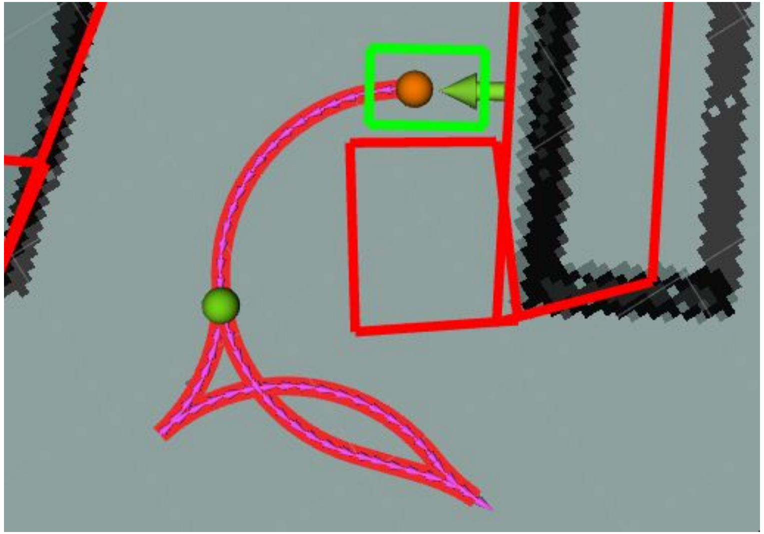
Module: Path Planning



Module: Path Planning

Challenges with RRT:

- Fast, but at what cost?
  - → Randomness makes the paths somewhat unpredictable
  - → Paths can be unreasonable
- Possible Solutions:
  - → Post processing of Paths
  - → Optimation such as RRT Star, etc.
    - ■ Problem: Too slow
  - → Many RRT variants are available and all do it somewhat differently
    - ■ Problem: Hard to choose within available time
- Our Solution:
  - → Having redundant paths



Module: Controller

General idea: Use MPC for long distance path following and parking

- Plan the car's movement over horizon 20, find control inputs minimizing a cost function

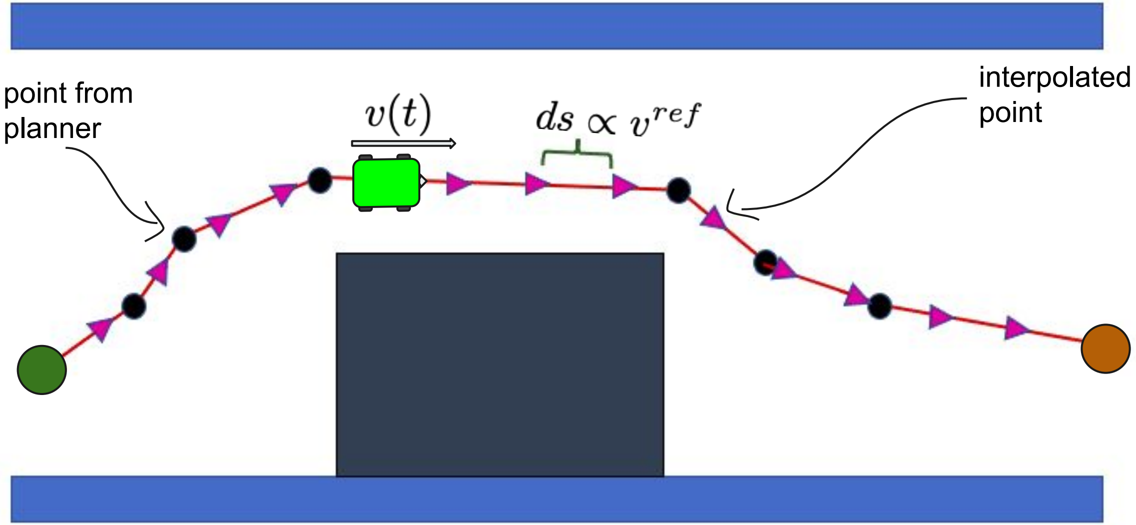
$$J_t(\mathbb{X}, u) = \underbrace{}_{\text{Cost at time: t}} \sum_{k=0}^{N-1} \left(e_k^T Q e_k + u_k^T R u_k + (u_k^{vel} - v_k)^T G (u_k^{vel} - v_k) \right) + e_N^T P e_N ,$$

- $J_t(\mathbb{X}, u)$: Cost at time: t
- $e_k^T Q e_k$: Follow the given path!
- $u_k^T R u_k$: Drive and steer conservatively!
- $(u_k^{vel} - v_k)^T G (u_k^{vel} - v_k)$: Hold a smooth speed!
- $e_N^T P e_N$: Try to end at the horizon end state!

where $e_k = x_k^{ref} - x_k$

Module: Controller

- Points from planner **not** uniformly spaced
  → irregular/too fast speed

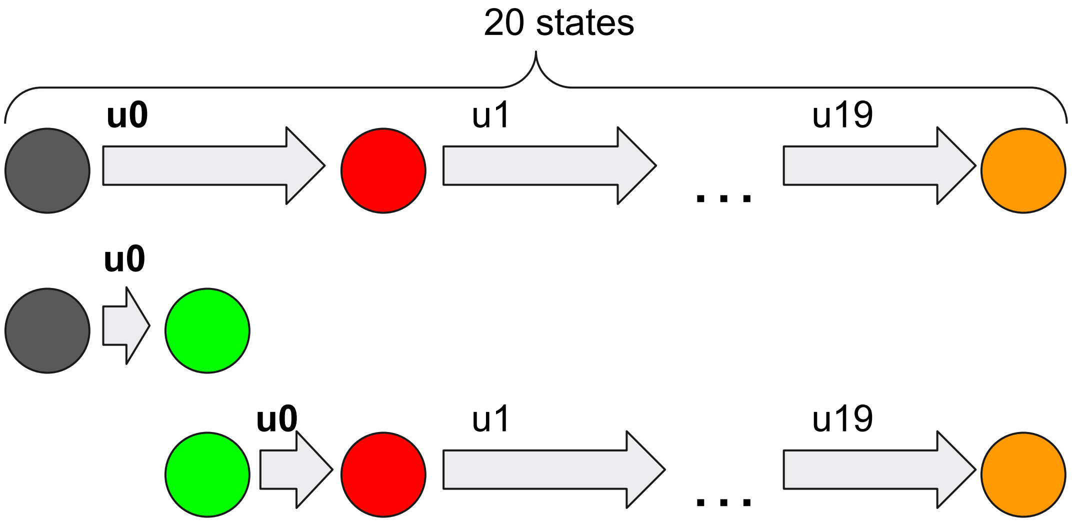


Control velocity:

- Interpolate points for path s.t. the distance between following points are the same
- Given a reference velocity, determine the interpolation distance

Module: Controller

Don't always move horizon at each step → move horizon when next true state is close to next reference state → might have to plan over the same reference states multiple times



1. Plan over horizon 20 from the grey state

2. Execute first control input, reach green state

3. If the green state is far away from red state → plan over same reference states one more time, else move to next

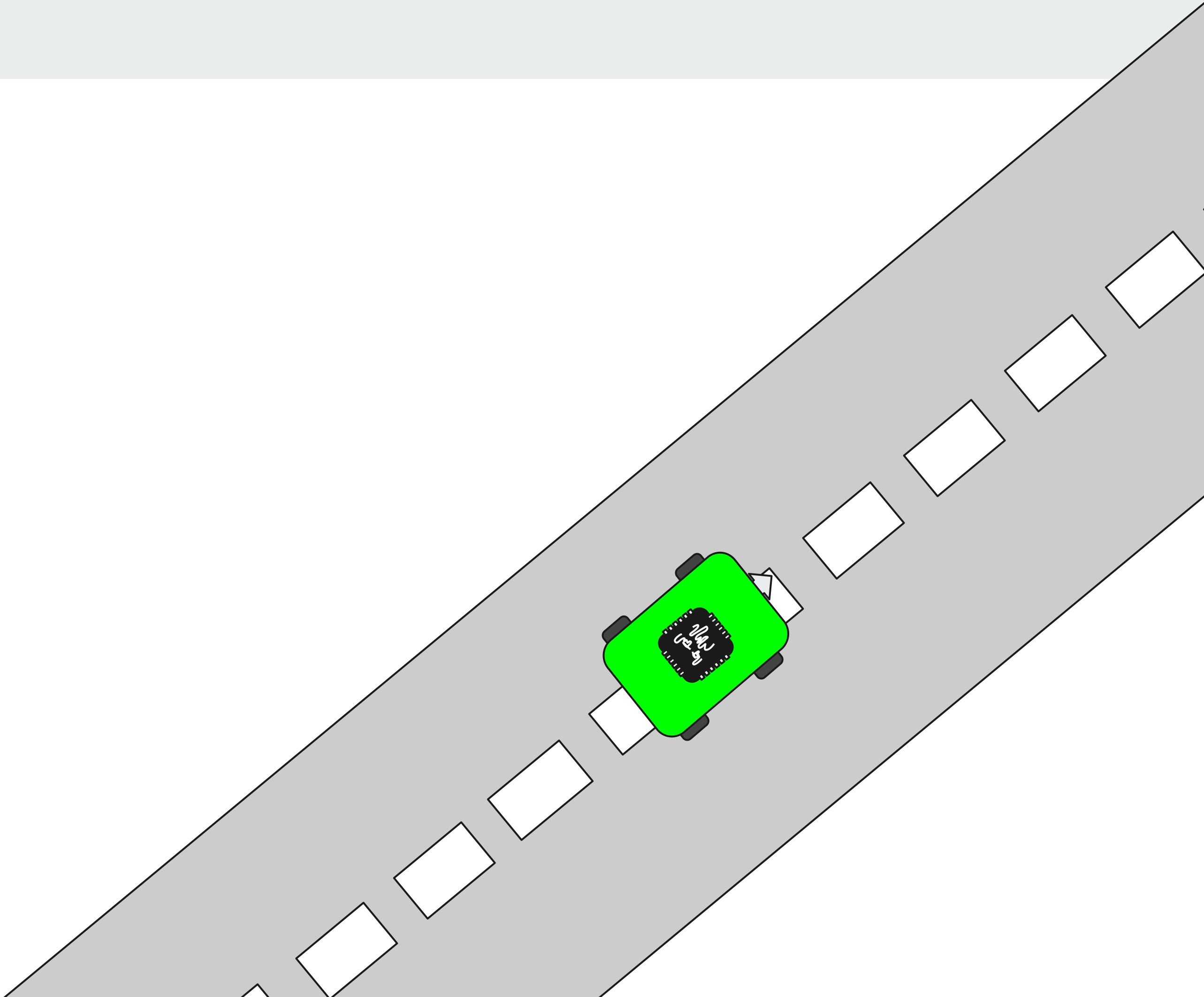
Module: Controller

Why MPC?

First idea: use Stanley controller for long path following and MPC for parking

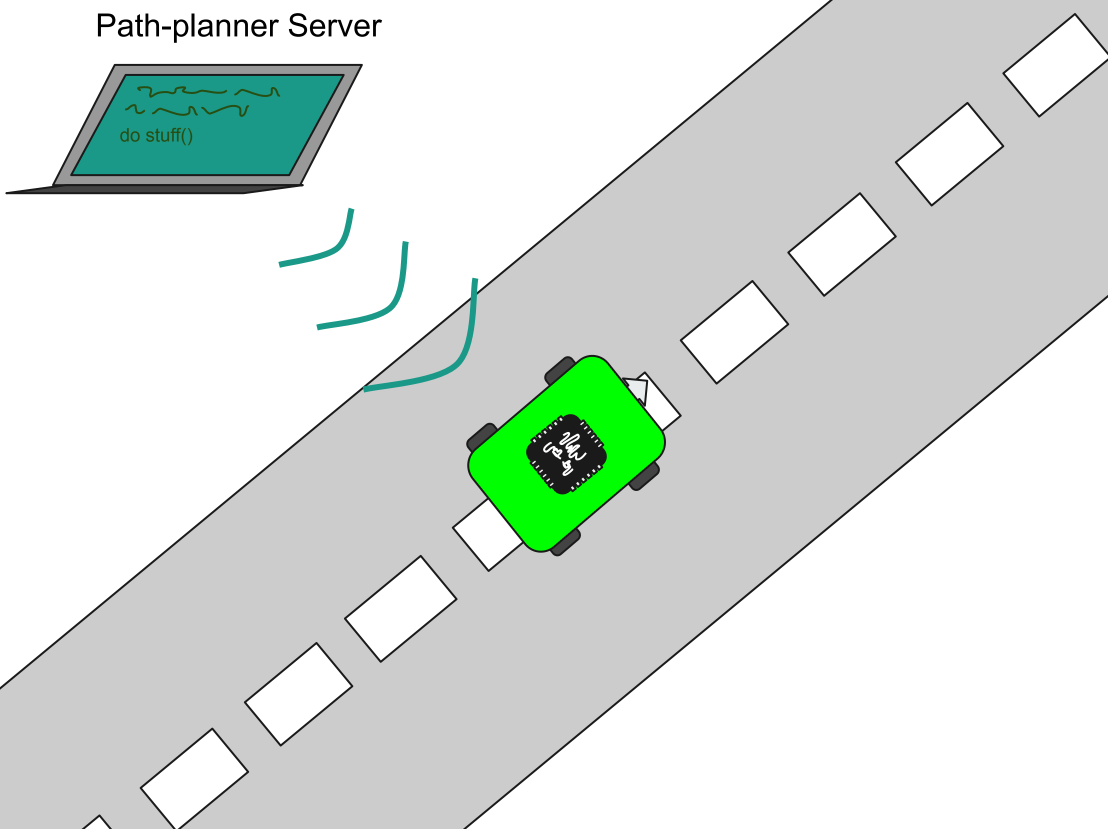
→ MPC was faster than expected and worked very well → used MPC for all path following tasks

Implementation



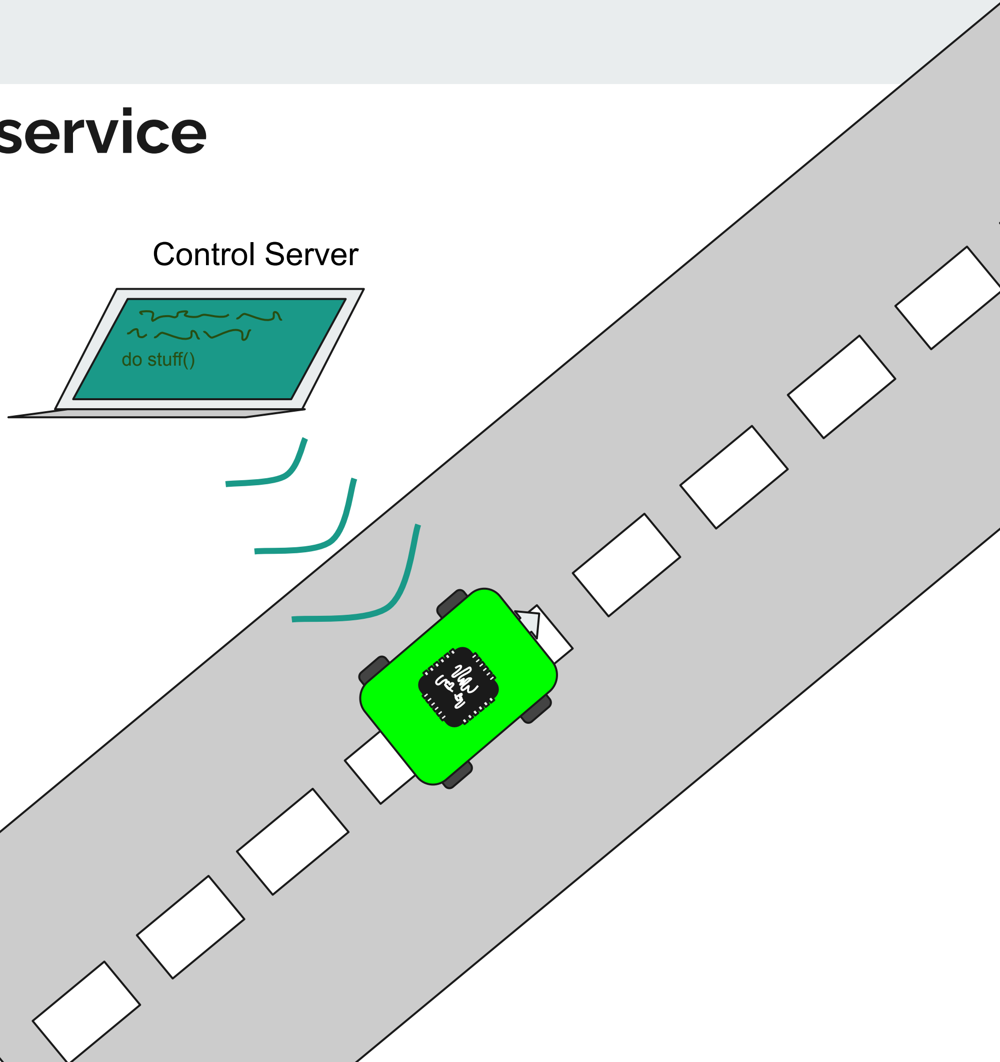
Implementation: Planner services

- Why? Approx. 10 times faster planning.
- Implemented as server with option to abort planning from the client.
- Use infrastructure to account for planner issues: **multiple planners simultaneously**



Implementation: Controller service

- Why? Approx. 6 times faster MPC.
  - Makes the control feasible in real time
- Messages to the controller
  - Setup: trajectory and controller type
  - Get control: State → Control
- Planned for backup controller on SVEA



Thank you!

Questions?

Implementation: State Machine

High level state machine

Low level state machine