Collision-Averse and Dynamics-Aware B-Spline Derived Motion Planning for Autonomous Vehicles

by

Sruti Ganti

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
of Honors Bachelor of Science in Software Engineering

Approved April 2023 by the
Undergraduate Supervisory Committee:

Dr. Wenlong Zhang, Chair
Ruben Acuña

ARIZONA STATE UNIVERSITY

May 2023

To my father, who endowed me with a great passion for learning.

ACKNOWLEDGEMENTS

I would like to express my gratitude to **Dr. Wenlong Zhang** for allowing me this enriching learning opportunity. I would like to thank my committee, **Dr. Wenlong Zhang** and **Ruben Acuña**, for the valuable insight, guidance, and expertise provided throughout my thesis. An immense thank you to my PhD mentor, **Sunny Amatya**, for her indispensible mentorship, encouragement and patience. I would like to extend my thanks to the members of the **Robotics and Intelligent Systems (RISE) Lab** at Arizona State University - Polytechnic. It has been an absolute privilege and pleasure learning from such extraordinary PhD students.

ABSTRACT

The seamless integration of autonomous vehicles (AVs) into highly interactive and dynamic driving environments requires AVs to safely and effectively communicate with human drivers. Furthermore, the design of motion planning strategies that satisfy safety constraints inherit the challenges involved in implementing a **safety-critical** and **dynamics-aware** motion planning algorithm that produces **feasible** motion trajectories. Driven by the complexities of arriving at such a motion planner, this thesis leverages a motion planning toolkit that utilizes spline parameterization to compute the optimal motion trajectory within a dynamic environment.

Our approach is comprised of techniques originating from optimal control, vehicle dynamics, and spline interpolation. To ensure dynamic feasibility of the computed trajectories, we formulate the optimal control problem in relation to the intrinsic state constraints derived from the bicycle state space model. In addition, we apply input constraints to bound the rate of change of the steering angle and acceleration provided to the system. To produce collision-averse trajectories, we enforce extrinsic state constraints extracted from the static and dynamic obstacles in the circumambient environment. We proceed to exploit the mathematical properties of B-splines, such as the Convex Hull Property, and the piecewise composition of polynomial functions.

Second, we focus on constructing a highly interactive environment in which the configured optimal control problem is deployed. Vehicle interactions are categorized into two distinct cases: Case 1 is representative of a single-agent interaction, whereas Case 2 is representative of a multi-agent interaction. The computed motion trajectories per each case are displayed in simulation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Recent advances in robotics and ubiquitous access to computational resources distinctively position autonomous vehicles as a promising solution to critical problems faced within the sector of urban driving. In 2020, approximately 6,516 pedestrian fatalities were the result of inebriated vehicular operation and human error related to distracted driving, traffic violations, and speeding Tra (2020). Autonomous vehicles have the potential to reduce the likelihood of such fatalities by eliminating human error as a contributing factor to vehicle-pedestrian collisions. In addition, autonomous vehicles simplify vehicular operation for the elderly and individuals with mobility impairments by eradicating the overhead tasks associated with driving.

Despite significant progress, existing motion planning algorithms for current autonomous vehicles lack the human-like intelligence necessary for deployment in high-density interaction settings. For example - recent Cruise crashes highlight current motion planning algorithms deployed for autonomous vehicles failure to uphold safety and performance guarantees when exposed to unstructured and unknown environment models Korosec (2022).

How can one translate this human-like intelligence to motion planning algorithms while considering safety guarantees and kinematic constraints to produce collision-averse and dynamics-aware motion trajectories? In this thesis, we deploy an established motion planning framework, OMG-Tools Meco-Group (2016), to simulate a high-density interaction setting representative of a real driving scenario to address the aforementioned problem statement. This thesis focuses on the **collision-averse** and **dynamics-aware** aspects of motion planning for autonomous vehicles, while providing a basis for future work that addresses social intelligence. In order to provide a comprehensive overview, the following sections will illuminate the limitations of current motion planning algorithms with respect to the human-like intelligence previously discussed, as well as **collision aversion** and **dynamics-awareness**.

1.2 Legal and Social Norms

Social autonomous vehicles necessitate a foundation for intelligently communicating with human drivers. Existing motion planning algorithms substantially rely on legal norms, categorizes as traffic rules, to construct intent inference models for learning human driver behavior (Figure 1.1). However, previous research has ascertained that relying solely on legal norms results in an incomplete intent inference model, hence, increasing collision risk between autonomous vehicles and human drivers Wang *et al.* (2022).

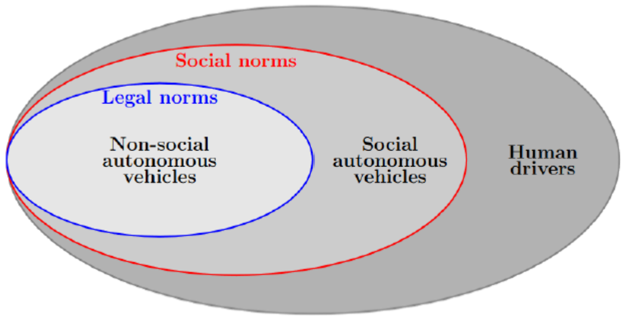


Figure 1.1: Legal and social norm variances between human drivers, social autonomous vehicles, and non-social autonomous vehicles Wang *et al.* (2022).

Contrastingly, there exist a moderate number of motion planning algorithms that employ both legal and social norms when developing intent inference models for motion planning. However, these algorithms tend to be computationally expensive, indicating the motion planning algorithm execution time is excessive and directly relates to delayed decision making. Further, these motion planning algorithms tend to deviate from expected behavior when exposed to unstructured environments. In addition, the integration of social navigation algorithms requires substantial safety and verification testing within a real human driver dataset (Figure 1.2).

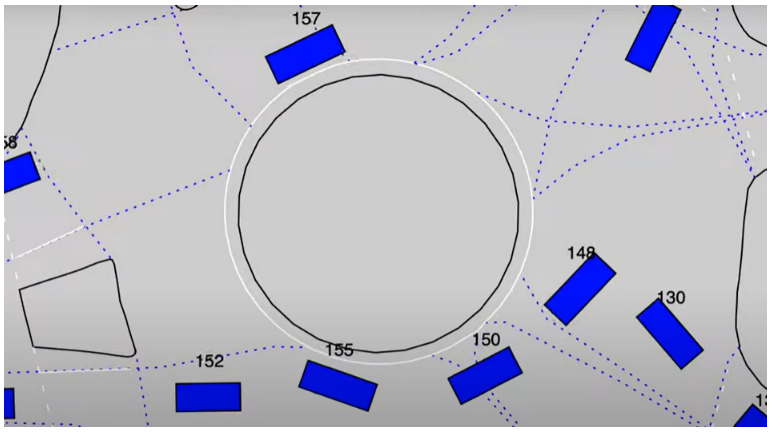


Figure 1.2: Modelling of highly interactive driving scenarios within a real human driver dataset (INTERACTION Dataset) Zhan *et al.* (2019).

1.3 Approaches to Motion Planning

Social autonomous vehicle research is categorized into two frameworks: data driven and model-based (Figure 1.3). Data-driven approaches require an extensive amount of human driver data for algorithm training, testing, and validation. Moreover, the data extracted from various driving scenarios is often inconsistent due to changes in the extraction environments and differences in road structure. In addition, the data-driven approach often produces incomprehensible results for in-depth motion planning analysis due to the black-box like nature of the framework. However, there is a field of research that concentrates on mitigating the inconsistent results produced by this framework through ensuring the motion planning algorithm is online and receiving real-time data.

The model-based approach is categorized into two components: decision-making and motion planning. In contrast to the data-driven approach, the model-based framework can be employed to a widespread set of unknown environments and produce explainable results Schwarting *et al.* (2019). However, the model-based approach consists of extensive algorithmic computation; this framework requires constant motion prediction throughout the decision-making process.

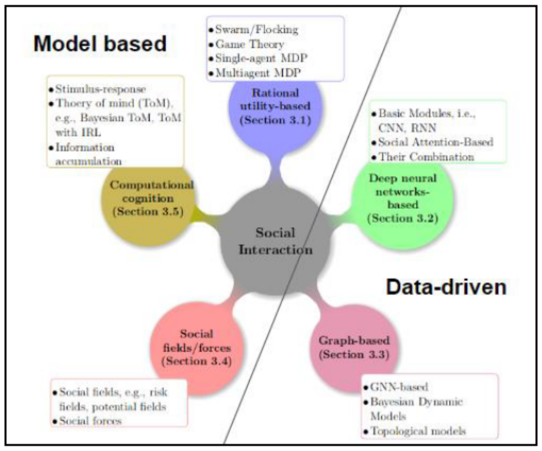

Model based
•Swarm/Flocking
•Game Theory
•Single-agent MDP
•Multiagent MDP
Rational utility-based (Section 3.1)
•Stimulus-response
•Thoery of mind (ToM), e.g., Bayesian ToM, ToM with IRL
•Information accumulation
Computational cognition (Section 3.5)
•Basic Modules, i.e., CNN, RNN
•Social Attention-Based
•Their Combination
Deep neural networks-based (Section 3.2)
Social Interaction
Data-driven
Social fields/forces (Section 3.4)
Graph-based (Section 3.3)
•Social fields, e.g., risk fields, potential fields
•Social forces
•GNN-based
•Bayesian Dynamic Models
•Topological models

Figure 1.3: Summary of two main approaches to interaction modeling and learning Wang *et al.* (2022).

1.4 Related Work

This chapter surveys previous foundational work within dynamics aware and spline-based motion planning for autonomous vehicles.

1.4.1 Dynamics-Aware Motion Planning

Alternative motion planning toolkits, such as CommonRoad Althoff *et al.* (2017), do not incorporate NLP support to compute the optimal motion trajectory. Substitute approaches to motion planning, such as Frenet frames and motion primitives were evaluated prior to selecting OMG-Tools. Frenet frames reduce motion trajectory dimensionality from the x-y frame to the s-d frame (Figure 5.2). Although, a limitation of this optimization approach is that the input acceleration relies upon the Frenet frame and cannot be transferable to the x-y frame Zhan *et al.* (2019). Further, employment of Frenet frames results in non-transferable motion trajectories, due to point-mass simplification, and frame-dependent results. An alternate optimization technique that was investigated is motion primitivesLöw *et al.* (2020). Motion primitives reduce the sample space of the vehicle's trajectory. However, the feasibility of the motion primitive is heavily dependent on uncertainties within the surrounding environment.

The OMG-Tools motion planner Meco-Group (2016) counterbalances the dynamics related limitations corresponding to the aforementioned alternatives. Moreover, the OMG-Tools motion planner integrates kinematic constraints to support the vehicle dynamics for a moderate number of vehicle models. For instance, the holonomic, quadrotor, dubins, and bicycle model are supported by the toolkit.

1.4.2 Collision Avoidance

Employing and validating safety guarantees, such as collision aversion, is standard practice for motion planning algorithms. Social behavioral models such as social force theory and social force field analysis have been proposed as alternate approaches. Moreover, these approaches use a physics-derived approach to convey social interaction in terms of force, mass, and acceleration (Figure 1.4). However, due to physical limitations, the approaches Wang *et al.* (2022) produce rigid motion that does not mimic human driving.

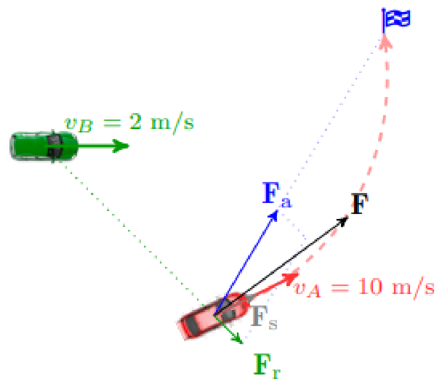


Figure 1.4: Two agent interaction deploying the social force model Wang *et al.* (2022).

OMG-Tools proves advantageous compared to the previous mentioned approaches due the smooth and feasible motion trajectories generated by the toolkit. Further, OMG-Tools exploits the Convex Hull Property of splines Mercy *et al.* (2017) to guarantee smooth, human driver-like motion.

1.4.3 Multi-Agent Systems

Empathetic Intent Inference

The process of embedding human-like intelligence within motion planning requires the autonomous vehicle to implicitly negotiate with external vehicles. Further, intelligent communication necessitates the autonomous constantly surveys the encircling environment for obstacle avoidance and implicit negotiation. Moreover, knowing *when* to implicitly negotiate with external vehicles requires substantial decision-making skill Amatya *et al.* (2022).

Splines

Traditionally, splines have been employed to solve motion planning problems by utilizing knot intervals that are assigned specific weights in order to determine the motion trajectory of a vehicle. In addition, splines simplify the kinematics of the vehicle by modelling the vehicle as a point mass. Consequently, the dynamics of the vehicle are not considered for computing motion trajectories. This results in generating vehicle trajectories that produce accurate results within simulation, but prove infeasible for the vehicle to follow in real-world application. However, spline constraint optimization has produced accurate results within a dynamic environment, and are thus being utilized for our specific motion planning problem Mercy *et al.* (2017).

In addition to supporting a variety of vehicle models, OMG-Tools also supports a suite of motion planning problems. One specific problem that was leveraged for this work was the RendezvousProblem. Further, the Rendezvous problem ensured kinematic constraints were upheld while maintaining safety guarantees through avoiding obstacles within the environment.

Conclusively, the motion planner developed by OMG-Tools generates smooth and feasible vehicle trajectories through enforcing kinematic constraints and utilizing splines. In addition, the toolkit upholds safety guarantees through enforcing state constraints derived from static and dynamic obstacles within the environment. Moreover, limited inter-vehicle avoidance is supported to provide as a base for future work related to embedding social intelligence. Further, the bicycle state space model is supported and integrated within the motion planning algorithm provided by the toolkit. The conglomeration of the aforementioned features supported by the toolkit positioned the OMG-Tools motion planner as a suitable choice for our single-agent and multi-agent interaction scenarios.

Motion planning plays a critical role in governing the computed motion trajectories that intelligently communicate with other vehicles while maintaining kinematic constraints and safety guarantees. Hence, employing a **collision-averse** and **dynamics-aware** motion planner is essential for the integration of autonomous vehicles in highly interactive driving scenarios.

1.5 Thesis Structure

1.5.1 Chapter 1: Introduction

We provide a background for the work presented within this thesis. Limitations of current motion planning algorithms are discussed with respect to collision aversion and simplification or disregard for vehicle dynamics. In addition, a detailed literature review was included to relay a step-by-step justification for the motion planning toolkit utilized throughout this work. In essence, we lay the foundation for the work that will be discussed, as well as future directions for this thesis.

1.5.2 Chapter 2: Methodology

We mathematically formulate the optimal control problem with respect to the intrinsic and extrinsic state constraints derived from the bicycle state space model and encompassing environment. In addition, we enforce input constraints by bounding the rate of change of the steering angle and acceleration given to the system. We guarantee the generated trajectories are smooth by enforcing the aforementioned input constraints and also ensuring the gradients of the velocity and steering angle from the inputs given to the system are negated. In addition, we discuss the software architecture employed for OMG-Tools.

1.5.3 Chapter 3: Simulation Results

We describe the environmental setup for Case 1 and Case 2 in detail and provide snapshots of the simulation results generated per each case. In addition, we display results for a simplified vehicle dynamics model, quadrotor, as well as a more complex vehicle dynamics model, the bicycle model. The bicycle model was chosen for its similarities to the vehicle dynamics considered when operating a vehicle. Results are displayed for both the quadrotor and bicycle model to visually demonstrate the difference in produced trajectories for point mass and dynamic models.

1.5.4 Chapter 4: Discussion

We provide an in-depth software analysis regarding the motion planning problems proposed for Case 1 and Case 2. In specific, we investigate potential factors relating to the infeasibility of the specified motion planning problem deployed for Case 2 and propose possible solutions to mitigate the factors contributing to the infeasibility. Moreover, we discuss how the software architecture employed for the leveraged toolkit contributes to limitations

1.5.5 Chapter 5: Conclusion

We summarize the contributions of this thesis and outline future directions for the work presented.

Chapter 2

METHODOLOGY

The following sections provide a foundation for the mathematical formulation of the established motion planning problem. Further, the Optimal Control Problem (OCP) that is the core of the motion planning problem, and its translation to the Non-linear Programming Problem (NLP) will be discussed thoroughly.

2.1 Mathematical Background

2.1.1 Point Mass Model

The point mass model is a simplification of dynamics. Moreover, the dimensions of the vehicle are deemed negligible, thus kinematic constraints are not enforced. This ultimately leads to “free” movement that is not constrained by the dynamics of the vehicle. Figure 2.1 depicts the centripetal nature of motion for a 2D quadrotor model. The tangential velocity (u_1) is applied directly to the center of the quadrotor. The direction the rotational velocity (u_2) is applied towards implies the quadrotor translates about its respective y-axis. However, this simplified model is not ideal for mimicking real driving motion. For example - human drivers naturally account for vehicle dimensions, such as the length of the vehicle and the turning angle, when performing a three-point turn. Due to the point mass model disregarding vehicle dimensions, the motion trajectories produced from employing this approach tend to be infeasible in high-density driving environments.

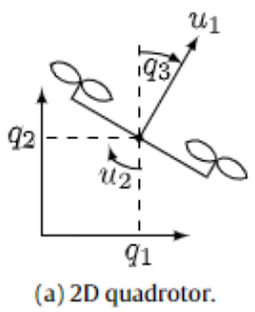


Figure 2.1: 2D quadrotor point-dynamics Van Parys and Pipeleers (2017).

2.1.2 Bicycle State-Space Model

The bicycle vehicle model was chosen due to its similarity, with respect to vehicle dynamics and kinematics, to an automobile. Unlike the aforementioned point mass model, the bicycle state space model considers vehicles dynamics. Moreover, previous work has ascertained that this vehicle model is suitable for mimicking the driving behavior of a more complex vehicle model Polack *et al.* (2017). The kinematic bicycle model is displayed within Figure 2.2, and the respective state space model is formulated below.

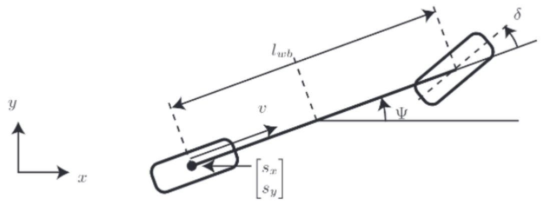


Figure 2.2: Kinematic bicycle model Althoff *et al.* (2017)

$$\dot{X}_i(t) = f_i(t, X_i, u_i) \;\; (0 \leq t \leq T)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\delta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\phi) \\ v\sin(\phi) \\ \frac{v}{L}\tan(\delta) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_{steer} \\ u_{accel} \end{bmatrix}$$

Bicycle State-Space Model Equation: For each vehicle with the ***state*** defined as: speed, distance, and direction. Where $\mathbf{X} = (x, y, \phi, \delta, v)$ and L is defined as the length of the vehicle. The control input of each vehicle is determined by the acceleration (u_{accel}) and the steering angle (u_{steer}) provided to the system. For the simulation results presented in the following section, L is set to 2.

A component of the aforementioned objective is producing ***feasible*** motion trajectories. Moreover, we are interested in computing trajectories that a vehicle can follow in a real-world driving environment. Hence, it is essential we account for the kinematic constraints innate to the chosen vehicle model. We translate the kinematic constraints of the model into intrinsic state constraints that are then enforced on our OCP, thus producing ***dynamics-aware*** trajectories. Further discussion on the intrinsic state constraints applied to the minimization problem will be outlined in the following sections.

2.1.3 Properties of B-Splines

This work exploits specific mathematical properties of B-Splines. In specific, the Convex Hull Property and the piecewise composition of splines. Further, splines can be expressed as a linear combination of B-spline basis functions.

$$s(\tau) = \sum_{i=1}^{n} c_i \cdot B_i(\tau)$$

Piecewise functions represented as linear combination of B-spline basis functions Mercy *et al.* (2017). Where B_i is representative of the B-spline basis functions, n is defined by the degree of the spline, the number of piecewise components the spline consists of, and the nature of the spline continuity. The remaining properties are established by the number of spline knots. Figure 2.3 visually depicts these selective properties of splines.

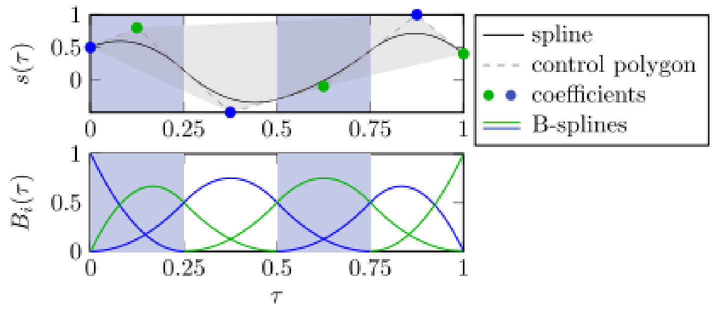


Figure 2.3: Graphical illustration of the properties of a spline Mercy *et al.* (2017)

Further, the primary reason behind utilizing splines is the Convex Hull Property; the B-spline basis functions are positive and sum to one Mercy *et al.* (2017). Thus, a spline is defined within the convex hull of its control polygon Cohen *et al.* (1978). This characteristic will become crucial when discussing the transformation of the infinite-horizon OCP to a finite-horizon NLP.

2.2 Problem Formulation

Eliminating the consideration of any constraints or optimizations, the motion planning problem is quite trivial in nature. The object can be qualitatively stated as follows: Begin at the starting point and accelerate as efficiently as possible towards the destination. However, we are interested in acquiring collision-averse and dynamics-aware motion trajectories. Thus, our motion planning problem can be reconstructed as an optimization problem. Further, we are interested in determining the control and state trajectories for a dynamic system while satisfying the optimization criteria: collision-averse and dynamics-aware motion.

In order to compute dynamics-aware trajectories, we are concerned with two main components: **feasibility** and **smoothness**. In order to generate safety-critical trajectories, we are concerned with one main component: **collision aversion**. The importance of these components, with respective to computing optimal motion trajectories, will be discussed within the succeeding sections.

2.2.1 Intrinsic Constraints

It has been previously established that we must consider the kinematic limitations of the vehicle model in order to produce feasible motion trajectories. Hence, the primary intrinsic constraints applied to our OCP are defined as: constraints derived from the bicycle state space model equation. In order to guarantee smoothness, we must restrict the vehicle's acceleration (u_{accel}) and rate of change of steering angle (u_{steer}). Equation 2.1 constrains the acceleration inputted to the system relative to the associated minimum and maximum acceleration values. Equation 2.2 constrains the rate of change of steering provided to the system relative to the minimum and maximum rate of change of steering values. In addition, the spline gradients extracted from the input given to the system must cancel out in order to guarantee trajectory smoothness.

$$u_{min,i_{accel}} \leq u_{i_{accel}}(t) \leq u_{max,i_{accel}} \tag{2.1}$$

$$\dot{u}_{min,i_{steer}} \leq \dot{u}_{i_{steer}}(t) \leq \dot{u}_{max,i_{steer}} \tag{2.2}$$

2.2.2 Extrinsic Constraints

In a dynamic environment that consists of static and dynamic obstacles, collision aversion constraints are critical. To address collision aversion, we enforce extrinsic constraints derived from the static and dynamic obstacles within the surrounding environment. To clarify, external vehicles added to the system are modelled as dynamic obstacles. Thus, the extrinsic constraints are changing and updating over the time horizon. These safety-critical constraints are expressed in Equations 2.3, 2.4, 2.5.

$$s(t)^T x_i(t) - b(t) \geq r_{veh,i}, \tag{2.3}$$

$$s(t)_y^T c(t) - b(t) \leq 0, \forall_c \in 1, ..., n_c, \tag{2.4}$$

$$\|s(t)\|_2 \leq 1 \tag{2.5}$$

2.2.3 Infinite-Dimension Optimal Control Problem

The solution space of the desired motion trajectories is within the infinite space. Further, the state and input constraints must persist for the entire time horizon. In addition, the state and input constraints are not classified as convex or linear. Thus, this OCP is complex in nature and proves difficult to solve. Moreover, the driving environment is unstructured and consists of a multitude of unknowns. The infinite-dimension OCP must be translated to a finite-dimension NLP in order to ensure the tractability of the motion planning problem. The following section explains the translation from the infinite-dimension to the finite-dimension through utilizing splines.

2.2.4 Translation to Finite-Dimension Optimal Control Problem

In order to translate the OCP and ensure the motion problem is tractable, a spline parameterization is first applied to the trajectories. Second, the OCP is re-formulated in terms of the updated constraints and trajectories. Further, the trajectories are approximated as splines, and thus, correlate to piecewise functions. The formulation for the piecewise functions are presented within Equation 2.6.

$$x_i(t) = \sum_{y=1}^{n} x_{i,y} b_k(t), \tag{2.6}$$

Where the B-spline bases are defined as and the B-spline coefficients as $b = (b_1, ..., b_n)$ and B-spline coefficients as $x_i = (xi, 1, ..., xi, n)$. Furthermore, the Convex Hull Property guarantees that a B-spline function is constricted by the minimum and maximum values of its coefficients. Thus, simply constraining the coefficients translates to imposing semi-infinite constraints on the spline functions. Through utilizing spline parameterization, the OCP is concisely formulated as:
x_i for each vehicle i:

$$\min_{\forall i : \boldsymbol{x}_i(.)} \sum_{i=1}^{N} J_i(x_i) \tag{2.7}$$

such that

$$\boldsymbol{x}_i(t) \in X_i \tag{2.8}$$

$$\forall i \in 1, ..., N \tag{2.9}$$

2.2.5 OMG-Tools Software Architecture

As established within Chapter 1, OMG-Tools was selected as the basis motion planner. OMG-Tools is an open-source motion planning toolkit that models and simulates motion planning problems Meco-Group (2016). The OMG-Tools motion planner provides a suite of vehicle models and motion problem planning types to select from. For the purposes of this work, the bicycle model as well as the quadrotor model was chosen along with the Point2point problem for single-agent interaction and Rendezvous problem for multi-agent interaction. CasADi, is an open-source tool for non-linear optimization. IPOPT, a non-linear optimization technique within the CasADi library is the NLP solver that is applied for the OMG-Tools motion planner Andersson *et al.* (2019).

The state diagram shown in Figure 2.4 depicts the iterative computation process the NLP solver utilizes to converge to a solution. To initialization the motion planning problem, a vehicle is first created and the desired environment is constructed. Sequentially, the type of motion planning problem is selected and a simulator is instantiated. The iterative process takes place between running the simulator and generating results. The simulator runs per each respective piecewise function following the configuration of the NLP with respective to splines. This process repeats until we have iterated through the entire time horizon. Upon reaching the end of the time horizon, the individual results generated for each piecewise are conglomerated to produce the motion trajectory.

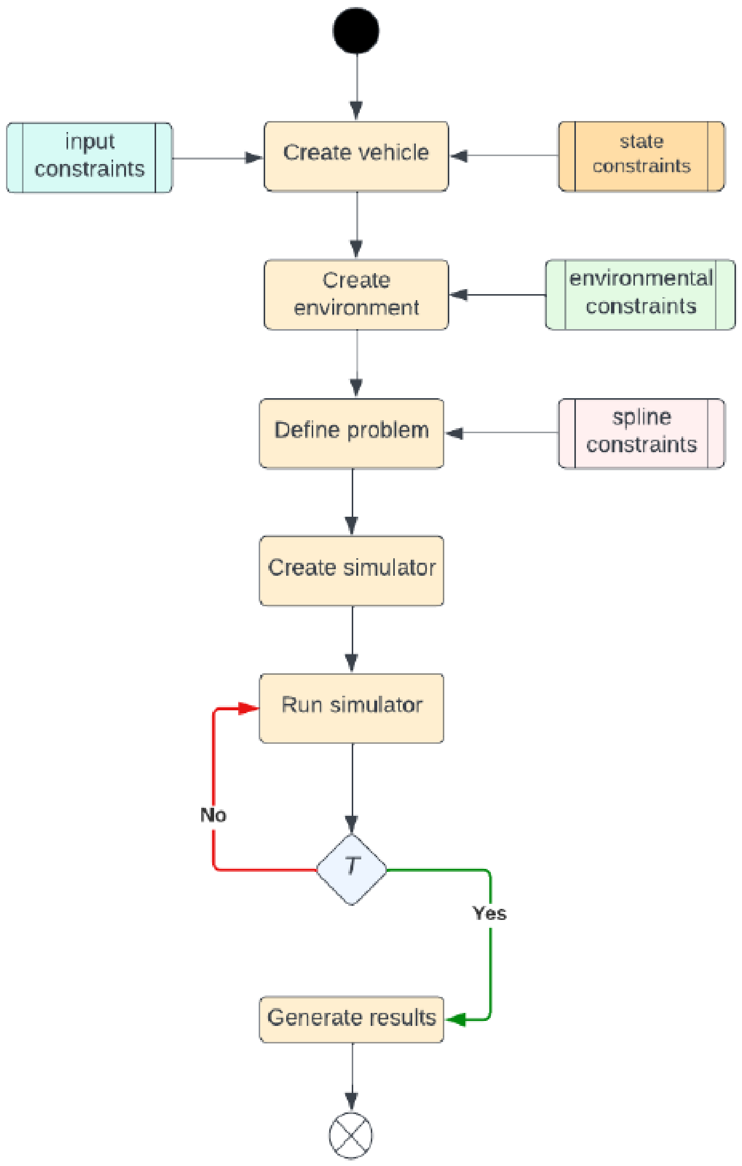


Figure 2.4: OMG-Tools Software Architecture; high-level state diagram.

Chapter 3

SIMULATION RESULTS

For a comprehensive overview of the generated simulations, please view the ‘Simulation_Results’ directory within the respective GitHub repository:
https://github.com/sganti2/Social-Navigation-Autonomous-Vehicles

3.1 Simulation Environment

The objective of this simulation setup was to construct an environment that closely replicates a highly interactive driving setting representative of the real-world. Thus, the roundabout scenario was chosen due to its composition consisting of a high interaction density as well as static and dynamic obstacles. Figure 3.1 concisely displays the nature of interactions seen within this chosen roundabout scenario.

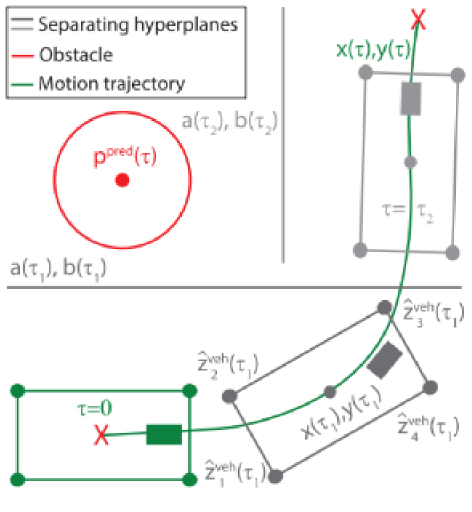


Figure 3.1: Selected roundabout scenario indicating static obstacle and vehicle motion trajectories. Mercy *et al.* (2017)

To account for the complexity variances between single-agent and multi-agent interactions, to distinct cases have been initiated. Case 1 is representative of a single-agent interaction that consists of a static roundabout. The primary agent's objective within this case is to begin at the established starting position, and accelerate towards the established destination while considering vehicle dynamics and avoiding collision with the static roundabout. Case 2 is representative of a multi-agent interaction and more closely mirrors interactions seen in real driving scenarios. Further, Case 2 extends the setup for Case 1 by incorporating more complexity due to static and dynamic obstacles. A secondary agent is introduced to the environment that is also initiated with a starting point and destination. The primary agent's objective for Case 2 is to begin at the established starting position, and accelerate towards the established destination while considering vehicle dynamics, avoiding collision with the static roundabout, as well as avoiding ***avoiding collision with the secondary vehicle***, which is classified as a dynamic obstacle. Further, the motion planning problem introduced to Case 1 is considered a Point2Point problem. However, due to introducing a secondary vehicle to the environment, the motion planning problem is translated to a Rendezvous problem in Case 2. The upheld constraint for the Rendezvous problem is that all vehicles within the fleet must converge at the same destination.

3.2 Case 1: Single-Agent Interaction

3.2.1 Algorithmic Overview

Case 1 is representative of a trivial single-agent interaction. The algorithmic execution is depicted in Figure 3.2 and defined as follows:

- First, we consider the intrinsic and extrinsic constraints applied to the optimal control problem.
- Next, we iterate based on the following condition:
  while the current time is not equivalent to the time horizon
  - Create vehicle based on the selected vehicle model.
  - Define the knot intervals for the spline computation.
  - Construct roundabout environment.
  - Define velocity inputs for the vehicle's trajectory. *If obstacles are present within the constructed environment*
    - Add vehicle to the environment.
    - Define the problem type.
    - Initiate the problem.
    - Set the vehicle problem attribute equivalent to the pre-defined problem.
    - Set the simulator problem attribute equivalent to the pre-defined problem.
    - Run the simulator.

```
Algorithm 1 Single-Agent Interaction
Require: $x_i(t) \in X_i$,
  $\forall t \in [0, T]$,
  $\forall i \in 1, ..., N$.
  while $t \neq T$ do
    $veh \leftarrow$ create vehicle
    $knot\_intervals \leftarrow$ define knot intervals
    $env \leftarrow$ create environment
    $traj \leftarrow \{\text{vel} : \{0.5\}, \{0.3, 0.0\}\}$
    if obs != null then
      $env \leftarrow$ environment.add_obs()
      $prob \leftarrow$ Point2point(veh, env)
      prob.init()
      veh.prob = prob
      $sim \leftarrow$ Simulator(prob)
      sim.run()
    end if
  end while
```

Figure 3.2: Case 1: Single-Agent Interaction; step-wise algorithmic execution.

3.2.2 Quadrotor Results

Figure 3.3 illustrates the trajectory computed for the quadrotor model. The quadrotor is modelled as a point-mass; thus, the dynamics are not considered and the motion depicted is an idealization - the dimensions of the quadrotor are disregarded. As a result, we see the quadrotor is less constrained in terms of movement, and follows a smooth, parabolic-like path.

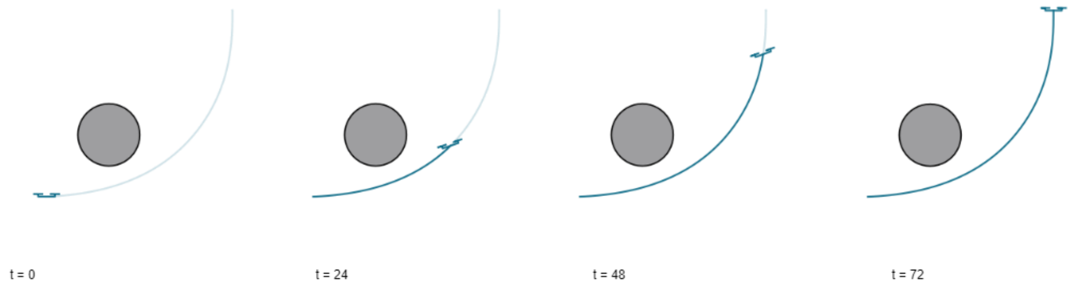


Figure 3.3: Case 1: Single-Agent Interaction; quadrotor model simulation results.

3.2.3 Bicycle Results

Figure 3.4 depicts the computed trajectory for the bicycle model. Unlike the quadrotor, the bicycle is modelled as a dynamic system. Thus, the motion depicted is a bit more complex as the vehicle's length and kinematic restrictions are taken into account. Moreover, we are able to visualize the bicycle following a smooth, spline-like trajectory. Due to intrinsic state constraints that are applied, we see the bicycle vehicle is not able to move as "freely" as the aforementioned results generated for the quadrotor model.

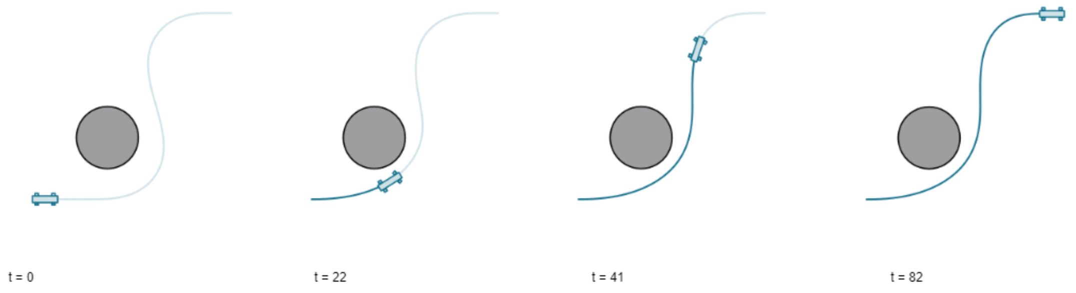


Figure 3.4: Case 1: Single-Agent Interaction; bicycle model simulation results.

3.3 Case 2: Multi-Agent Interaction

3.3.1 Algorithmic Overview

Case 2 is representative of a more complex multi-agent interaction. The algorithmic execution is quite similar in nature to Case 1, with the main differentiating factor stemming from the introduction of a second vehicle. The stepwise algorithmic execution is depicted in Figure 3.5 and defined as follows:

- First, we consider the intrinsic and extrinsic constraints applied to the optimal control problem.

- Next, we iterate based on the following condition:

  while the current time is not equivalent to the time horizon

  - Create fleet based on the selected vehicle model and number of vehicles.
  - Define the knot intervals for the spline computation.
  - Construct roundabout environment.
  - Define velocity inputs for the fleet's trajectory.

    If obstacles are present within the constructed environment

    * Add fleet to the environment.
    * Define the problem type.
    * Initiate the problem.
    * Set the fleet problem attribute equivalent to the pre-defined problem.
    * Set the simulator problem attribute equivalent to the pre-defined problem.
    * Run the simulator.

```
Algorithm 2 Multi-Agent Interaction
Require: $x_i(t) \in X_i$,
  $\forall t \in [0, T]$,
  $\forall i \in 1, ..., N$.
  while $t \neq T$ do
      $fleet$ ← create fleet
      $knot\_intervals$ ← define knot intervals
      $env$ ← create environment
      $traj$ ← {vel : {0.5}, {0.3, 0.0}}
      if obs != null then
          $env$ ← environment.add_obs()
          $prob$ ← Rendezvous(fleet, env)
          prob.init()
          fleet.prob = prob
          $sim$ ← Simulator(prob)
          sim.run()
      end if
  end while
```

Figure 3.5: Case 2: Multi-Agent Interaction; step-wise algorithmic execution.

3.3.2 Quadrotor Results

Figure 3.6 depicts the computed trajectories for both quadrotors within the interaction case. Case 2 utilizes the Rendezvous problem to compute the respective trajectories, thus the destination point for both vehicles is equivalent. As previously mentioned, we see both quadrotors following a smooth, somewhat parabolic-like motion trajectory.

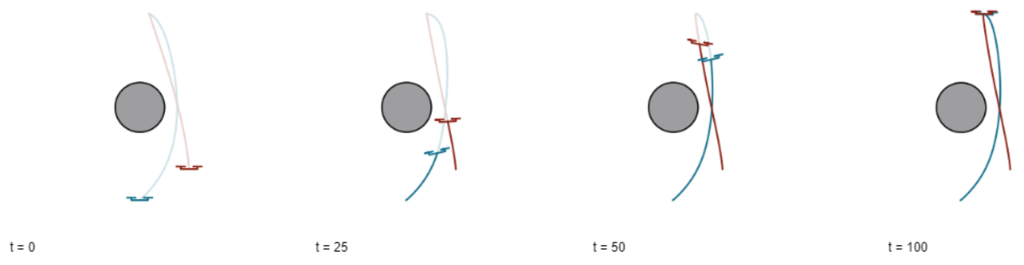


Figure 3.6: Case 2: Multi-Agent Interaction; quadrotor model simulation results.

3.3.3 Bicycle Results

The simulation results for the two bicycle vehicles are shown within Figure 3.7. The vehicles attempt to reach the same destination point, However, the bicycle vehicles are unable to converge and an infeasible motion planning problem is detected. Potential contributing factors to this infeasibility will be addressed and investigated within the following section.

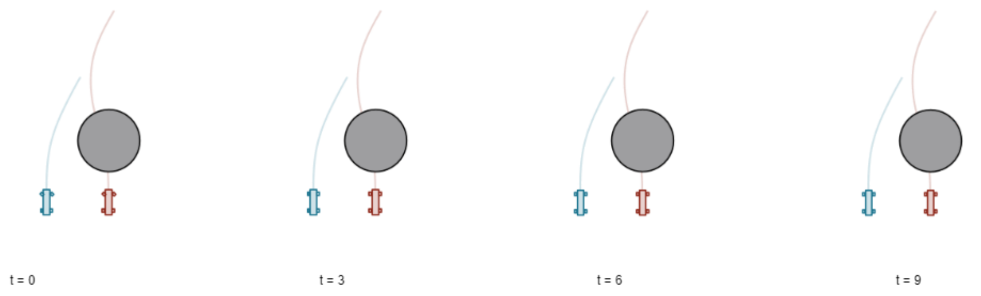


Figure 3.7: Case 2: Multi-Agent Interaction; bicycle model simulation results.

Chapter 4

DISCUSSION

4.1 Discussion Foundation

In order to provide a basis for the ensuing discussion, or analysis of results, we introduce Eq. 4.1 to address the number of sub computations per the motion planning problem and selected vehicle model. To elaborate, Eq. 4 states the number of sub-computations is equivalent to the number of knot intervals multiplied by the number of states derived from the vehicle dynamics model, multiplied by the number of vehicles present within the environment.

$$num\_comps = (num\_knots * num\_states) * num\_vehs \tag{4.1}$$

Per Eq. 4.1, we proceed to determine the number of sub-computations per each distinct interaction case, accounting for the vehicle model employed for each case. Table 4.1 below summarizes the number of sub-computations the NLP attempts to converge per each case. However, the total number of sub-computations is not all-inclusive; further, these values within the table are approximations. There are additional sub-computations attributed to spline and gradient computations.

Case	Model	Sub-computations
Case 1: Single-agent interaction	Quadrotor	num comps = (num knots * num states)* num vehs = (5*3)*1=**15** sub_computations
Case 1: Single-agent interaction	Bicycle	num comps = (num knots * num states)* num vehs = (5*5)*1=**25** sub_computations
Case 2: multi-agent interaction	Quadrotor	num comps = (num knots * num states)* num vehs = (5*3)*2=**30** sub_computations
Case 2: multi-agent interaction	Bicycle	num comps = (num knots * num states)* num vehs = (5*5)*2=**50** sub_computations

Table 4.1: Sub-computations per each interaction case and vehicle model.

In addition, we introduce the 2D quadrotor point-dynamics for the specific type of Quadrotor modelled within OMG-Tools in Figure 2.1. From this model, the corresponding state equation is displayed on the right-handside.

Quadrotor 2D state dynamics equation (derived from Figure 2.1):

$$\dot{X}_i(t) = f_i(t, X_i, u_i) \;\; (0 \leq t \leq T)$$
$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} a\cos(\theta) \\ a\sin(\theta) \\ u_2 \end{bmatrix} \begin{bmatrix} u_{vel_x} \\ u_{vel_y} \end{bmatrix}$$

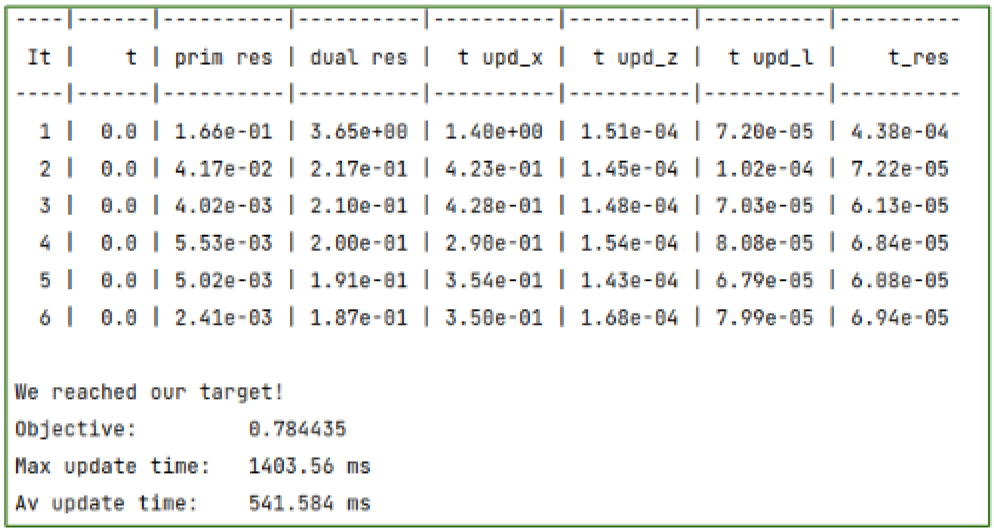
The states displayed within the left-handside of the equation determine the number of states to include when computing the number of Quadrotor sub-computations per each interaction. Specifically, rows two and four within Table 4.1

4.2 Infeasible Problem Detected

The simulation result from Case 2: Multi-Agent Interaction is unable to converge. Moreover, we receive either a *false* success case from the NLP solver, or a failure case indicating an "infeasible_problem_detected". As discussed within Chapter 2, The NLP solver utilized for OMG-Tools is IPOPT, a large-scale, non-linear optimization tool from the CasADi optimization framework. An investigation pertaining to each failure type is outlined in the following sections.

4.2.1 False Success Case

The solution is attempted, however, the resulting trajectory is infeasible. Firstly, the secondary agent is not taken into account when computing the joint trajectory. This is shown within Figure 4.1, where the IPOPT table generated includes values for the primary agent and does not account for the secondary agent per each time step. Secondly, the projected trajectory appears to indicate forward motion through the static obstacle (Figure 3.7).



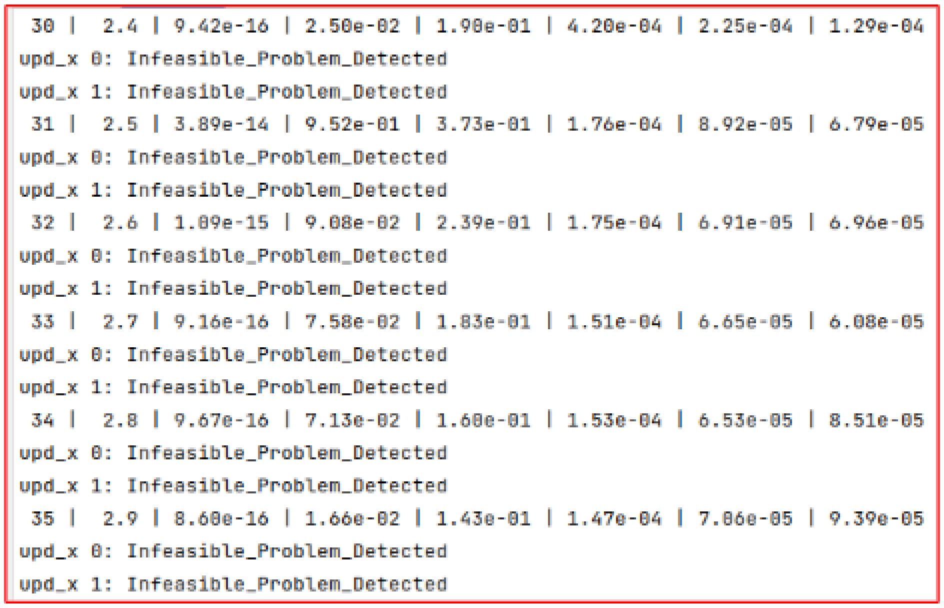
```
----|------|----------|----------|----------|----------|----------|----------
 It |    t | prim res | dual res |  t upd_x |  t upd_z |  t upd_l |    t_res
----|------|----------|----------|----------|----------|----------|----------
  1 |  0.0 | 1.66e-01 | 3.65e+00 | 1.40e+00 | 1.51e-04 | 7.20e-05 | 4.38e-04
  2 |  0.0 | 4.17e-02 | 2.17e-01 | 4.23e-01 | 1.45e-04 | 1.02e-04 | 7.22e-05
  3 |  0.0 | 4.02e-03 | 2.10e-01 | 4.28e-01 | 1.48e-04 | 7.03e-05 | 6.13e-05
  4 |  0.0 | 5.53e-03 | 2.00e-01 | 2.90e-01 | 1.54e-04 | 8.08e-05 | 6.84e-05
  5 |  0.0 | 5.02e-03 | 1.91e-01 | 3.54e-01 | 1.43e-04 | 6.79e-05 | 6.08e-05
  6 |  0.0 | 2.41e-03 | 1.87e-01 | 3.50e-01 | 1.68e-04 | 7.99e-05 | 6.94e-05


We reached our target!
Objective:          0.784435
Max update time:    1403.56 ms
Av update time:     541.584 ms
```

Figure 4.1: NLP Solver (IPOPT): False success case detected.

4.2.2 Failure Case

Contrastingly, the failure case indicates the NLP solver is unable to converge to a solution from initiation (Figure 4.2). Thus, it is most likely an environmental factor that is contributing to this case. To clarify, setting the same end destination of both vehicles is not directly contributing to this failure case. The Rendezvous problem selected to compute the optimal trajectories for the multi-agent interaction case necessitates the end destinations are equivalent. Further, setting the same end destination for each vehicle has been tested in isolation, while maintaining all environmental variables, and the NLP converged to a solution. Therefore, setting the same destination for both vehicles is not the root cause of the infeasibility. Extensive hyper-parameter and edge-case testing has been testing to identify the primary contributing factor to the infeasibility of the specified motion planning problem.



```
 30 |   2.4 | 9.42e-16 | 2.50e-02 | 1.90e-01 | 4.20e-04 | 2.25e-04 | 1.29e-04
upd_x 0: Infeasible_Problem_Detected
upd_x 1: Infeasible_Problem_Detected
 31 |   2.5 | 3.89e-14 | 9.52e-01 | 3.73e-01 | 1.76e-04 | 8.92e-05 | 6.79e-05
upd_x 0: Infeasible_Problem_Detected
upd_x 1: Infeasible_Problem_Detected
 32 |   2.6 | 1.09e-15 | 9.08e-02 | 2.39e-01 | 1.75e-04 | 6.91e-05 | 6.96e-05
upd_x 0: Infeasible_Problem_Detected
upd_x 1: Infeasible_Problem_Detected
 33 |   2.7 | 9.16e-16 | 7.58e-02 | 1.83e-01 | 1.51e-04 | 6.65e-05 | 6.08e-05
upd_x 0: Infeasible_Problem_Detected
upd_x 1: Infeasible_Problem_Detected
 34 |   2.8 | 9.67e-16 | 7.13e-02 | 1.60e-01 | 1.53e-04 | 6.53e-05 | 8.51e-05
upd_x 0: Infeasible_Problem_Detected
upd_x 1: Infeasible_Problem_Detected
 35 |   2.9 | 8.60e-16 | 1.66e-02 | 1.43e-01 | 1.47e-04 | 7.86e-05 | 9.39e-05
upd_x 0: Infeasible_Problem_Detected
upd_x 1: Infeasible_Problem_Detected
```

Figure 4.2: NLP Solver: Failure case detected.

Moreover, extensive hyper-parameter testing for both the false-success and failure cases indicates this error is most likely originating from the NLP solver's inability to support the number of sub-computations for the Case 2 motion planning problem. This conclusion is further supported by the planner excluding examples including multiple bicycle vehicles. However, additional exhaustive hyper-parameter testing is required to confirm this inability to support the number of sub-computations is the root cause of this false success case. The results of the hyper-parameter testing and factors that may be contributing to the infeasibility will be discussed in detail in the following sub-section.

4.3 Software Analysis

In order to narrow down the scope of contributing infeasibility factors, the following tests were conducted: input testing, sub-computation overload analysis, and vehicle model testing. A brief overview of each test objective accompanied with results will be described in the following sections.

4.3.1 Input Testing

The starting condition, terminal condition, vehicle orientation and environment size were identified as parameters inputted to the system. For the aforementioned, we constructed boundary test cases and selectively chose test cases that were representative of each boundary. The results of this test suite are displayed in Table 4.2 below.

Factor	Change	Result
Starting condition	Initial position $(0, 1)$	Failure
Terminal condition	Final position $(0, 4)$	Failure
Vehicle orientation	Orientation $\left(\frac{\pi}{2}\right)$	Failure
Environment size	Reduced size (2) Increased size (10)	Failure

Table 4.2: Selective display of input boundary testing values.

4.3.2 Sub-computation Overload

As previously established, the multi-agent interaction case for the bicycle model is approximated to consists of fifty sub-computations. However, this value is not representative of the spline and input gradient calculations. Further, the knot intervals for both Case 1: Single-Agent Interaction and Case 2: Multi-Agent Interaction are set to five for consistency. Thus, to investigate if the OMG-Tools NLP is simply unable to support the number of sub-computations required to converge, we tested a combination of input parameters (Table 4.2) and available vehicle models. The results of this test suite are displayed within Table 4.3 below. For brevity, every combination tested has been condensed to single test cases representative of a specific boundary.

Vehicle Model	Result
Quadrotor	Success
Dubins	Failure
Holonomic	Success
Bicycle	Failure

Table 4.3: Vehicle model testing results. These results include combinations of input parameters alongside selective vehicle models.

4.4 Possible Solutions

In order to resolve the infeasible motion planning problem that has been detected, we propose the following solutions. Moreover, a decision matrix (Table 4.4) was constructed to hierarchically categorize each proposed solution. Proposed solutions are included within the first column of the matrix. Per category, indicated within the first row of the table, each solution is ranked from zero to four. A score of zero indicates the solution is ineffective for that specific category. Contrastingly, a score of four indicates the solution is effective for the indicated category.

1. **Exhaustive hyper-parameter testing** In order to conclusively identify the primary contributing factor to the infeasibility of the problem, further hyper-parameter testing is required. Moreover, the hyper-parameter testing we have conducted is representative of a majority of input combinations. However, this test suit is not exhaustive. Thus, it is essential exhaustive hyper-parameter testing is continued. This proposed solution is ranked within the second row of the

decision matrix (Table 4.4). In addition, this approach requires a considerable amount of time, some expertise, and is inefficient due to a lack of automated testing. Although, vehicle dynamics are still considered for the OCP, the overall score indicates this solution will be least fruitful. Due to the tight integration of intrinsic and extrinsic state constraints within the OMG-Tools motion planner, systematically removing state constraints is a complex process and requires deconstructing the entire motion planning algorithm.

2. **Simplify OCP** An alternative solution is to simplify the OCP by disregarding the vehicle dynamics. There exist motion planning frameworks that utilize this approach, however, the computed trajectories are no longer dynamics-aware and are often infeasible. Further, the objective of this work is to provide a basis for embedding embedding empathetic intent inference. However, successful incorporation of empathetic intent inference requires vehicle dynamics consideration to generate accurate motion Wang *et al.* (2019). Thus, if large-scale real-world deployment is the overarching intent, this proposed solution is not viable.

3. **Implement NLP Solver** The primary hindrance emanating from utilizing the OMG-Tools motion planner appears to be the NLP solver's inability to support the number of sub-computations for our specific interaction case. Thus, the most optimal solution is developing an NLP solver that is specifically designed for the desired interaction case. A few drawbacks of this solution, as summarized within the decision matrix, indicate this approach requires expertise and is relatively time-intensive. However, the advantages of this solution outweigh the disadvantages; the motion planner maintains its dynamics-aware approach while improving trajectory generation efficiency with respect to OMG-Tools. Thus, developing an NLP solver from the ground up is the high-yielding solution.

Proposed Solution	Time	Expertise	Vehicle Dynamics	Efficiency	Score
Exhaustive hyper-parameter testing	1	2.5	3.5	1.5	8.5
Simplify OCP	4	4	0	2.5	10.5
Implement NLP Solver	2	1	4	4	11

Table 4.4: Decision matrix that hierarchically categorizes each proposed solution.

Chapter 5

CONCLUSION

This thesis deployed an established motion planning framework for computing safety-critical and dynamics-aware motion trajectories through incorporating collision aversion and enforcing constraints derived from vehicle dynamics. Consequently, the work presented within this thesis provides a basis for the future work discussed in the following subsection.

Chapter 1 provided an introduction to autonomous vehicle deployment within the urban driving domain. In addition, the limitations of existing motion planners and current approaches to motion planning were discussed in detail. A throughout literature review was provided justifying the selection of OMG-Tools as the motion planner.

Chapter 2 translated the qualitative problem statement to an optimal control problem in accordance to intrinsic and extrinsic constraints. To recapitulate, intrinsic constraints were derived from the bicycle state-space model and enforced to account for vehicle dynamics. The smoothness of the generated trajectories was expanded upon and consisted of two main considerations:

- Primarily, bounding the input constraints to reduce rough qualities within the resulting trajectories.
- In addition, the gradients of the steering angle and velocity were examined to ensure smoothness.

Furthermore, Chapter 2 provided a mathematical basis for translating the infinite-horizon optimal control problem to a receding-horizon non-linear programming problem through the introduction of B-splines and assigning weights to the established knot intervals.

Chapter 3 presented the simulation results generated for the two distinct interaction cases: Case 1 and Case 2, respectively. Moreover, the parabolic-like motion of the Quadrotor vehicles and spline-like motion of the Bicycle vehicles was highlighted. Simulation results that acted as a foundation for the analysis and discussion within the following chapters were outlined in detail.

Chapter 4 deconstructed the infeasible motion problem detected for Case 2 from a software and vehicle dynamics perspective. Contributing factors pertaining to the infeasible characteristics of the proposed motion planning were investigated thoroughly. Sub-computations per each vehicle model were computed and software limitations were probed. Solutions to mitigate the infeasibility of the motion planning problem, accompanied with a justification were proposed as well.

Conclusively, **Chapter 5** summarizes the key points presented within this thesis and establishes a foundation for extending this work.

5.1 Future Work

5.1.1 Robust NLP Solver

Per the aforementioned discussion, OMG-Tools motion planning software is characterized by substantial interdependence amongst subclass components Richards (2015). Consequently, modifying one subclass component may result in unintended modifications affecting adjacent subclass components. Ultimately, the tightly coupled interactions present within the current motion planner result in inconsistencies at the software level. Hence, future work includes developing an NLP solver from the ground up that is designed for our specific interaction environment. Further, the solver should be robust to computation errors emanating from the complexity of the motion planning problem.

5.1.2 Social Intelligence Integration

The current motion planning framework classifies external vehicles as dynamic obstacles. However, this type of classification fails to incorporate human-like intelligence, which is synonymous to social intelligence for the work presented. As established within the introduction, overlooking social intelligence results in an incomplete behavioral model that ultimately produces incoherent results when deployed in the real world. Thus, it is essential to embed social intelligence within the developed motion planner through integrating empathetic intent inferenceAmatya *et al.* (2022). The majority of work discussed within this thesis address the motion planning aspect of autonomous vehicles. Extension of the work presented, with regard to social intelligence, requires analyzing the decision-making process alongside motion planning. Figure 5.1 depicts the overarching objective of embedding empathetic intent inference within the developed motion planner.

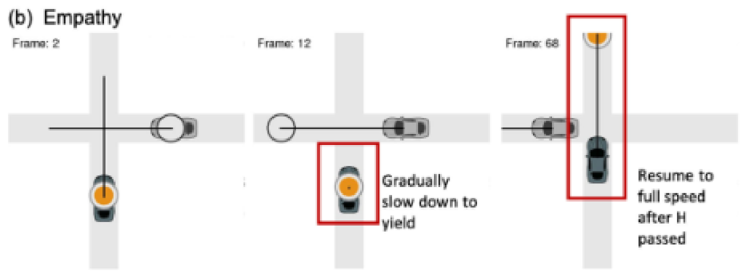

(b) Empathy
Frame: 2
Frame: 12
Frame: 68
Gradually slow down to yield
Resume to full speed after H passed

Figure 5.1: Empathetic intent inference within the realm of decision-making Wang *et al.* (2022).

5.1.3 Human Driver Dataset Validation

In order to ensure the computed motion trajectories are dynamically compliant and uphold deployment in real driving environments, the motion planner must be extensively validated within a human driver dataset. The INTERACTION Dataset Zhan *et al.* (2019) is proposed as a validation dataset for two primary reasons:

- The dataset consists of real human driver data, thus, evaluation within the dataset will provide an accurate indication of whether the computed trajectories closely mimic human driver behavior.
- A consistent theme within this work is the highly dynamic nature of urban driving scenarios. Hence, the INTERACTION Dataset was chosen due to the data composing this dataset being derived from high-density urban driving scenarios. Further, Table 5.1 summarizes the nature of the data contained within the INTERACTION Dataset against other available datasets for validation.

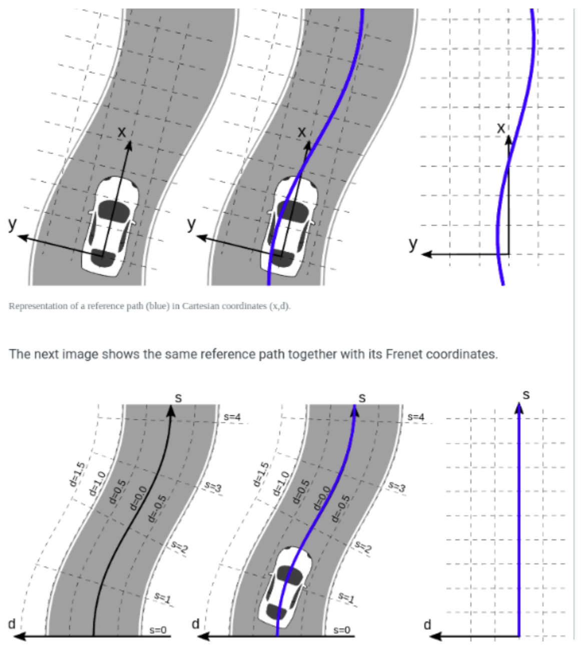
	highly interactive scenarios	complexity of scenarios	density of aggressive behavior	near-collision situations and collisions	HD maps with semantics	completeness of interaction entities & viewpoint
NGSIM [13]	ramp merging, (double) lane change	structured roadk, explicit right-of-way	low	very few near-collision	no	yec, bird's-eye-view from a building
highD [17]	lane change	structured roads, explicit right-of-way	low	very few near-collision	no	yes, bird's-eye-view from a drone
Argoverse [19]	unsignalized intersections, pedestrian crossing	unstructured roads, inexplicit right-of-way	low	no	yes, but partially	only for the ego data-collection vehicle
INTERACTION	roundabouts, ramp merging. double lane change unssgnalized intersections	unstructured roads, inexplicit right-of-way	high	yes	yes	yes, bird's-eye-view from a drone

Table 5.1: Comparison with existing motion datasets Zhan *et al.* (2019).

5.1.4 Baseline Evaluation

Frenet frames simplify vehicle dimensionality from the x-y frame to the s-d frame (Figure 5.2). However, the acceleration inputted to the system is dependent on the Frenet frame itself, and is often not feasible to translate to the x-y frame. Furthermore, Frenet frames utilize point-mass simplification to compute motion calculations. While this approach may produce computationally efficient results, the resulting motion trajectories are infeasible due to a lack of considering vehicle dynamics.

Motion primitives reduce the sample space of the vehicle's trajectory. However, depending on uncertainties present within the environment model, not all motion primitives are achieve able. Nevertheless, for validation purposes, Frenet frames and motion primitives act as foundational baselines for performance comparison.


Representation of a reference path (blue) in Cartesian coordinates (x,d).

The next image shows the same reference path together with its Frenet coordinates.

Figure 5.2: Frenet frame translation from x-y frame to s-d frame Löw *et al.* (2020).

To concisely summarize, the proposed future work includes comprehensively assessing the model-based motion planning approach to assess transferability and generalization to unstructured environment models. Further, the developed motion planning algorithm should be validated against baselines such as Frenet frames Sun *et al.* (2020) and motion primitives Löw *et al.* (2020) to assess the optimality of the computed motion trajectories.

5.2 Thesis Reflection

"The test of all knowledge is experiment. Experiment is the sole judge of scientific truth."

- *Professor Richard Feynman*

My undergraduate thesis has perhaps been the most demanding, yet gratifying academic endeavour I have had the utmost joy of undertaking. The essential knowledge I have acquired is a rudimentary understanding of the scientific process underlying conducting research. Moreover, I have ascertained that this scientific progression of beginning with a deceivingly simple question, conducting a literature review to illuminate similar work, developing and proposing a hypothesis, generating results, evaluating the accuracy of the generated results, and investigating alternative solutions as needed, is innate to conducting research.

Through this process I have come to the realization that unquenchable curiosity and zeal lie at the heart of research, as well as a great deal of grit, tenacity, and resilience. As a result, my admiration for PhD students has strengthened immensely. I have also discovered that I enjoy the challenges associated with this scientific process. Although conducting research is often grueling, I believe the satisfaction of achieving a result, however trivial, is absolutely worth it.

In addition, I gained a fundamental technical understanding of optimal control; specifically receding horizon control, vehicle dynamics, and vital mathematical concepts such as spline parameterization and the Convex Hull Property.

I am beyond grateful for this learning opportunity. Thank you.

REFERENCES

"Traffic safety facts annual report tables", `https://cdan.nhtsa.gov/tsftables/tsfar.htm`, accessed: 2023-05-02 (2020).

Althoff, M., M. Koschi and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads", in "2017 IEEE Intelligent Vehicles Symposium (IV)", pp. 719–726 (IEEE, 2017).

Amatya, S., M. Ghimire, Y. Ren, Z. Xu and W. Zhang, "When shall i estimate your intent? costs and benefits of intent inference in multi-agent interactions", in "2022 American Control Conference (ACC)", pp. 586–592 (IEEE, 2022).

Andersson, J. A., J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control", Mathematical Programming Computation **11**, 1–36 (2019).

Cohen, E., T. Lyche and R. Riesenfeld, "4. carl deboor, a practical guide to splines, springer-verlag, new york, 1978. 5. kj versprille," computer-aided design applications of the rational b-spline approximation form," phd disser-tation, syracuse univ., syracuse, ny, feb. 1975.", (1978).

Korosec, K., "Gm's cruise recalls, updates software in 80 robotaxis following crash", URL `https://techcrunch.com/2022/09/01/gms-cruise-recalls-updates-software-in-80-robotaxis-following-crash/` (2022).

Löw, T., T. Bandyopadhyay and P. V. Borges, "Identification of effective motion primitives for ground vehicles", in "2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 2027–2034 (IEEE, 2020).

Mangasarian, O. L., *Nonlinear programming* (SIAM, 1994).

Meco-Group, "Meco-group/omg-tools: Optimal motion generation-tools: Motion planning made easy", URL `https://github.com/meco-group/omg-tools` (2016).

Mercy, T., R. Van Parys and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment", IEEE Transactions on Control Systems Technology **26**, 6, 2182–2189 (2017).

Naidu, D. S., *Optimal control systems* (CRC press, 2002).

Polack, P., F. Altché, B. d'Andréa Novel and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?", in "2017 IEEE intelligent vehicles symposium (IV)", pp. 812–818 (IEEE, 2017).

Richards, M., *Software architecture patterns*, vol. 4 (O'Reilly Media, Incorporated 1005 Gravenstein Highway North, Sebastopol, CA , 2015).

Schwarting, W., A. Pierson, J. Alonso-Mora, S. Karaman and D. Rus, “Social behavior for autonomous vehicles”, Proceedings of the National Academy of Sciences **116**, 50, 24972–24978 (2019).

Sun, L., M. Cai, W. Zhan and M. Tomizuka, “A game-theoretic strategy-aware interaction algorithm with validation on real traffic data”, in “2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)”, pp. 11038–11044 (IEEE, 2020).

Van Parys, R. and G. Pipeleers, “Distributed mpc for multi-vehicle systems moving in formation”, Robotics and Autonomous Systems **97**, 144–152 (2017).

Wang, W., L. Wang, C. Zhang, C. Liu, L. Sun *et al.*, “Social interactions for autonomous driving: A review and perspectives”, Foundations and Trends® in Robotics **10**, 3-4, 198–376 (2022).

Wang, Y., Y. Ren, S. Elliott and W. Zhang, “Enabling courteous vehicle interactions through game-based and dynamics-aware intent inference”, IEEE Transactions on Intelligent Vehicles **5**, 2, 217–228 (2019).

Zhan, W., L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle *et al.*, “Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps”, arXiv preprint arXiv:1910.03088 (2019).

APPENDIX A

DEFINITIONS

OCP: (Optimal Control Problem) Defined at a high-level, an optimal control problem is concerned with acquiring a control law that satisfies the optimality criterion for a specific system Naidu (2002).

NLP: (Non-linear Programming Problem) Defined at a high-level, a mathematical model is referred to as an NLP if the objective function contains nonlinear functions and/or constraints applied are nonlinear equalities or inequalities Mangasarian (1994).

APPENDIX B

SOURCE CODE

Please refer to the following GitHub repository in order to acquire the respective source code: https://github.com/sganti2/Social-Navigation-Autonomous-Vehicles

B.1 Case 1: Single-Agent Interaction

B.1.1 Quadrotor Model

```python
# This file is not part of OMG-tools.

from omgtools import *

# create vehicle
vehicle = Quadrotor()
vehicle.define_knots(knot_intervals=5)  # choose lower amount of
    knot intervals

vehicle.set_initial_conditions([0., 0.])
vehicle.set_terminal_conditions([3., 3.])

# create environment
environment = Environment(room={'shape': Square(5.), 'position':
    [1.5, 1.5]})
environment.add_obstacle(Obstacle({'position': [1., 1.]}, shape=
    Circle(0.5)))

# create a point-to-point problem
problem = Point2point(vehicle, environment, freeT=True)
# extra solver settings which may improve performance
#problem.set_options({'solver_options': {'ipopt': {'ipopt.
    linear_solver': 'ma57'}}})
problem.init()

vehicle.problem = problem  # to plot error if using substitution

# create simulator
simulator = Simulator(problem)
problem.plot('scene')
vehicle.plot('input', knots=True, label=['Thrust force (N/kg)',
                                         'Pitch rate (rad/s)'])

# run it!
simulator.run()
problem.save_movie('scene', format='gif', name='problemgif',
    number_of_frames=80, movie_time=4, axis=False)
```

B.1.2 Bicycle Model

```python
# This file is not part of OMG-tools.

from omgtools import *

# create vehicle
vehicle = Bicycle(length=0.4, options={'plot_type': 'car', 'substitution': False})
vehicle.define_knots(knot_intervals=5)  # choose lower amount of knot intervals

vehicle.set_initial_conditions([0., 0., 0., 0.])  # x, y, theta, delta
vehicle.set_terminal_conditions([3., 3., 0.])  # x, y, theta

# create environment
environment = Environment(room={'shape': Square(5.), 'position': [1.5, 1.5]})
environment.add_obstacle(Obstacle({'position': [1., 1.]}, shape=Circle(0.5)))

# create a point-to-point problem
problem = Point2point(vehicle, environment, freeT=True)
# extra solver settings which may improve performance
#problem.set_options({'solver_options': {'ipopt': {'ipopt.linear_solver': 'ma57'}}})
problem.init()

vehicle.problem = problem  # to plot error if using substitution

# create simulator
simulator = Simulator(problem)

problem.plot('scene')
vehicle.plot('input', knots=True, labels=['v (m/s)', 'ddelta (rad/s)'])
vehicle.plot('state', knots=True, labels=[
             'x (m)', 'y (m)', 'theta (rad)', 'delta (rad)'])
if vehicle.options['substitution']:
  vehicle.plot('err_pos', knots=True)
  vehicle.plot('err_dpos', knots=True)

# run it!
simulator.run()

problem.save_movie('scene', format='gif', name='problemgif', number_of_frames=80, movie_time=4, axis=False)
```

B.2 Case 2: Multi-Agent Interaction

B.2.1 Quadrotor Model

```python
from omgtools import *

# create fleet
N = 2
vehicles = [Quadrotor(0.2) for l in range(N)]
fleet = Fleet(vehicles)

configuration = [[0.], [-0.]]
init_positions = [[0., 0.], [0.8, 0.5]]
terminal_positions = [[3., 3.], [3., 3.]]

fleet.set_configuration(configuration)
fleet.set_initial_conditions(init_positions)
fleet.set_terminal_conditions(terminal_positions)

# create environment
environment = Environment(room={'shape': Square(5.), 'position':
    [0., 2.]})
environment.add_obstacle(Obstacle({'position': [0., 1.5]}, shape=
    Circle(0.4)))

# create a formation point-to-point problem
options = {'horizon_time': 5., 'codegen': {'jit': False}, 'rho': 3.}
problem = RendezVous(fleet, environment, options=options)
problem.init()

# create simulator
simulator = Simulator(problem)
problem.plot('scene')
fleet.plot('input', knots=True)

# run it!
simulator.run()
problem.save_movie('scene', format='gif', name='problemgif',
    number_of_frames=80, movie_time=4, axis=False)
```

B.2.2 Bicycle Model

```python
# This file is not part of OMG-tools.
# Author: Sruti Ganti
# Date: March 8, 2023
import numpy as np

from omgtools import *

# create fleet
N = 2
vehicles = [Bicycle(length=0.4, options={'plot_type': 'car', '
    substitution': False}) for l in range(N)]
for vehicle in vehicles:
    vehicle.define_knots(knot_intervals=5)

fleet = Fleet(vehicles)
configuration = [[1.], [-1.]]
init_positions = [[-1.5, 0.5], [0.8, 0.5]]
terminal_positions = [[2., 2.], [2., 2.]]

init_pose = np.c_[init_positions, np.zeros(N), np.zeros(N)]
term_pose = np.c_[terminal_positions, np.zeros(N)]

fleet.set_configuration(configuration)
fleet.set_initial_conditions(init_pose.tolist())
fleet.set_terminal_conditions(term_pose.tolist())

# create environment
environment = Environment(room={'shape': Square(5.), 'position':
    [0., 2.]})
environment.add_obstacle(Obstacle({'position': [0., 1.75]}, shape=
    Circle(0.5)))

# create a formation point-to-point problem
options = {'horizon_time': 5., 'codegen': {'jit': False}, 'rho': 3.}
problem = RendezVous(fleet, environment, options=options)
problem.init()

# create simulator
simulator = Simulator(problem)
problem.plot('scene')
fleet.plot('input', knots=True)

# run it!
simulator.run()
problem.save_movie('scene', format='gif', name='problemgif',
    number_of_frames=80, movie_time=4, axis=False)
```