Autonomous Driving Path Planning under Temporal Logic Specifications

Akshay Dhonthi, Nicolas Schischka, Ernst Moritz Hahn, Vahid Hashemi

SBMF 2024
Brazil | 04-12-2024

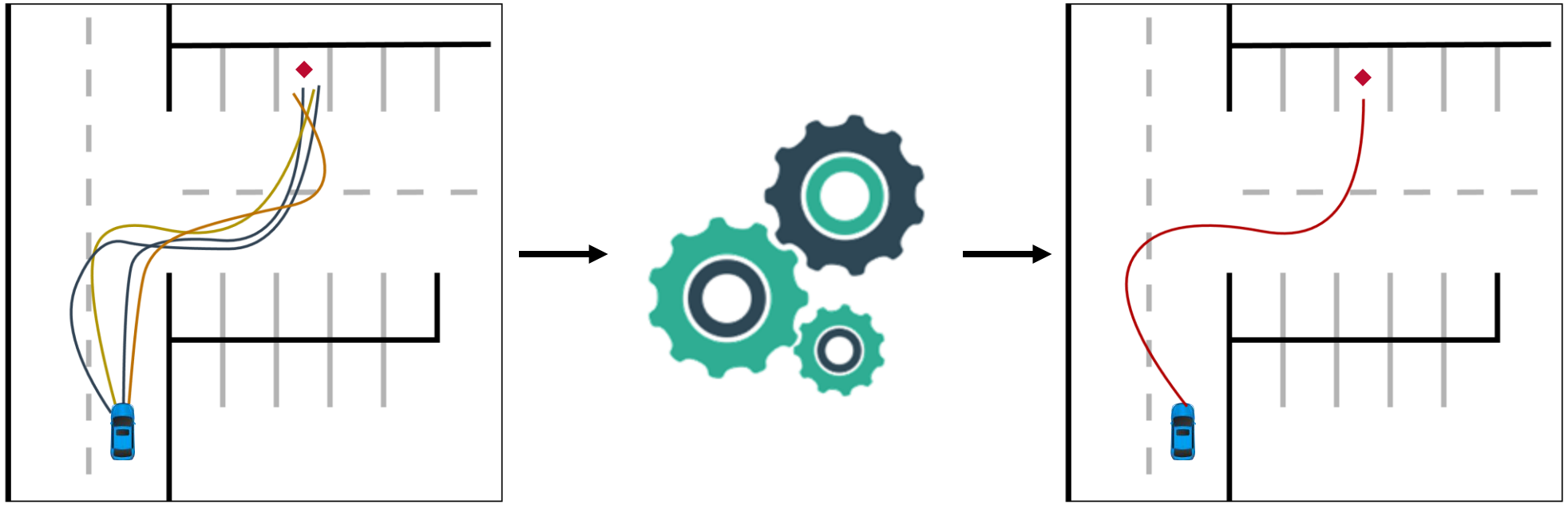
UNIVERSITY OF TWENTE.

Introduction

Autonomous Driving Path Planning under Temporal Logic Specifications

An Overview

Learning from Demonstration



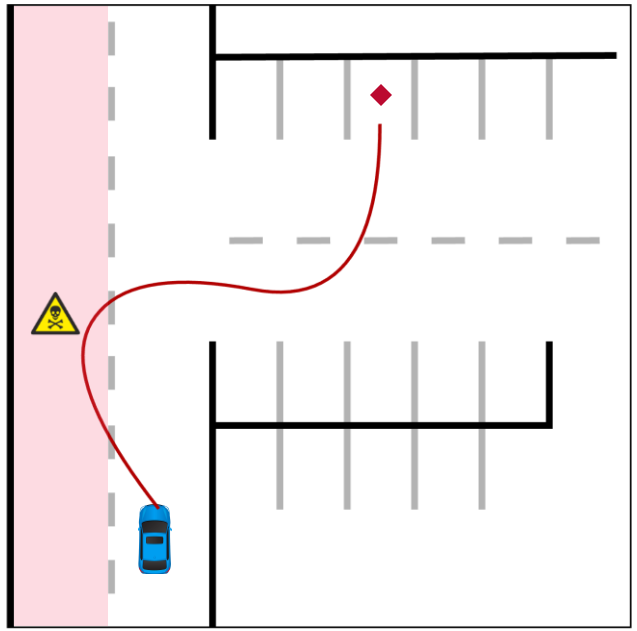
Demonstrate the path **Learn from the demos** **Reproduce the path**

Introduction

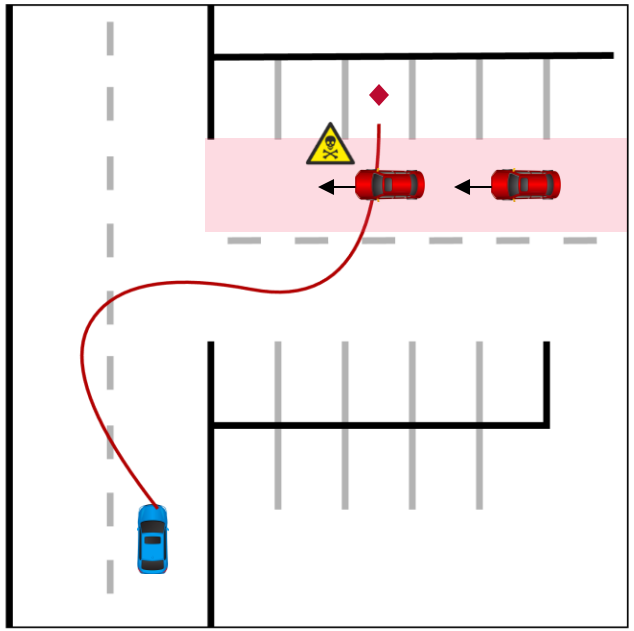
Autonomous Driving Path Planning under Temporal Logic Specifications

The Problem

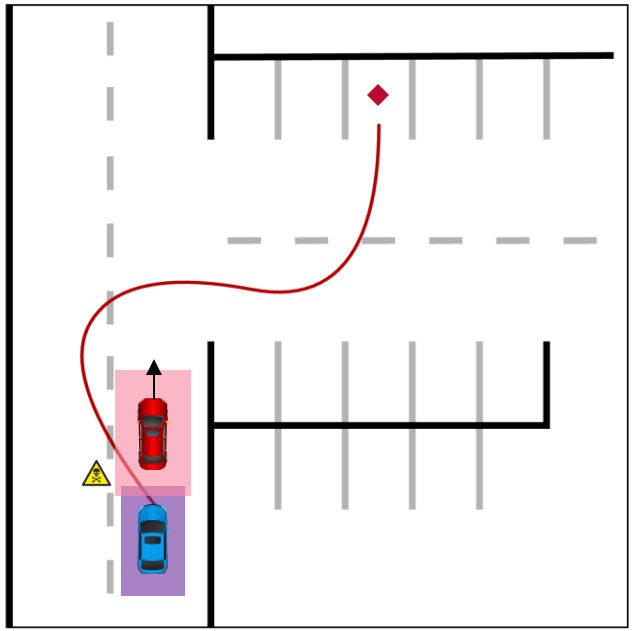
Learning from Demonstration



(1) May not follow **traffic rules**



(2) May **collide** with oncoming traffic

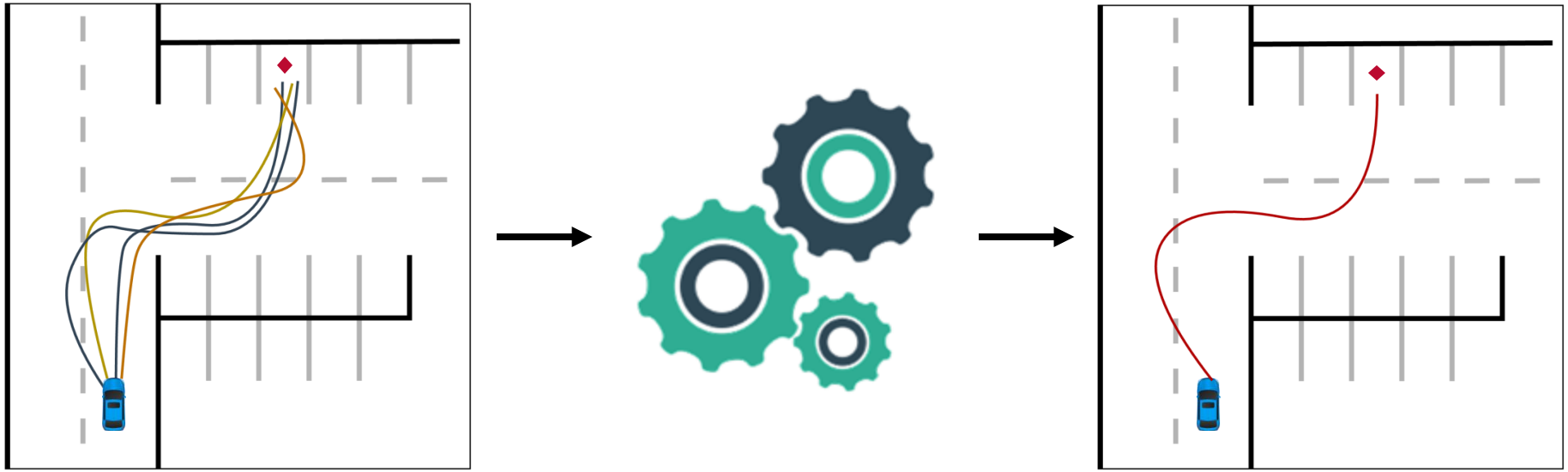


(3) May violate road **safety constraints**

Introduction

Autonomous Driving Path Planning under Temporal Logic Specifications

IDEA !

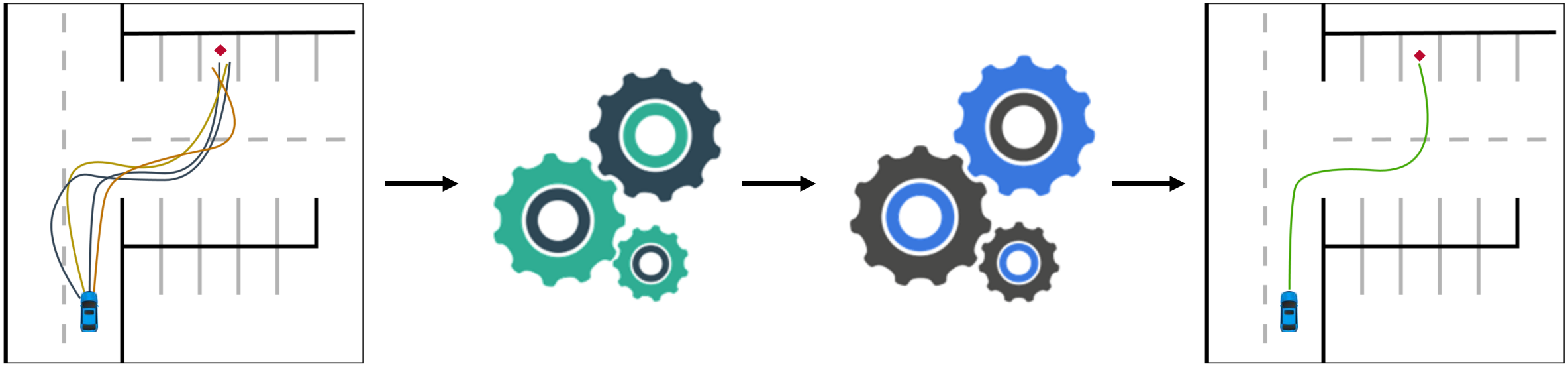


Demonstrate the path | **Learn from the demos** | **Reproduce the path**

Introduction

Autonomous Driving Path Planning under Temporal Logic Specifications

IDEA !



Demonstrate the path | **Learn from the demos** | **Refinement via STL constraints** | **Reproduce the path**

Introduction

Autonomous Driving Path Planning under Temporal Logic Specifications

Signal Temporal Logics

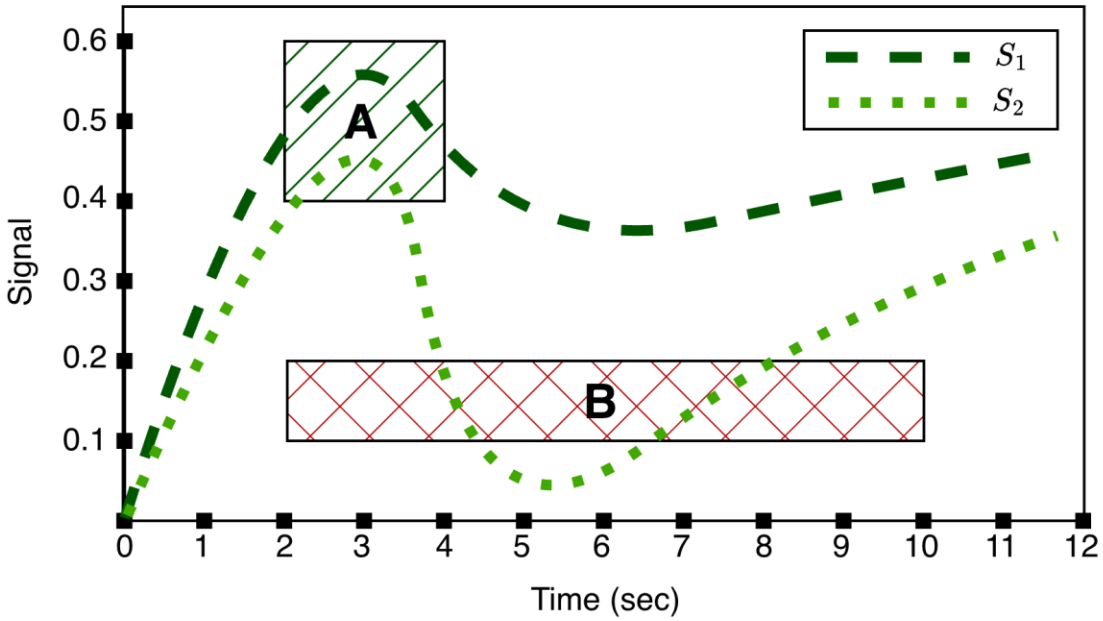
Refinement via
STL constraints

- STL specifications can be quantified to a real-number and it is called **Robustness Degree**.
- Robustness degree defines how well a specification is satisfied

Example

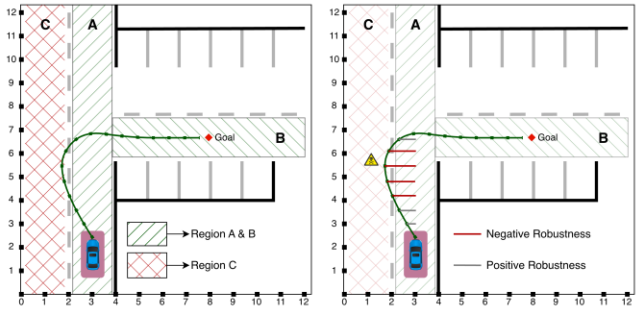
During [2, 4] seconds, visit **Region A** and during [2, 8] seconds, visit **Region B**

Signal S1 may have **Negative** Robustness Degree
Signal S2 may have **Positive** Robustness Degree

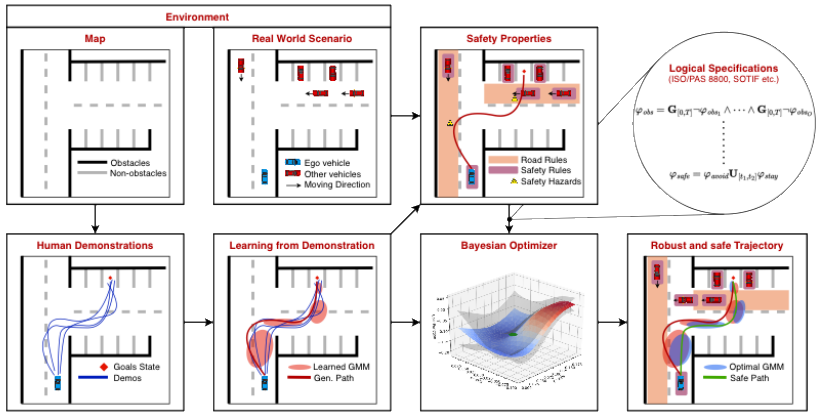


Introduction

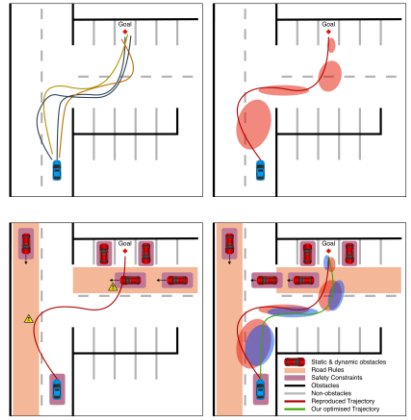
Summary



Approach



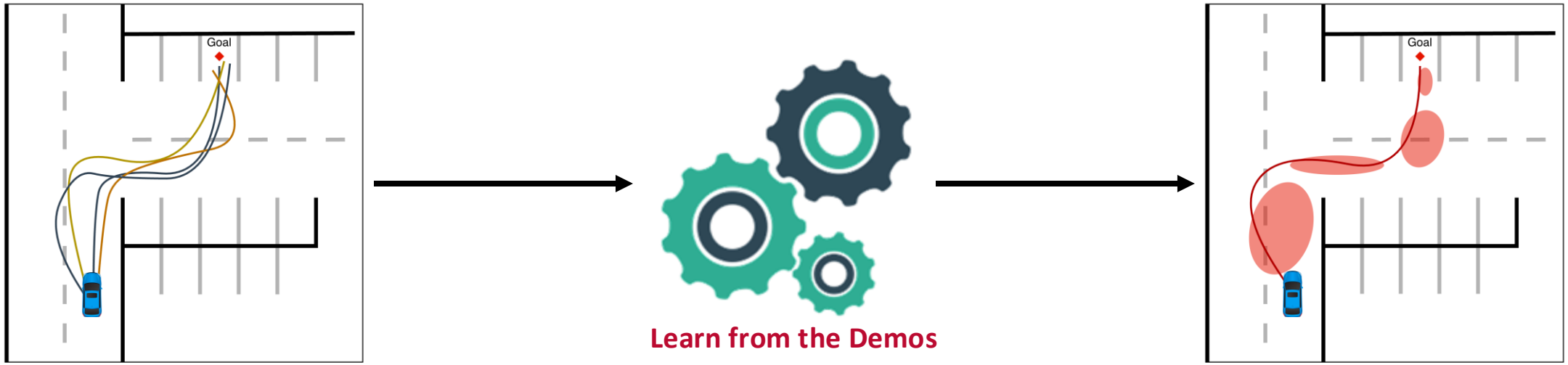
Algorithm



Experiments

Approach

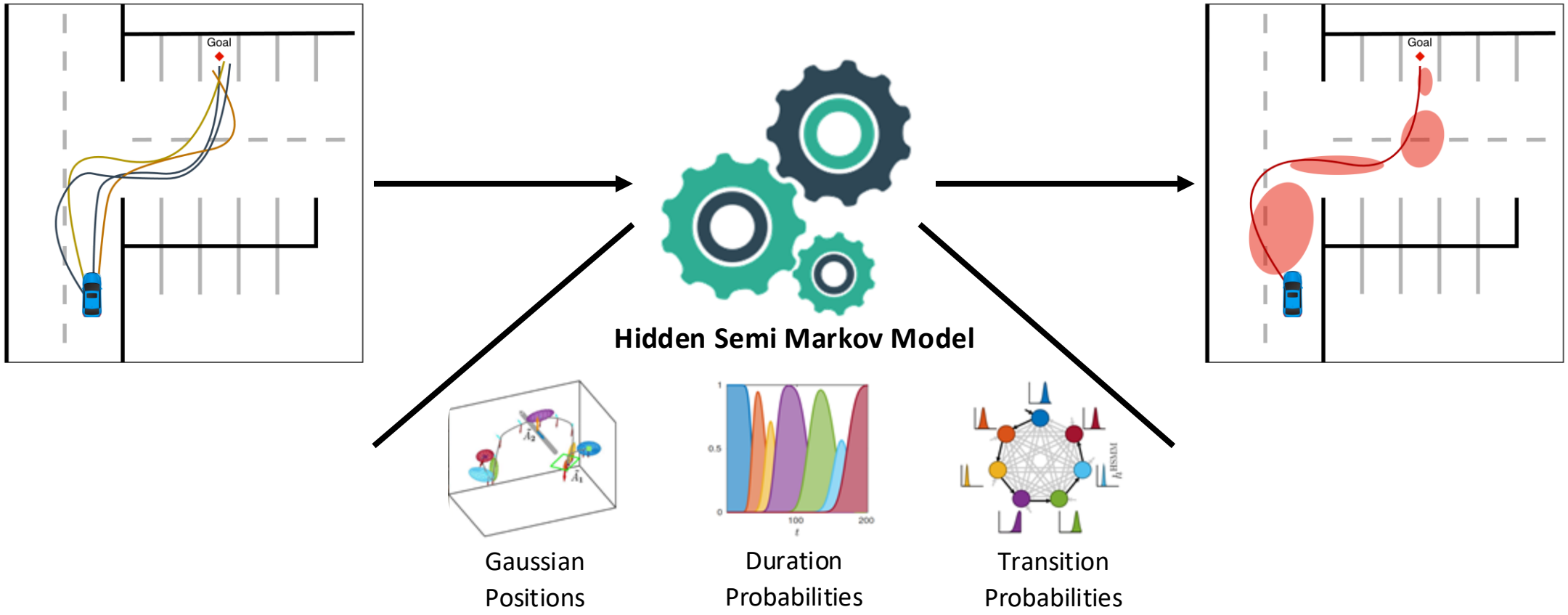
Learning from Demonstration



Approach

Learning from Demonstration

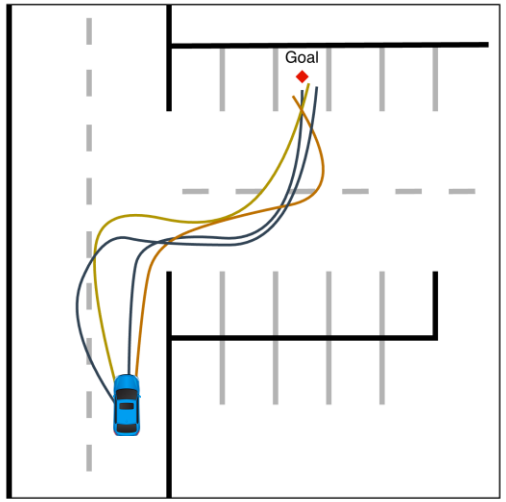
Model and its parameters



Approach

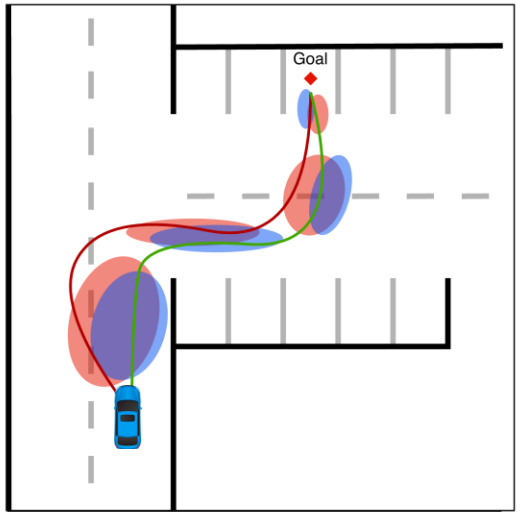
Learning from Demonstration

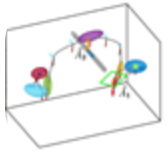
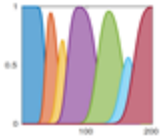
Model and its parameters



HSMM

Parameter Refinement

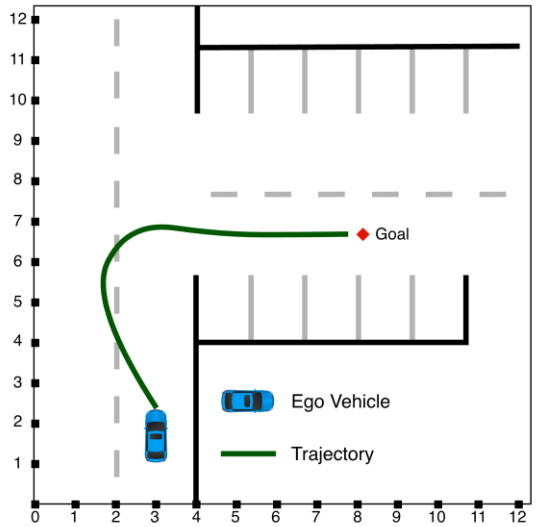


Gaussian Positions		Update gaussian positions in x- and y- directions
Duration Probabilities		Update how to long to stay in each gaussian
Transition Probabilities		Skip a gaussian if it is no longer useful for the task

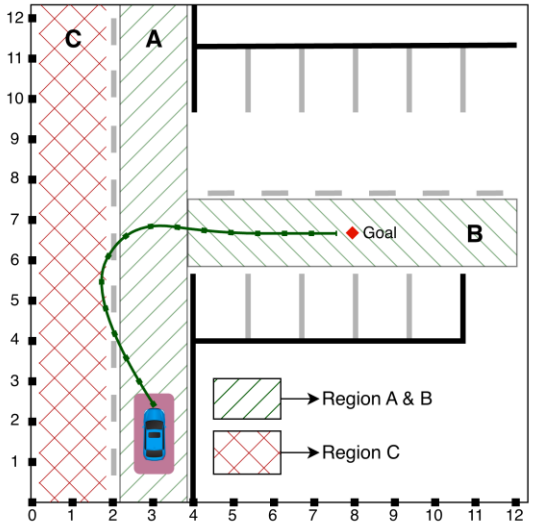
Approach

Proposed Method

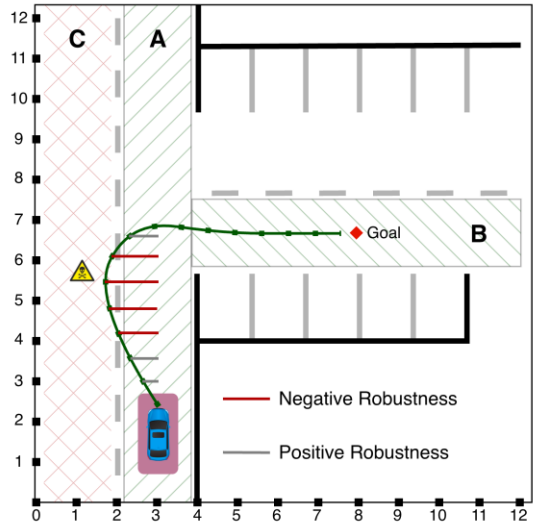
Signal Temporal Logics (STL)



Obtained Trajectory



Discretization



Robustness Degree Calculation

= -1.5

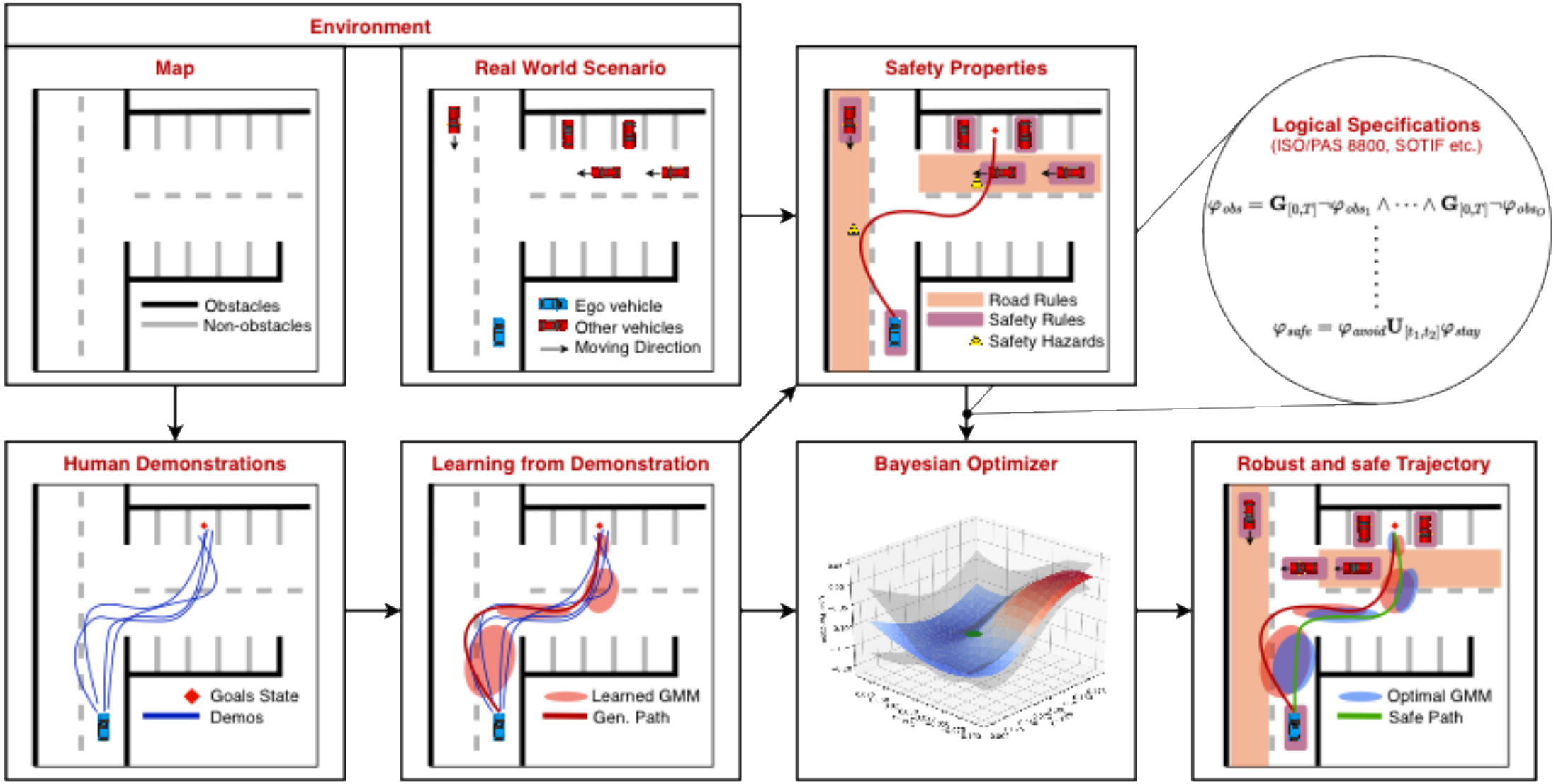
Road Rules

STL Specification

*"The trajectory must **avoid Region C** **AND** stay **inside Region A and B**"*

$$\varphi_1 = \boldsymbol{G}_{[0,\,10]}\varphi_A \wedge \boldsymbol{G}_{[10,\,20]}\varphi_B \wedge \boldsymbol{G}_{[0,\,10]}\neg\varphi_C$$

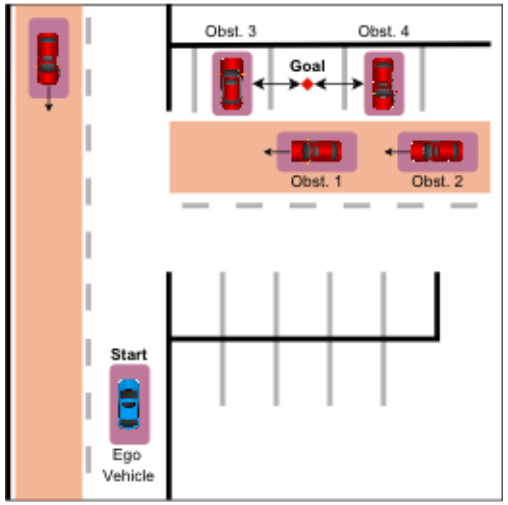
Algorithm



Experiments

Scenario A

Valet Parking while avoiding dynamic vehicles



"The trajectory must avoid Obst 1 and Obst 2 AND maintain maximum distance between the Obst 3 and Obst 4"

$$\varphi_A = \underbrace{\boldsymbol{G}_{[0,20]}\neg\varphi_{Obs_O}}\ \wedge\ \underbrace{\boldsymbol{G}_{[0,20]}\varphi_{rules}}\ \wedge\ \underbrace{\boldsymbol{F}_{[16,20]}\varphi_{safe}}$$

Note: Temporal constraints are extracted from the observations

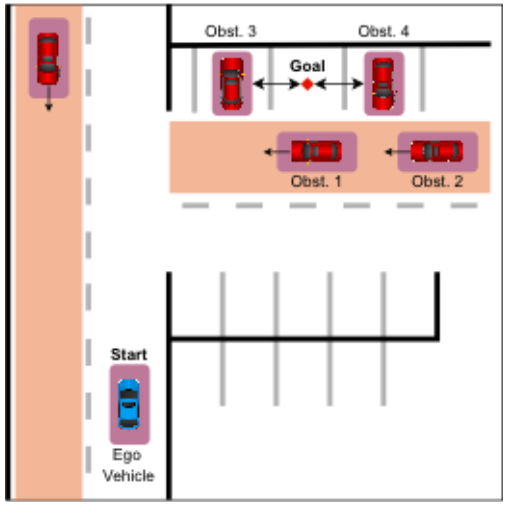
Two Observation Experiments:

1. **Full Observation:** We know the position of each vehicle and their respective future positions
2. **Partial Observation:** The ego vehicle can observe for a defined fixed range

Experiments

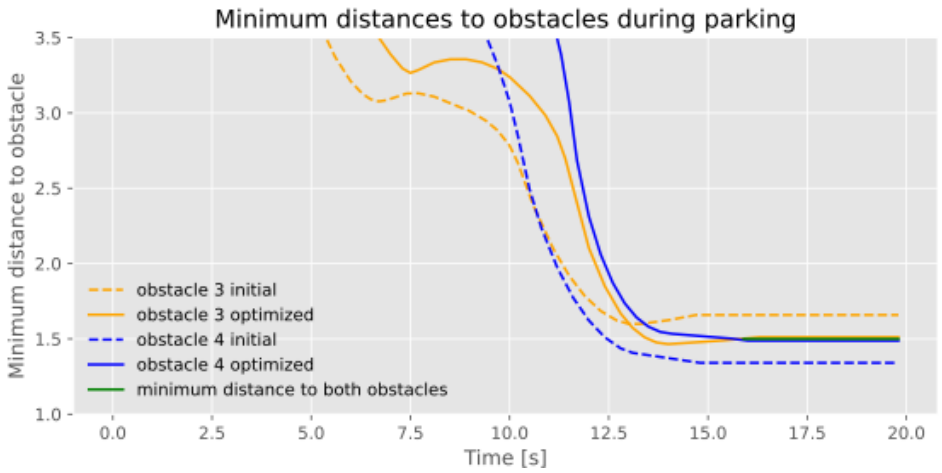
Scenario A

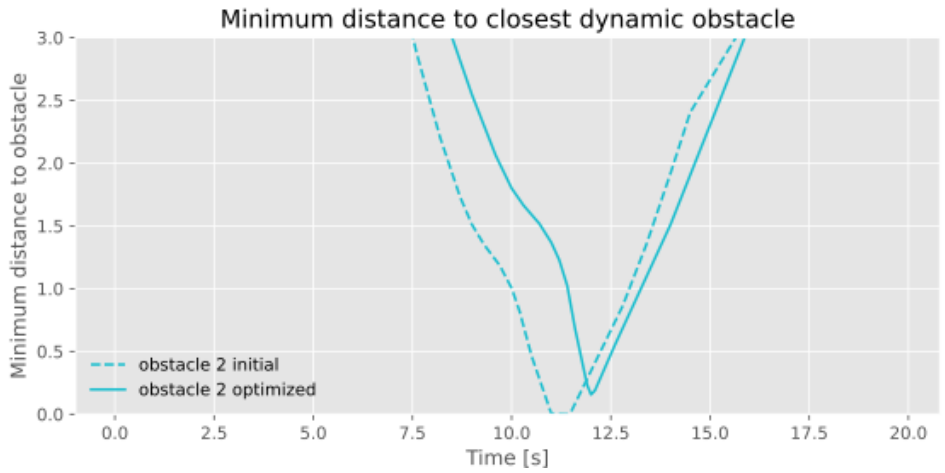
Valet Parking while avoiding dynamic vehicles



"The trajectory must avoid Obst 1 and Obst 2 AND maintain maximum distance between the Obst 3 and Obst 4"

$$\varphi_A = \boldsymbol{G}_{[0,\,20]} \neg \varphi_{Obs_O} \wedge \boldsymbol{G}_{[0,\,20]} \varphi_{rules} \wedge \boldsymbol{F}_{[16,\,20]} \varphi_{safe}$$

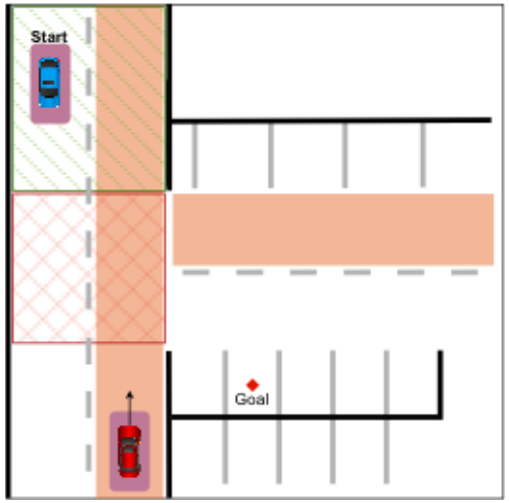




Experiments

Scenario B

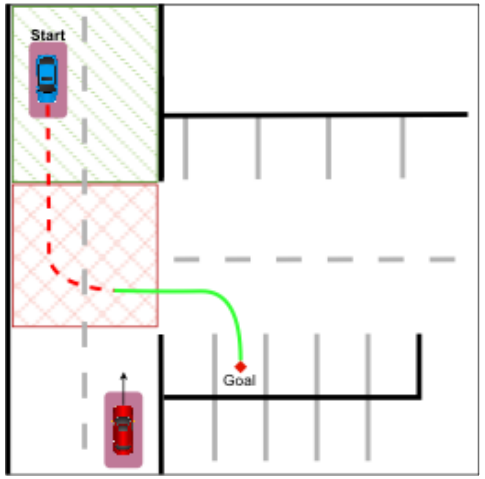
Valet Parking while adhering to traffic signal



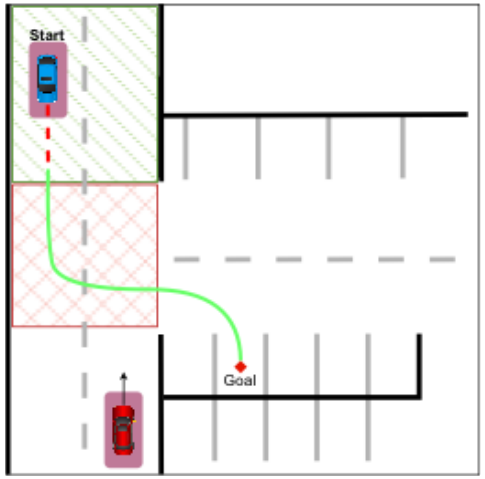
"The trajectory must avoid the Obstacle AND stay behind the junction in the first four seconds when the traffic light is red"

$$\varphi_B = \boldsymbol{G}_{[0,20]} \neg \varphi_{Obs_O} \wedge \boldsymbol{G}_{[0,20]} \varphi_{rules} \wedge \varphi_{avoid} \boldsymbol{U}_{[0,4]} \varphi_{safe}$$

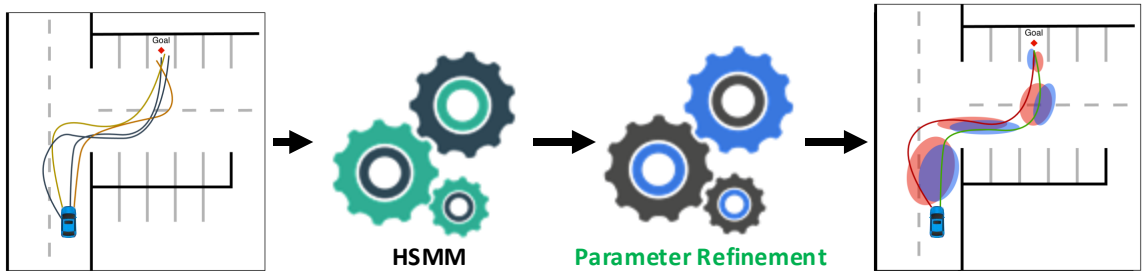
Before

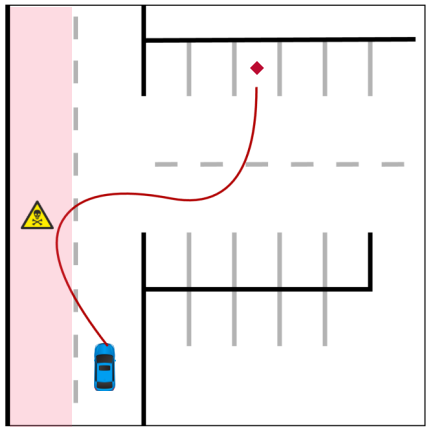


After



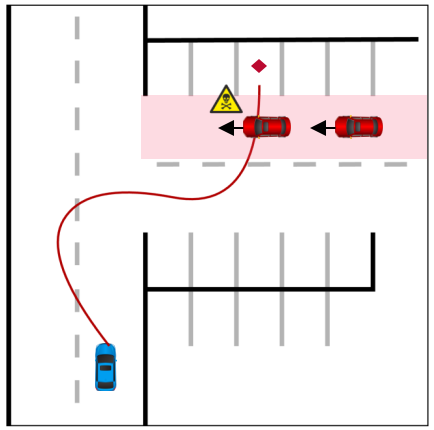
Conclusion





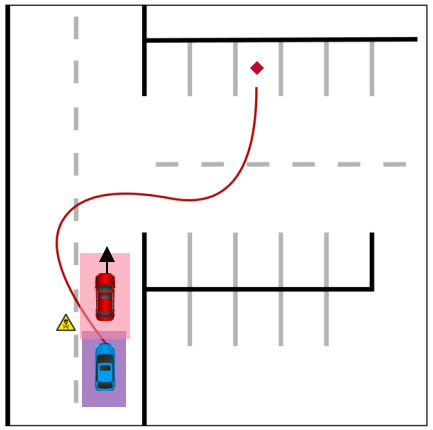
~~(1) May not follow traffic rules~~

Model learn the traffic rules by adding them as logical constraints



~~(2) May collide with oncoming traffic~~

We propose a method to define dynamic obstacle via STL logics and compute robustness



~~(3) May violate road safety constraints~~

Our method can handle multiple constraints at once