

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

CSR-26-11

Dynamic Navigation Function in CARLA Simulation Environment

S M Tanvir Abid · Julkar Nine · Wolfram Hardt

November 2026

Chemnitzer Informatik-Berichte



TECHNISCHE UNIVERSITÄT
CHEMNITZ

Dynamic Navigation Function in CARLA Simulation Environment

Master Thesis

Submitted in Fulfilment of the
Requirements for the Academic Degree
M.Sc.

Dept. of Computer Science
Chair of Computer Engineering

Submitted by: S M Tanvir Abid
Student ID: 625881
Date: 07.05.2026

Supervising tutor: Prof. Dr. W. Hardt
MSc. Julkar nine

Abstract

Autonomous navigation in simulation environments requires dynamic navigation methods that remain consistent with the conditions under which a simulation run is actually executed. In many navigation workflows, routes are computed on a nominal road network representation, while scenario specific environmental constraints are introduced only at the execution stage. This can create a mismatch between the configured simulation environment and the route assigned to the ego vehicle. This thesis addresses this problem in the CARLA simulation environment by developing a navigation framework in which route generation is adapted to the environmental conditions of each simulation run before execution begins.

This thesis addresses dynamic navigation under scenario dependent environmental conditions in the CARLA simulation environment. The objective is to develop a navigation framework in which the generated route reflects the effective road availability of a specific simulation run before execution begins. For this purpose, the drivable road network is represented as a graph and adapted according to blocked or undesired road segments defined by the active scenario. A valid route is then computed on the modified planning graph and executed by the ego vehicle in the simulator. In this way, the work realizes a structured form of dynamic navigation at the level of run-specific route generation rather than continuous online replanning during motion.

The framework was implemented in Python and integrated into CARLA as a complete pipeline including scenario realization, graph construction, constraint integration, route generation, route execution, visualization, and CSV-based logging. The developed approach was evaluated in Town02 and Town01 under nominal and constrained scenario modes using route feasibility, route length, travel time, executed distance, speed-related indicators, planning time, and qualitative route observation. The results show that valid routes could be generated in all evaluated scenarios and that increasing scenario constraints led to visible and measurable route adaptation while planning times remained low. Within the defined scope of the thesis, the work demonstrates that scenario-dependent environmental constraints can be integrated directly into route generation in CARLA in a technically interpretable, computationally manageable, and experimentally reproducible way.

Keywords: Dynamic navigation, autonomous driving, CARLA simulation path planning, adaptive routing

Contents

Contents . . . 3

List of Figures . . . 5

List of Tables . . . 6

List of Abbreviations . . . 7

1 Introduction . . . 8
- 1.1 Background and Context . . . 9
- 1.2 Motivation . . . 10
- 1.3 Problem Statement . . . 12
- 1.4 Objective and Scope of the Thesis . . . 13
- 1.5 Research Contribution . . . 14
- 1.6 Thesis Structure . . . 16

2 State of the Art . . . 17
- 2.1 Simulated Path Planning . . . 17
  - 2.1.1 Simulation Tools . . . 17
  - 2.1.2 Comparison of Simulation Tools . . . 19
  - 2.1.3 Analytical comparison . . . 21
  - 2.1.4 Rationale for selecting CARLA . . . 22
  - 2.1.5 Discussion & Platform Choice – Decision-tree Diagram . . . 23
- 2.2 Path-Planning Foundations in Simulation . . . 24
  - 2.2.1 Layered Path Planning Stack . . . 24
  - 2.2.2 Graph-Search Shortest-Path . . . 26
  - 2.2.3 Optimization-Based Motion Planning . . . 29
  - 2.2.4 Sampling & Search-Based Planning . . . 31
  - 2.2.5 Learning-Based & Hybrid Approaches . . . 32
- 2.3 Evaluation Metrics and Benchmarks for Autonomous Navigation . . . 35
  - 2.3.1 Safety Metrics . . . 35
  - 2.3.2 Navigation Efficiency Metrics . . . 36
  - 2.3.3 Computational Performance Metrics . . . 37
- 2.4 NPC and Traffic Control Mechanisms in Simulation . . . 38
  - 2.4.1 Built-in Traffic Control Systems in Simulation Environments . . . 40
  - 2.4.2 Scenario Description Frameworks and Scripted Traffic Generation . . . 43

2.4.3 Interaction Between Planner and Traffic Agents: Social and Negotiation Models . . . 45
2.4.4 Impact on Planner Evaluation . . . 47
2.5 Simulation Fidelity vs Real-Time Performance . . . 49
2.5.1 Physics and Rendering Trade-offs . . . 50
2.5.2 Determinism and Stochasticity in Simulation . . . 51
2.5.3 Computational Bottlenecks and System Constraints . . . 52
2.5.4 Scalability and System-Level Limitations . . . 54
2.6 Analysis of Existing Approaches and Identified Gaps . . . 56
2.6.1 Limitations in Planning Approaches . . . 57
2.6.2 Limitations in Simulation-Based Evaluation . . . 58
2.6.3 Identified Gap and Positioning . . . 60

3 Methodology . . . 62
3.1 Overview of the Proposed Methodology . . . 62
3.2 Environment Aware Route Adaptation for Navigation . . . 64
3.3 Route Preparation and Execution . . . 66
3.4 Methodological Summary . . . 67

4 Implementation . . . 68
4.1 Implementation Overview . . . 68
4.2 System Components and Software Structure . . . 70
4.3 Scenario Realization in Town02 and Town01 . . . 72
4.4 Graph Construction and Constraint Integration . . . 74
4.5 Route Generation and Execution Pipeline . . . 77
4.6 Visualization and Logging . . . 79
4.7 Implementation Summary . . . 81

5 Results and Evaluation . . . 82
5.1 Evaluation Setup . . . 82
5.2 Evaluation of the Navigation Framework in Town02 . . . 84
5.3 Evaluation of the Navigation Framework in Town01 . . . 88
5.4 Cross-Town Applicability of the Navigation Framework . . . 92
5.5 Interpretation of the Evaluation Results . . . 94
5.6 Scope and Limitations of the Evaluation . . . 95

6 Discussion and Future Work . . . 97
6.1 Discussion . . . 98
6.2 Future Work . . . 100

7 Conclusion . . . 102

Bibliography . . . 104

List of Figures

2.1 NPC-Control Capability Matrix For Different Simulators 22
2.2 Decision-Tree Diagram for Selecting CARLA 23
2.3 Layered planning architecture for autonomous vehicles 25
2.4 Conceptual architecture of a learning-assisted navigation system . . . 34
2.5 Evaluation pipeline for autonomous navigation algorithms in simulation environments 38
2.6 The architectural differences between traffic control mechanisms in CARLA, SUMO, and VTD 42
2.7 Role of scenario description frameworks in simulation-based evaluation 44
2.8 Conceptual representation of interaction-aware navigation in multi-agent environments 46

3.1 Conceptual workflow of the proposed navigation methodology in CARLA 63
3.2 Environment-aware navigation route adaptation under changed scenario conditions 65

4.1 System-level architecture of the implemented navigation framework in CARLA 69

5.1 Average planned route length and average travel time across the evaluated scenario modes in Town02. 86
5.2 Representative split-screen execution view of the implemented navigation framework in Town02 87
5.3 Average planned route length and average travel time across the evaluated scenario modes in Town01. 90
5.4 Representative split-screen execution view of the implemented navigation framework in Town01 91
5.5 Cross-town comparison of average planned route length and average travel time across the evaluated scenario modes in Town02 and Town01.. 93

List of Tables

2.1 Licensing and rendering characteristics of candidate simulators 20
2.2 Empirical sensor realism and traffic-scenario tooling 20
2.3 Planner-integration interfaces and prevailing research focus 20
2.4 Complexity Comparison 31
2.5 Trade-offs between simulation fidelity and scalability in autonomous driving environments. 55

5.1 Overview of the evaluated scenario configurations in Town02 and Town01 84
5.2 Planning feasibility and main navigation performance results for the Town02 evaluation 85
5.3 Execution related and computational results for the Town02 evaluation 85
5.4 Scenario structure and robustness metrics for the Town02 evaluation . 86
5.5 Planning feasibility and main navigation performance results for the Town01 evaluation 88
5.6 Execution-related and computational results for the Town01 evaluation 89
5.7 Scenario-structure and robustness metrics for the Town01 evaluation . 90
5.8 Cross-town applicability of the implemented navigation framework across Town02 and Town01 92

List of Abbreviations

API	Application Programming Interface
ASAM	Association for Standardisation of Automation and Measuring Systems
AV	Autonomous Vehicle
CARLA	Car Learning to Act
CSV	Comma-Separated Values
FPS	Frames Per Second
GUI	Graphical User Interface
HIL	Hardware-in-the-Loop
IDM	Intelligent Driver Model
IMU	Inertial Measurement Unit
ISO	International Organization for Standardization
LiDAR	Light Detection and Ranging
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
NPC	Non-Player Character
PRM	Probabilistic Roadmap
RADAR	Radio Detection and Ranging
RL	Reinforcement Learning
RMSE	Root Mean Square Error
ROS	Robot Operating System
RRT	Rapidly-exploring Random Tree
RRT*	Rapidly-exploring Random Tree Star
SDK	Software Development Kit
SUMO	Simulation of Urban MObility
SVL	Silicon Valley Lab
TM	Traffic Manager
TraCI	Traffic Control Interface
USD	Universal Scene Description
VTD	Virtual Test Drive
XML	Extensible Markup Language

1 Introduction

Autonomous driving has become one of the central research areas in modern intelligent transportation systems. Its development depends on the reliable interaction of perception, decision-making, planning, and control, all of which must operate under complex and changing environmental conditions. Among these functions, navigation plays a fundamental role because it determines how a vehicle reaches a destination through a road network while respecting the practical constraints of the surrounding environment [53].

In real traffic as well as in simulation, route selection cannot always be treated as a purely static problem. Road segments may become unavailable, traffic conditions may vary, and the feasibility of a route can depend on a specific situation where the driving task is executed. This is specially relevant in autonomous driving research. In this field, simulation environments are widely used to investigate system behavior under controlled as well as variable conditions. Simulation enables repeatable experimentation, systematic testing of challenging scenarios, as well as safe validation of navigation strategies that would be difficult or costly to assess exclusively on public roads [32], [21].

Within this context, the present thesis addresses route generation in the CARLA simulation environment under scenario-dependent environmental conditions. The work focuses on a navigation function that adapts the route to the conditions of a given simulation run before execution begins. This means that the route is not continuously replanned during driving. Instead, relevant environmental constraints are incorporated into the road-network representation, and a valid route is then generated for that run. The ego vehicle subsequently follows this route in the simulator. This formulation allows dynamic adaptation at the level of run-specific route generation while remaining compatible with a structured and computationally manageable simulation workflow.

In order to establish the background for this work, the present chapter first introduces the broader context of autonomous navigation. It introduces simulation based development as well. It then motivates the problem addressed in this thesis, defines its objectives and scope, and then outlines the contribution as well as structure of the work. Based on this foundation, next chapter then reviews the relevant literature on simulation tools, path-planning methods, traffic mechanisms, evaluation criteria, and the limitations of existing approaches.

1.1 Background and Context

Autonomous driving systems are commonly organized as layered architectures in which high-level navigation, tactical decision-making, and low-level motion generation interact to produce vehicle behavior. In such systems, navigation is responsible for determining a route from a start location to a destination on the basis of the available road network, while downstream planning components refine this route into maneuvers and executable trajectories [53]. Although this decomposition improves modularity, it also means that the quality of global route selection has a direct influence on the effectiveness of the subsequent planning and control stages.

The importance of simulation in this field is closely related to the difficulty of validating autonomous driving systems exclusively through real-world testing. Safety-critical events are often rare, and statistically meaningful evaluation would require extremely large driving mileage. Kalra and Paddock showed that demonstrating autonomous vehicle reliability through road testing alone would demand a scale of experimentation that is difficult to achieve in practice [32]. For this reason, simulation has become a central methodological tool in autonomous driving research. It supports controlled testing, repeatability, rapid scenario variation, and safe experimentation in conditions that would otherwise be difficult to reproduce [21], [25].

Among the available simulation platforms, CARLA has become one of the most prominent open research environments for autonomous driving. CARLA was introduced as an open urban driving simulator that supports the development and evaluation of autonomous driving systems under configurable weather, traffic, and sensor settings [21]. Its relevance extends beyond sensor simulation and visual realism. From the perspective of navigation research, CARLA is especially useful because it provides access to map information, waypoint structures, traffic agents, as well as programmable interfaces that enable controlled integration of route planning with the simulation environment. This makes it a suitable platform for investigation of the influence of environmental constraints on route generation in a reproducible experimental setting.

At the route-planning level, the road network is typically represented as a graph whose nodes and edges describe reachable parts of the drivable infrastructure. Traditional graph search based methods, for example, Dijkstra's algorithm and A* remain highly relevant because they provide transparent and computationally efficient solutions for shortest-path problems on structured networks [19], [29]. In a simulator like CARLA, the road structure can be mapped to connected waypoints or lane segments. And these methods offer a practical basis for route generation. They also make it possible to integrate environmental constraints directly into the planning space by removing, modifying, or reweighting parts of the graph that should no longer be considered during route computation.

However, the suitability of a route does not depend only on the nominal geometry of the road network. It can also depend on the conditions under which the simulation run is executed. A path that is valid in one case may be unsuitable in another case if the relevant road segment is blocked or if the configured environment makes the route

practically undesirable. This issue becomes important when navigation is evaluated in scenario driven simulation workflows. In such workflows, the environment is not merely a neutral background. It is an active part of the experimental design, because traffic configurations, blockages, as well as other conditions are deliberately varied in order to examine system behavior [24], [22].

This creates an important methodological perspective for the present thesis. If route planning is performed independently of the environmental configuration of a simulation run, then the resulting route may not reflect the conditions under which the system is actually evaluated. In that case, the route remains tied to a nominal map, while the experiment itself assumes a modified environment. For a study that focuses on blocked roads and scenario-dependent route validity, this separation is not sufficient. Instead, the navigation process must consider the relevant environmental information before the route is assigned to the ego vehicle.

The present thesis is positioned within this problem setting. It investigates environment aware route generation in CARLA, where the road graph is adapted according to the conditions of a given simulation run and the resulting route is then executed by the vehicle. The work therefore addresses a form of dynamic navigation at the level of scenario dependent route computation. This distinction is central to the scope and contribution of the thesis and will be developed further in the following sections.

1.2 Motivation

Reliable navigation is a fundamental requirement for autonomous driving because route selection influences not only whether a destination can be reached, but also how safely and efficiently the vehicle can operate on the way. In practical driving environments, however, the suitability of a route cannot always be determined from the nominal road network alone. Temporary road blockages, traffic disturbances, or scenario-specific environmental constraints may render a nominally valid route ineffective for a particular run. This makes it necessary to consider not only map topology, but also the conditions under which navigation is executed [53].

From a methodological perspective, this issue is especially relevant in simulation based autonomous driving research. Simulation is widely used because it allows controlled experimentation under repeatable conditions, supports systematic variation of traffic situations, as well as enables testing of cases that would be difficult, costly, or unsafe to reproduce on public roads [21], [32]. At the same time, the value of a simulation study depends on whether the navigation logic is actually compatible with the configured environment. If the simulated world is deliberately changed by introducing blocked roads or altered traffic situations, but the route remains based on an unmodified nominal graph, then the experiment no longer reflects a coherent navigation process. The motivation for this thesis therefore arises from the need to connect environment configuration and route generation more directly.

A further motivation lies in the role of scenario based validation in autonomous

driving. Formal and scenario driven testing approaches have shown that structured scenario specification is essential for evaluating autonomous systems under meaningful and reproducible conditions [25], [22]. Yet in many workflows, the scenario mainly affects the execution environment, while the route itself is treated as fixed. This creates a practical limitation when the experimental objective concerns route validity under changed road conditions. In such cases, it is not sufficient to observe whether the vehicle reacts locally during execution. It is also necessary to examine whether the route should have been generated differently from the beginning.

The motivation of this thesis is therefore not to replace established planning architectures with a fully online replanning system. Instead, it is to address a more targeted and practically relevant problem in CARLA: how to generate a route that already reflects the environmental conditions of a given simulation run before the ego vehicle begins execution. This perspective is particularly useful in scenario driven experiments where the environment is known at the start of the run and where structured adaptation at the route-generation stage can improve the consistency of the overall navigation workflow.

There is also a practical engineering motivation for this approach. High frequency replanning and interaction aware online adaptation can be computationally demanding, especially in simulator environments where rendering, physics, traffic management, and agent execution already share system resources [53]. For many thesis scale simulation studies, a more structured route generation process is therefore methodologically attractive. It allows route adaptation to be examined in a transparent and reproducible way without requiring a full online decision architecture. In this sense, the thesis is motivated both by scientific relevance and by implementation feasibility.

Finally, the work is motivated by the specific characteristics of CARLA as a research platform. CARLA offers programmable access to road maps, waypoint structures, traffic actors, and scenario-relevant environment control, which makes it suitable for investigating environment-aware route generation under controlled conditions [21]. This creates an opportunity to study how graph based planning can be combined with scenario dependent road network modification in a way that is experimentally clear, technically interpretable, and suitable for comparative evaluation. The resulting motivation is therefore twofold. One is to improve the consistency between scenario configuration and route planning, and another is to establish a reproducible framework for evaluating environment-aware navigation in CARLA.

1.3 Problem Statement

Autonomous navigation requires more than the computation of a geometrically valid path between a start and a destination. In practical driving environments, the suitability of a route depends on whether the road network remains traversable under the conditions present at the time of execution. A route that is optimal on a nominal map may become ineffective if relevant road segments are blocked, temporarily unavailable, or otherwise unsuitable in the configured environment. This problem is well aligned with the broader planning literature, which treats navigation in autonomous driving as a task that must account for constraints arising from dynamic and structured environments [53].

In simulation based research, this challenge appears in a particularly clear form. Modern autonomous driving simulators support controlled changes to the environment, including traffic composition, obstacle placement, and structured scenario execution [21], [25], [22]. However, route generation is often still performed on a nominal road representation that does not explicitly incorporate the environmental conditions defined for a particular simulation run. As a result, the navigation system may be evaluated in a world whose relevant constraints are not fully reflected in the route assigned to the ego vehicle. This creates an inconsistency between the simulation scenario and the route planning process.

The specific problem addressed in this thesis therefore lies at the interface between route planning and scenario dependent environment configuration. When a road segment is blocked in a given run, or when a scenario is deliberately configured in such a way that the nominal route is no longer desirable, the navigation system should not simply rely on an unchanged baseline route. Instead, the route generation step should reflect the environmental conditions under which the run is executed. If this does not happen, the ego vehicle may be directed toward segments that are unavailable or practically unsuitable, which can lead to inefficient behavior, failed navigation, or a distorted evaluation of the planning approach.

This issue is especially relevant for graph-based route planning in CARLA. CARLA provides access to road topology and waypoint based map structures that are suitable for shortest path search and related planning procedures [21]. Traditional graph search based methods like Dijkstra's algorithm and A* are highly effective when the underlying graph accurately represents the set of feasible road segments [19], [29]. The difficulty arises when the graph used for route generation continues to represent the nominal road network, while the actual simulation run includes additional constraints such as blocked areas or environment specific exclusions. In that case, the graph remains formally complete, but it no longer corresponds to the effective navigation space of the experiment.

The problem considered in this thesis is not identical to continuous online replanning. The aim is not to develop a navigation system that recomputes its global route at every instant during motion. Rather, the problem is defined at the level of run specific route generation. Before the ego vehicle begins execution, the environment of the current simulation run is analyzed with respect to the relevant constraints,

and the road graph used for planning is adapted accordingly. A route is then computed on this modified representation and subsequently executed in the simulator. The problem can therefore be stated as follows: how can a navigation function in CARLA generate a valid route that reflects the environment conditions of a given simulation run before execution begins, instead of relying on a static nominal route?

This formulation is both technically precise and methodologically relevant. It focuses on the consistency between the configured simulation environment and the route that the vehicle is asked to follow. At the same time, it avoids overstating the contribution as a fully reactive online replanning framework. The thesis instead addresses a more specific but practically meaningful problem in simulation based autonomous navigation. This is the integration of scenario dependent environmental constraints into route generation at the beginning of simulation run.

1.4 Objective and Scope of the Thesis

The main objective of this thesis is to develop and evaluate an environment aware navigation function in the CARLA simulation environment that generates a valid route according to the conditions of a specific simulation run. The work focuses on the route generation stage of navigation and investigates how blocked road information and other scenario dependent environmental constraints can be incorporated into the planning process before the ego vehicle begins to drive. In this way, the thesis addresses the problem of route validity under changed simulation conditions while remaining within a structured graph based planning framework [21], [53].

More specifically, the thesis aims to represent the relevant drivable road network in CARLA as a graph, to adapt this graph according to the environmental conditions of a given run, and to compute a route on the basis of the modified graph. The generated route is then executed by the ego vehicle inside the simulator. The intended result is a navigation process in which the assigned route is no longer based solely on the nominal road topology, but instead reflects the effective road availability of the current scenario. This objective is methodologically relevant because it strengthens the consistency between environment configuration and route-level planning in simulation-based evaluation.

A further objective of the thesis is to demonstrate that such an approach can be implemented transparently and reproducibly within CARLA using a classical graph search pipeline. Rather than relying on a black box decision architecture, the work uses an interpretable route generation process in which the effect of environmental constraints can be traced directly through the underlying road graph and the resulting route computation. This is specially useful for experimentation because it supports clear system design, controlled comparison, as well as understandable failure analysis [19], [29].

Along with design and implementation of the navigation function, the thesis also aims to evaluate the proposed approach in multiple simulation scenarios. This includes the analysis of route behavior under different blockage conditions and across

different CARLA towns, as well as a comparison with baseline route generation on the unmodified road network. The purpose of this evaluation is not only to show that the vehicle can reach the destination under selected constraints, but also to examine whether environment aware route generation leads to a more coherent navigation process in scenario-dependent simulation settings.

The scope of the thesis is intentionally limited in several ways. First, the work is restricted to simulation based experimentation in CARLA and does not claim direct deployment on a physical autonomous vehicle platform. Second, the thesis focuses on global route generation under run specific environmental constraints. It does not aim to develop a full end-to-end autonomous driving stack with advanced perception, prediction, and continuous on the fly decision making. Third, the proposed navigation logic that the route is adapted before execution begins, based on the environmental state of the current run, and is then followed by the ego vehicle during simulation. This distinction is central to the definition of the contribution and must be maintained clearly throughout the thesis.

The scope is also limited with respect to the type of environmental adaptation considered. The thesis concentrates on structured road network constraints, particularly blocked or undesired road segments that influence route feasibility at the graph level. It does not attempt to model the full complexity of probabilistic multi agent interaction, or long horizon online route revision under continuously changing uncertainty. Such capabilities are important in the broader field of autonomous driving, but they would require a different system architecture and a substantially wider experimental framework [61], [53].

Within these boundaries, the thesis makes a focused contribution that it investigates how scenario dependent environmental information can be integrated into route generation in CARLA in a way that is technically precise, computationally manageable, as well as suitable for reproducible evaluation. The resulting scope is therefore narrow enough to remain methodologically clear, but broad enough to provide a meaningful contribution to simulation-based navigation research.

1.5 Research Contribution

The contribution of this thesis lies in the development of a structured navigation approach for CARLA in which route generation is explicitly linked to the environmental conditions of a given simulation run. In contrast to workflows that compute a route only on the basis of the nominal road network, the proposed approach incorporates scenario-dependent constraints into the graph used for planning before the ego vehicle begins execution. This creates a navigation process in which route computation reflects the effective availability of the road network for the current run rather than relying solely on a static baseline representation [21], [53].

A first contribution of the thesis is therefore conceptual. The work defines a form of dynamic navigation that is based on run-specific environment-aware route generation rather than continuous online replanning. This distinction is important because

the term dynamic navigation is often associated with reactive or continuously updated planning systems. The present thesis adopts a more precise interpretation that is aligned with the implemented system: the route is adapted to the known conditions of the simulation scenario before execution starts, and the ego vehicle then follows the computed route during the run. This positioning contributes methodological clarity by separating scenario-dependent route adaptation from fully online replanning architectures [61].

A second contribution is the integration of environmental constraints into a graph-based route-planning pipeline in CARLA. The thesis demonstrates how the road topology of the simulator can be represented as a planning graph and how blocked or undesired road segments can be excluded from the effective search space before route computation. This makes the route-generation step directly responsive to scenario configuration while preserving the interpretability and transparency of classical graph-search methods such as A* [19], [29]. The contribution here is not a new shortest-path algorithm in itself, but a structured way of coupling graph construction and environment configuration for navigation in simulation.

A third contribution is the implementation of this navigation logic within a reproducible CARLA-based framework. The thesis applies the approach in multiple scenario configurations and across different CARLA towns, thereby moving beyond a purely conceptual discussion and providing an executable system-level realization of the proposed idea. The implemented framework combines road-graph construction, environment-dependent graph filtering, route generation, route visualization, and vehicle execution in a single workflow. This contributes practical value because it enables the proposed navigation logic to be examined not only theoretically, but also experimentally in a controlled simulation environment.

A fourth contribution lies in the evaluation perspective adopted in the thesis. The work does not treat route planning as an isolated shortest-path exercise on a fixed map. Instead, it evaluates navigation in the presence of scenario-driven environmental changes and compares the resulting route behavior to baseline planning on the nominal graph. In this way, the thesis contributes to simulation-based navigation research by emphasizing the consistency between the configured environment and the route assigned to the vehicle. This is relevant because scenario based testing has become important in autonomous driving research. The integration of scenario information into route level planning remains very less explicitly addressed in many workflows [25], [22].

Taken together, the thesis makes a focused contribution at the intersection of simulation based evaluation, graph based route planning, as well as scenario dependent environment modeling. It does not claim to provide a complete end-to-end autonomous driving architecture or a full online replanning framework. Its contribution is narrower and more specific: it establishes and evaluates a technically interpretable method for environment-aware route generation in CARLA, where the route is computed according to the conditions of each simulation run before execution begins. Within the defined scope of the work, this represents a meaningful contribution to the study of navigation under structured simulation constraints.

1.6 Thesis Structure

This thesis is organized into seven chapters. Chapter 1 introduces the general context of autonomous navigation and motivates the problem addressed in this work. It defines the problem statement, outlines the objective and scope of the thesis. beside, it explains the main contribution of the proposed navigation framework.

Chapter 2 presents the state of the art. It reviews simulation tools for autonomous driving, navigation and path planning foundations, evaluation metrics, traffic control mechanisms, and the trade-offs between simulation fidelity and computational performance. The chapter concludes by analyzing the limitations of existing approaches as well as identifying the research gap addressed in this thesis.

Chapter 3 describes the proposed methodology. It explains the overall navigation concept, the principle of environment-aware route adaptation, and the preparation of the generated route for execution in CARLA. The chapter provides the conceptual basis for the implemented framework.

Chapter 4 presents the implementation of the proposed approach. It describes the software structure of the developed system, the realization of the navigation scenarios in Town02 and Town01, the graph-construction and constraint-integration process, the route-generation and execution pipeline, and the visualization and logging components used during the simulation runs.

Chapter 5 reports the results and evaluation of the implemented framework. It first defines the evaluation setup and then presents the results for Town02 and Town01 separately. The chapter also examines the applicability of the framework across both towns and interprets the evaluation results in relation to the objectives of the thesis.

Chapter 6 discusses the significance of the developed framework and reflects on its contribution within the defined scope of the work. It also outlines possible directions for future research based on the implemented system and the findings of the evaluation.

Finally, Chapter 7 concludes the thesis by summarizing the central objectives, implementation results, and evaluation findings, and by restating the contribution of the work to environment aware navigation and route generation in CARLA.

2 State of the Art

Autonomous vehicle (AV) path planning is ultimately a safety issue: each trajectory the planner generates must maintain the vehicle within the law and steer clear of collisions with infrastructure and dynamic agents. Statistical studies indicate that tens of billion kilometers are required to show parity with human drivers for rare event safety, despite the fact that real-world mileage is slow to accrue and costly (fuel, safety drivers, test-track rental). Relaying, instrumenting, and stress testing such edge scenarios without putting lives in danger is made possible via closed-loop simulation.

2.1 Simulated Path Planning

For rare fatal accidents, statistical analyses show that hundreds of millions to billions of road kilometers are required to show parity with human drivers [32]. That distance would take decades, even for ambitious real-world test fleets. Carefully biased simulations can reduce such needs by four orders of magnitude, saving both calendar time and money, according to accelerated evaluation frameworks [81].

2.1.1 Simulation Tools

CARLA

CARLA is a standerd open source research simulator. Built on Unreal Engine, which exposes Python and ROS 2 APIs and configurable sensor suites (RGB, LiDAR, RADAR, event camera) and an extensible scenario runner. Dosovitskiy et al. demonstrated its utility for benchmarking modular, imitation learning and reinforcement-learning planners, all evaluated under systematically varied weather and traffic conditions [21]. For many modeling and RL investigations, CARLA is the standard option due to its active GitHub community and permissive MIT license.

SVL

Although it is implemented in Unity and comes with native bridges for both the Apollo and Autoware autonomy stacks, SVL is another full-stack, open-source platform. Rong et al. describe a sensor-accurate engine that preserves hardware-in-the-loop fidelity while enabling users to "hot-swap" perception, planning, or control

modules [56] . While SVL provides simpler multi-sensor calibration operations than CARLA, it currently has a limited collection of prepared maps and traffic scenarios.

AirSim

AirSim, which was first made available for aerial robotics, now has a ground-vehicle mode. A physics-based Unreal Engine backend with real-time hardware-in-the-loop capabilities and APIs for quick dataset creation and domain randomization are described by Shah et al [62]. AirSim performs well when the study topic focuses on perception quality or low-level vehicle dynamics rather than dense interaction because it has little traffic support—researchers usually inject single-ego or scripted agents.

NVIDIA DRIVE Sim

To satisfy ISO 26262 verification requirements, NVIDIA DRIVE Sim combines waveform-level sensor models with an RTX ray-tracing renderer. Several academic organizations have utilized the platform to stress-test planning algorithms in photorealistic landscapes, despite the private nature of the core software. For instance, Koren et al. used DRIVE Sim to look for adversarial planner failures with Adaptive Stress Testing [39]. The primary trade-off is hardware expense; in order to maintain real-time performance, multi-GPU clusters are frequently needed.

Simulation of Urban MObility (SUMO)

SUMO is a microscopic, open-source traffic simulator that operates on network graphs rather than photoreal 3-D scenes. Validated car-following (Krauss), lane-changing, and traffic-light logics that scale to tens of thousands of agents in real time are used at the modeling core of SUMO. López et al. provide a comprehensive overview of these algorithms and document their performance across representative urban scenarios [46]. To achieve realistic contact density at a reasonable computational cost, researchers frequently combine the large-scale traffic generation of SUMO with the high-fidelity representations of CARLA.

Virtual Test Drive (VTD)

From Software-in-the-Loop to Vehicle-in-the-Loop, VTD is a commercial toolchain that focuses on safety-critical verification. In order to assess an autonomous driving function across CarMaker, VTD, and CARLA, Becker et al. created an automated pipeline that modifies road-network topology. Their findings highlight how well VTD handles big OpenDRIVE maps without experiencing frame-rate reductions, which is a requirement for methodical scenario change [3]. To put it briefly, VTD offers scenario tooling and sensor realism of certification quality. This thesis uses CARLA as the main experimental platform, citing VTD results for background,

however its commercial license and GPU requirements make it less accessible for open research than CARLA or SVL.

BeamNG.tech

The lane keeping mechanism of self driving cars was tested using the BeamNG.tech simulator. The realistic physics engine of this simulator sets it apart from the competition by accurately simulating the distinct characteristics of every vehicle, such as braking, traction, and sliding [28]. BeamNG.tech was utilized in a recent lane keeping research by Gullu et al. to assess controller robustness during suspension failure and tire burst situations that are challenging to reproduce in other simulators. Although it only provides basic traffic AI, its strength is in high-fidelity vehicle dynamics [28].

Although all seven platforms are capable of autonomous driving loops, their priorities vary significantly: DRIVE Sim and VTD prioritize certification-grade sensor realism. SUMO scales to city-wide traffic. BeamNG.tech and AirSim focus on fine-grained vehicle dynamics. CARLA and SVL aim to open up academic exploration. Before explaining the rationale for the selection of CARLA for the ensuing adaptive path-planning studies, the following section will methodically quantify these discrepancies.

2.1.2 Comparison of Simulation Tools

To compare the seven shortlisted simulators, six dimensions were selected on the basis of their relevance to situation-aware path-planning research.
(1) License and cost determine the extent to which the results can be reproduced by other researchers and the financial feasibility of large-scale experiments.
(2) Graphics and the physics engine influence the fidelity of sensor outputs and the computational resources required for real-time execution.
(3) Empirical validation of sensor models ensures that simulated data approximate real-world measurements to a degree sufficient for perception-in-the-loop evaluations.
(4) Non-player character (NPC) and scenario control indicate the breadth of traffic behaviors and environment configurations that can be scripted.
(5) Planner integration interfaces refer to the availability of programmatic hooks in the mission, behavior, and trajectory planning layers.
(6) The typical use of research reflects the focus for which each tool is most often cited in the academic literature.

Simulator	License	Engine& Render Stack
CARLA 0.10/0.09x	MIT Open source [21]	Unreal-Engin Based [21]
SVL Simulator	Apache 2.0 open-source [56]	Unity HDRP renderer [56]
AirSim(Car Mode)	MIT open-source [62]	Unreal Engine 4 + PhysX [62]
NVIDIA DRIVESim	Commercial EULA (proprietary) [1]	Omniverse USD scene graph + RTX ray-tracing [1]
SUMO 1.20	Eclipse Public Licence 2.0 [46]	2-D microscopic traffic engine [46]
VTD 2024.3	Commercial [27]	Hexagon VIRES [27]
BeamNG.tech 2.x	Commercial R-&-D licence [26]	Custom soft-body physics engine [26]

Table 2.1: Licensing and rendering characteristics of candidate simulators

Simulator	Peer-reviewed sensor realism	NPC / Scenario Tools
CARLA 0.10/0.09x	R-CARLA shows sub-pixel RGB + LiDAR fidelity in closed-loop racing and urban tasks [8]	Traffic Manager, ScenarioRunner, OpenSCENARIO 2.0) [38]
SVL Simulator	LGSVL paper reports camera/LiDAR RMSE <5 % on real-world test route [56]	Deterministic random-traffic generator and visual scenario editor [79]
AirSim(Car Mode)	Photometric camera noise & IMU bias validated [62]	Scriptable actors plus domain-randomisation API [62]
NVIDIA DRIVESim	Survey on synthetic datasets notes physically-accurate multi-modal sensors via RTX path-tracing [65]	ScenarioGraph GUI with full ASAM OpenSCENARIO 2.0 export [70]
SUMO 1.20	López et al. validate car-following & emission models on real urban data [80]	Built-in lane-change, signal-control and TraCI-driven traffic models [80]
VTD 2024.3	Digital-twin track study shows curvature & superelevation error <0.2 m [59]	ScenarioDesigner GUI, complete OpenSCENARIO 1.x/2.0 support [45]
BeamNG.tech 2.x	spring-mass model accuracy for crash-pulse replication [63]	Lua / Python procedural terrain & scenario generation [6]

Table 2.2: Empirical sensor realism and traffic-scenario tooling

Simulator	Planning APIs	Typical Research Niche
CARLA 0.10/0.09x	Python waypoint & global-route API, ROS 1/2 bridge; CommonRoad–CARLA interface [38]	End-to-end planning and RL benchmarking
SVL Simulator	Python API plus native ROS 2, Apollo & Autoware connectors [56]	Full-stack prototyping on open HD maps
AirSim(Car Mode)	C++/Python RPC for control & sensor streaming [62]	Perception quality and low-level dynamics
NVIDIA DRIVESim	C++/Python SDK linked to the NVIDIA DRIVE AV stack [65]	Certification-grade sensor sweeps & failure search
SUMO 1.20	TraCI socket API (Python/C++/MATLAB) for real-time trajectory steering [80]	City-scale traffic generation & co-simulation
VTD 2024.3	C++ plug-ins; Simulink and dSPACE HIL bridges [59]	ISO 26262 verification pipelines
BeamNG.tech 2.x	BeamNGpy Python library for real-time control & telemetry [6]	Vehicle-dynamics edge-case testing

Table 2.3: Planner-integration interfaces and prevailing research focus

2.1.3 Analytical comparison

Several notable themes emerge from the consolidated comparison.

First – openness versus certified realism

Complete source code is made available under permissive licenses by CARLA, SVL, AirSim, and SUMO, allowing for transparent peer review and community additions [21], [56], [40]. The value proposition of NVIDIA DRIVE Sim and VTD, on the other hand, is a sensor chain that uses RTX-based wave-and ray-optics rendering and is thus compatible with ISO 26262 procedures. However, these programs are offered under proprietary terms [16], [30].

Second – coverage of the three planning layers

Callable interfaces are exposed at the mission (global-route), behavior, and trajectory layers by both SVL and CARLA. While SVL provides ROS-based bridges to Apollo and Autoware that serve a similar purpose [56], CARLA's Waypoint/Route Python API and the CommonRoad-CARLA interface offer learning-based planners, MPC, and conventional graph search in a single framework [12], [48]. Commercial platforms usually leave behavior-layer control to external toolchains and concentrate on perception or HIL integration.

Third – fidelity and controllability of traffic agents

CARLA's Traffic Manager provides fine-grained control over social driving patterns by allowing each non-player vehicle to be allocated unique characteristics (desired speed, gap-keeping, lane-change bias, and traffic-light compliance) [11]. While it exposes fewer per-actor controls, SVL provides a Visual Scenario Editor and deterministic random-traffic seeds [52]. Despite SUMO can be guided online via the TraCI socket API and scalable to tens of thousands of agents, it only functions in two dimensions and is unable to produce raw sensor data [66].

Fourth – computational footprint

Internal release notes for CARLA 0.10.0 (UE5) report 24–25 FPS on a single RTX 4090 in the default Town 10 scene [10]. DRIVE Sim achieves real-time path-traced sensors only on high-end RTX boards and is commonly deployed on multi-GPU servers [16]. BeamNG.tech users report similar hardware demand for photoreal crash-dynamics studies.

Finally – community momentum and benchmarks

The open CARLA Autonomous-Driving Leaderboard offers yearly updated baselines for modular, imitation-learning, and RL planners, and the original CARLA paper

has received well over 7,500 Google-Scholar citations. For SVL, DRIVE Sim, or VTD, there is currently no comparable open, regularly updated benchmark available.

2.1.4 Rationale for selecting CARLA

Licence transparency and reproducibility

Redistribution of source code, scenarios, and recorded sensor logs is permitted under the MIT license, which is crucial for an academic thesis that needs to be completely reproducible [21].

Balanced sensor realism without proprietary lock-in

While independent studies like R-CARLA validate sub-pixel RGB and LiDAR accuracy in closed-loop experiments , migration to Unreal Engine 5 with Lumen and Nanite offers photoreal illumination [8]. This fidelity circumvents the vendor-specific tool chain of RTX-only engines while yet being enough for perception-aware planning [16].

Granular NPC configuration for situation-aware experiments

The Traffic Manager's actor-level parameters (speed factors, lane-change aggression, and traffic-light compliance) allow for controlled social environment shifts, which is essential to the adaptive cost-function that this study examines. Within a single simulation campaign, CARLA's Python/ROS 2 stack and the CommonRoad-CARLA bridge enable quick switching between A*, non-linear MPC, and reinforcement-learning planners [11].

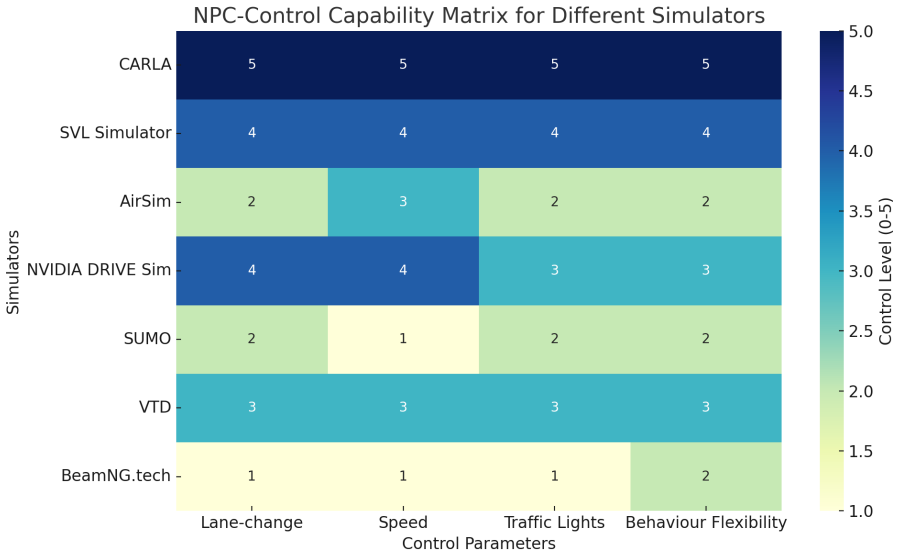


Figure 2.1: NPC-Control Capability Matrix For Different Simulators

Comprehensive planning interfaces

Within a single simulation campaign, CARLA's Python/ROS 2 stack and the CommonRoad-CARLA bridge enable quick switching between A*, non-linear MPC, and reinforcement-learning planners [12], [48].

Established benchmark ecosystem

With its pre-made scenarios, measures, and baseline outcomes, the CARLA Autonomous-Driving Leaderboard serves as an impartial benchmark for the innovative planner [21].

2.1.5 Discussion & Platform Choice – Decision-tree Diagram

To justify CARLA as the primary platform for this research, the decision-tree below outlines the key factors driving this choice. The diagram takes into account the simulator's ability to support end-to-end AV development, flexible NPC behaviour control, sensor fidelity, and computational scalability.

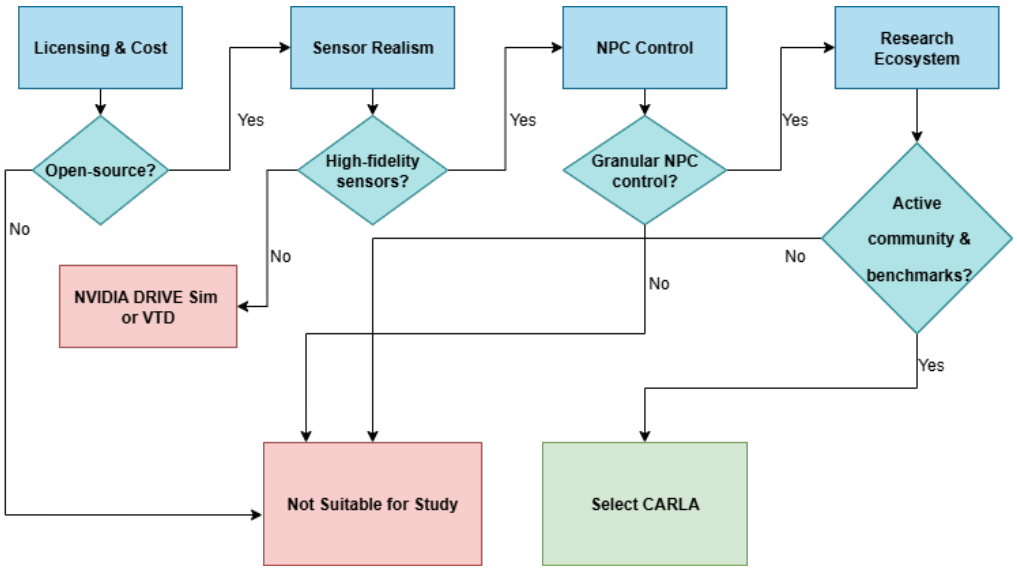


Figure 2.2: Decision-Tree Diagram for Selecting CARLA

Decision-tree explanation

- The first question in the decision-tree addresses licensing and cost: open-source platforms like CARLA, SVL, and AirSim provide the freedom necessary for academic work, while commercial options (like NVIDIA DRIVE Sim and VTD) come with cost and licensing restrictions.

- The next criterion evaluates sensor realism: CARLA's Unreal Engine 5 rendering, combined with verified multi-sensor models, offers a good balance of realism for path-planning experiments, whereas platforms like SUMO, with only 2-D models, or BeamNG.tech, which focuses on vehicle dynamics, fall short in sensor simulation.
- NPC control is a pivotal decision point for path-planning experiments that simulate real-world interactions. CARLA's flexible NPC system (Traffic Manager) gives it a substantial edge over other tools like SUMO or AirSim.
- Finally, the research ecosystem is an important factor: CARLA has an established benchmark system and a broad research community, ensuring reproducibility and validation of results.

For these reasons, CARLA is adopted as the reference simulation environment for all subsequent experiments on situation-aware, adaptive path planning. Given these factors, CARLA emerges as the most suitable, evidence-backed platform to investigate adaptive path planning in heterogeneous traffic, fulfilling both academic rigor and practical feasibility.

2.2 Path-Planning Foundations in Simulation

2.2.1 Layered Path Planning Stack

Autonomous vehicle path planning architectures are predominantly structured in a three-tier hierarchy: mission, behavior, and trajectory, each responsible for more detailed facets of decision making [74].

Mission (Global Route) Planning

At the top level, the mission planner computes a coarse path from the vehicle's current location to a goal, typically by performing graph search (e.g., Dijkstra's algorithm or A*) over a road network derived from digital maps [41]. This layer abstracts looks over dynamic obstacles and focuses purely on reachability and high-level routing (e.g., "take Highway I-80 to Exit 23, then turn onto Main Street").

Behaviour (Tactical) Planning

The behaviour planner converts the global route into a sequence of tactical maneuvers: lane changes, overtakes, yields at intersections, which are done using reasoning about traffic rules, predicted motions of other agents, and situational constraints [78], [74]. It effectively decides what to do next without yet specifying exact geometric paths. For example, 'change to the left lane to overtake slower traffic' or 'prepare to stop for a red light'.

Trajectory (Local) Planning

Finally, the trajectory planner generates a kinodynamically feasible path and velocity profile that implement the chosen manoeuvre, subject to vehicle dynamics, comfort constraints (e.g., limits on lateral acceleration and jerk) and collision avoidance [73], [74]. Methods at this layer range from sampling-based planners (RRT, RRT*) to optimization-based approaches such as nonlinear Model-Predictive Control (NMPC) or collocation, often augmented by spline smoothing to ensure continuity [72].

Figure 2.3 bellow depicts this layered structure, showing how each level refines the output of the previous one: a global route becomes a set of discrete actions, which in turn become continuous trajectories executable by the vehicle.

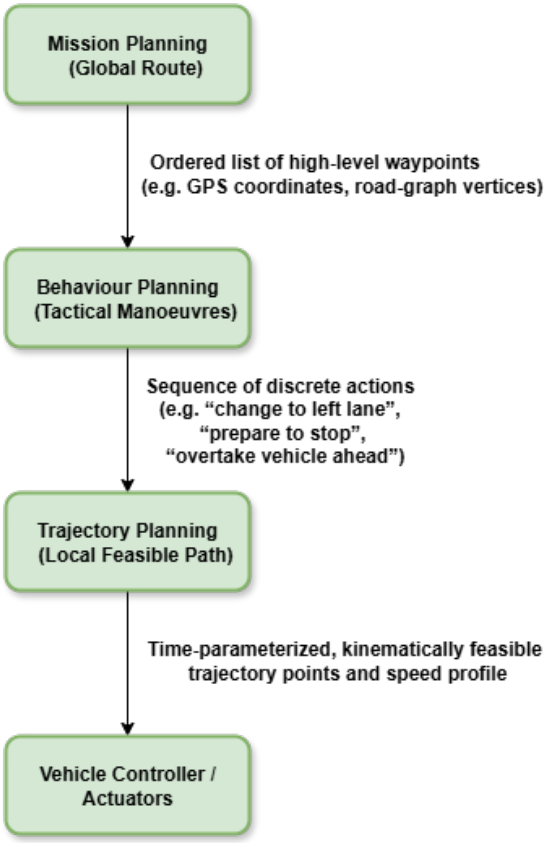


Figure 2.3: Layered planning architecture for autonomous vehicles

The mission planner (top) computes a global route as an ordered list of high-level waypoints over the road-network graph; the behaviour planner (middle) converts these waypoints into a sequence of tactical manoeuvres (e.g. lane changes, stops at signals); the trajectory planner (bottom) generates a time-parameterized, kinematically feasible path and velocity profile for low-level vehicle control.

2.2.2 Graph-Search Shortest-Path

At the mission-planning layer, the global route is typically obtained by applying a shortest-path algorithm over a directed, weighted graph derived from the road-network (e.g. an OpenDRIVE map). Two canonical methods are Dijkstra's algorithm and its heuristic-enhanced variant A*, which guarantee optimality under non-negative edge weights.

Dijkstra's Algorithm

Dijkstra's algorithm explores the graph from a source node s, maintaining a priority queue of tentative distances $d(v)$ to each node v At each step, it extracts the node u with minimum $d(u)$, then relaxes its outgoing edges (u, v) by

$$\text{if } d(u) + w(u, v) < d(v), \text{ then } d(v) \leftarrow d(u) + w(u, v)$$

where (u, v) is the non-negative cost of edge (u, v). Below is the pseudocode of the Dijkstra algorithm [77], [19].

Algorithm 1 Dijkstra's Shortest-Path Algorithm

Require: Graph $G = (V, E)$ with non-negative weights $w : E \to \mathbb{R}_{\geq 0}$, source vertex $s \in V$

Ensure: Arrays $d[v]$ (shortest-path distances) and $prev[v]$ (predecessors)

for all $v \in V$ **do** ▷ initialisation
    $d[v] \leftarrow \infty$
    $prev[v] \leftarrow$ null
end for

$d[s] \leftarrow 0$
$Q \leftarrow V$ ▷ min-priority queue keyed by d

while $Q \neq \emptyset$ **do**
    $u \leftarrow$ ExtractMin(Q)
    for all $v \in$ Adj$[u]$ **do**
        $alt \leftarrow d[u] + w(u, v)$
        if $alt < d[v]$ **then**
            $d[v] \leftarrow alt$
            $prev[v] \leftarrow u$
            DecreaseKey(Q, v, alt)
        end if
    end for
end while

return $(d,\ prev)$

The algorithm runs in $O(|E| + |V|log|V|)$ time when using a Fibonacci-heap or pairing-heap for the priority queue [77], [15].

A* Search Algorithm

A* augments Dijkstra by incorporating a heuristic function $h(v)$, an admissible estimate of the cost from v to the goal. The priority for node v becomes $f(v) = g(v) + h(v)$ where $g(v)$ is the cost from the source to v Only nodes for which

$$f(v) = g(v) + h(v)$$

is minimal are expanded first [76] [29]. Below is the pseudocode of the A* algorithm

Algorithm 2 A* Search Algorithm

Require: Graph $G = (V, E)$ with non-negative weights $w : E \to \mathbb{R}_{\geq 0}$; heuristic $h : V \to \mathbb{R}_{\geq 0}$; source $s \in V$; goal $t \in V$
Ensure: The shortest path π from s to t, or FAILURE

for all $v \in V$ **do** ▷ initialisation
  $g[v] \leftarrow \infty$; $f[v] \leftarrow \infty$; $prev[v] \leftarrow$ null
end for
$g[s] \leftarrow 0$
$f[s] \leftarrow h(s)$
$openSet \leftarrow \{s\}$ ▷ min-priority queue keyed by f
while $openSet \neq \emptyset$ **do**
  $u \leftarrow$ ExtractMin($openSet$)
  if $u = t$ **then** ▷ goal reached
    return reconstruct_path($prev$, t)
  end if
  for all $v \in$ Adj$[u]$ **do**
    $tent \leftarrow g[u] + w(u, v)$
    if $tent < g[v]$ **then**
      $prev[v] \leftarrow u$
      $g[v] \leftarrow tent$
      $f[v] \leftarrow g[v] + h(v)$
      if $v \notin openSet$ **then**
        Insert($openSet$, v)
      else
        DecreaseKey($openSet$, v, f[v])
      end if
    end if
  end for
end while
return FAILURE

When $h(v)$ is admissible (never over-estimates) and consistent (monotonic), A* is both complete and optimal. Its worst-case time complexity remains $O(b^d)$ or $O(|E| + |V|log|V|)$ depending on graph structure and heuristic quality [49], [76].

Worked Example on a CARLA OpenDRIVE Graph

Consider Town02 in CARLA, whose OpenDRIVE map yields a graph G where vertices represent road junctions and edges represent lane-segment centerlines weighted by segment length. To plan from a start waypoint W_s to a goal W_g:

1. Extract Graph:

Parse the OpenDRIVE XML to build $G = (V, E)$ assigning $\forall(u, v) \in E : w(u, v) = length(u \to v)$.

2. Run Dijkstra:

Initialize $d(W_s) = 0$ Extract nodes in increasing order of d. Upon termination, $prev[]$ encodes the shortest-path tree. Suppose the resulting route visits waypoints $(W_s, W_1, W_2, ...W_g)$.

3. Run A*:

Define $h(v)$ as the Euclidean distance from node v to W_g in 2-D map coordinates. A* typically visits fewer nodes, especially when obstacles (closed roads) are absent, reducing search time by up to 60% in open-map sections.

Complexity Discussion

Dijkstra is preferred when no reliable heuristic is available, guaranteeing $O(|E| + |V|log|V|)$. A* offers practical speedups in sparse-heuristic environments, but at the cost of managing both $g-$ and $f-$ values and storing visited nodes, leading to worst-case $O(b^d)$ memory use [76], [77].

2.2.3 Optimization-Based Motion Planning

Optimization-based planners formulate trajectory generation as an optimal control problem over a finite horizon, solving for a time-parameterized path and control inputs that minimize a cost function subject to dynamic and environmental constraints. These methods sit between classical graph search and learning-based approaches, offering smooth, dynamically feasible trajectories with explicit handling of vehicle models and obstacle avoidance [23].

Nonlinear Model-Predictive Control (NMPC)

At each control step, nonlinear MPC solves

$$\begin{aligned} \min_{u_0,\dots,u_{N-1},\, x_{1:N}} \quad & \sum_{k=0}^{N-1} \ell\big(x_k, u_k\big) \;+\; \ell_f(x_N) \\ \text{subject to} \quad & x_{k+1} = f\big(x_k, u_k\big), \qquad x_0 = x_{\text{current}}, \\ & u_k \in \mathcal{U}, \qquad\qquad x_k \in \mathcal{X}, \end{aligned} \tag{2.1}$$

where x is the state (e.g., position, heading, velocity), u the control input (steering angle, acceleration), $f(\cdot)$ the vehicle dynamics, and ℓ a stage cost penalising

deviation from a reference path and control effort. A common quadratic choice includes

$$\ell(x,u) \;=\; (x - x_{\text{ref}})^\top Q\,(x - x_{\text{ref}}) \;+\; u^\top R\,u,$$

with $Q, R \succeq 0$. A terminal cost ℓ_f ensures stability [47].

The key advantage is to have explicit satisfaction of state and control constraints, including obstacle avoidance and road boundaries. Built-in stability through terminal-cost design and theoretical convergence is guaranteed. Limitations are omputational complexity which grows non-linearly with horizon length N and state dimension, often limiting N to 20–30 steps. Performance is sensitive to model fidelity and solver initialization [37], [50].

Lattice-Based Motion Planning

Lattice planners discretize the continuous configuration space (x, y, θ) into a regular grid and employ a precomputed motion-primitive library of dynamically feasible, short trajectories generated offline under constant control inputs [54]. At runtime, planning reduces to a graph search (A* or Dijkstra) over these primitives, guaranteeing nonholonomic constraints by construction [2].

- **Motion Primitives:** Generated offline by simulating the vehicle model under constant control inputs; stored as short trajectories $(\Delta x,\ \Delta y,\ \Delta\theta)$.
- **Search:** At runtime, from a grid cell, apply each primitive to reach new cells, accumulating cost (e.g., path length + curvature penalty).

This method achieves real-time performance for highway and parking scenarios while respecting nonholonomic constraints.

Direct Transcription & Collocation

Direct transcription methods transcribe the optimal control problem into a large sparse NLP by sampling both states and controls at collocation points $\{t_0, \ldots, t_N\}$ and enforcing dynamics via collocation constraints (e.g., Hermite–Simpson) :

$$x(\tau_{k+1}) - x(\tau_k) - \frac{\Delta t}{6}\left(f_k + 4f_{k+1/2} + f_{k+1}\right) = 0.$$

The resulting NLP, with $O(N)$ variables, can be solved by interior-point methods (e.g., IPOPT) to high accuracy. While computationally heavier than NMPC, collocation yields smoother solutions and handles complex obstacle constraints naturally [5].

The advantage is high fidelity, accurately captures nonlinear dynamics and obstacle geometry.it has Smooth trajectories with continuity in derivatives. This has the Limitations such as it is computationally intensive: typical solve times of 0.5–2

s for $N \approx 50$. It is less suitable for real-time replanning without warm-starting or tailored solvers.

Gradient-Based Trajectory Optimization

Trajectory-optimization algorithms such as CHOMP [55] and TrajOpt [60] use gradient descent to optimize a collision-free trajectory over a discretized path. They minimize a cost comprising smoothness (e.g., acceleration, jerk) and obstacle proximity via signed-distance fields, iteratively updating the trajectory until convergence. These methods excel in high-dimensional spaces (e.g., manipulator arms) but have also been adapted for vehicle path smoothing in parking and off-road contexts.

This is effective in high-dimensional configuration spaces (e.g., manipulators). Also, it has flexible cost design, modular obstacle modeling [55], [60]. But, it is susceptible to local minima; requires good initialization. Real-time performance depends on cost complexity and collision-checking budget [55], [60].

Table 2.4: Complexity Comparison

Method	**Typical Solve Time**	**Memory**	**References**
NMPC	20–50 Hz cycles	$O(Nn^3)$	[23]
Lattice (A*/Dijkstra)	$<$ 10 ms per query	$O(P)$ primitives	[54]
Direct Transcription	0.5–2 s per solve	$O(N^3)$ NLP	[5]
CHOMP / TrajOpt	50–200 ms to converge	$O(K)$ states	[55], [60]

2.2.4 Sampling & Search-Based Planning

Sampling-based motion planners construct feasible trajectories by exploring the configuration space C through random sampling, rather than relying on explicit geometric decomposition. These methods' probabilistic completeness and capacity to function without full map parameterization have made them the "de facto standard" for high-dimensional planning problems. [43].

These algorithms are frequently separated into probabilistic-roadmap (PRM) approaches that pre-sample and link possible configurations and incremental tree-growth methods like the Rapidly-exploring Random Tree (RRT) family. [34] .Sampling-based planners generalize to continuous state spaces, which is crucial for autonomous cars operating under dynamic, non-convex constraints, whereas traditional graph-based search ensures optimality only on discretized lattices.

Probabilistic Roadmaps (PRM)

If there is a collision-free local path, the PRM algorithm connects each node to its closest neighbors, samples a set of n random configurations in free space, and saves these edges in a roadmap graph $G(V, E)$. Next, a Dijkstra or A* search is used

to extract a global route. With probabilistic completeness, the roadmap formally converges to the genuine connectivity of C_{free} as $n \to \infty$ [34].

2.2.5 Learning-Based & Hybrid Approaches

Traditional motion-planning methods depend predominantly on explicit mathematical models of the vehicle and its surroundings. These methods provide robust theoretical assurances of feasibility, optimality, and adherence to constraints. Yet purely model-based planners sometimes struggle to account for the complex interactions that occur in actual traffic scenarios, when numerous agents with unpredictable behaviour coexist on the roadway. Dynamic driving scenarios can entail imperfect observations, unpredictable actions from nearby vehicles, and fluctuating traffic conditions, hence complicating the formulation of exclusively rule-based planning solutions.

Recent research has investigated the incorporation of learning-based methodologies into the planning pipeline. These methodologies utilise data obtained from human driving, simulated environments, or prior autonomous driving trials to develop policies or cost functions that inform decision-making. Instead of calculating each trajectory only through first-principles optimisation or geometric search, learning-based planners try to encapsulate implicit driving behaviours using statistical models. Within the framework of a dynamic navigation function learning components can improve the ability of the system in a way that it has the ability to adapt the changing traffic patterns, anticipate the behavior of other road users, and generate candidate trajectories that reflect realistic driving strategies.

Learning-based planning does not necessarily replace classical algorithms. In many practical systems, learning components are combined with traditional planners to form hybrid architectures. In this method machine learning assists with prediction or decision making and deterministic planning algorithms maintain safety and constraint satisfaction.

Imitation Learning for Driving Policy Acquisition

Imitation learning is one of the earliest approaches used to introduce machine learning into autonomous driving systems. The core idea is to learn a driving policy by observing demonstrations from expert drivers. During training, the system receives a dataset of state–action pairs, where each state represents the sensory observation of the vehicle and each action corresponds to the control input chosen by the expert. A machine-learning model is then trained to approximate a policy function that predicts the appropriate action for a given state.

A significant study illustrated the feasibility of end-to-end imitation learning for driving through deep neural networks trained directly on camera images. This study involved training a convolutional neural network to directly translate raw visual input into steering commands, bypassing explicit modelling of intermediate perception

or planning steps [7]. The findings indicated that neural networks are capable of acquiring fundamental lane-following behaviour through human demonstrations.

Subsequent research expanded this notion by including advanced navigation directives into the learning process. Conditional imitation learning enables the driving policy to be influenced by navigational directives such as turn left, turn right, or proceed straight. This method allows a trained controller to adhere to a global trajectory while simultaneously responding to local environmental factors [14].

Despite its benefits, imitation learning includes certain limitations. The policy, being trained solely on expert demonstrations, may face scenarios upon deployment that were excluded from the training data. The phenomenon referred to as covariate shift may result in compounding errors when the vehicle diverges from the expert trajectory. Methods like Dataset Aggregation (DAgger) have been suggested to address this issue by systematically gathering corrective demonstrations and refining the policy through retraining [57].

Reinforcement Learning for Autonomous Driving

Reinforcement learning (RL) formulates autonomous driving as a sequential decision-making problem. The vehicle is modeled as an agent that interacts with an environment and learns actions that maximize cumulative reward. This interaction is commonly described using a Markov Decision Process (MDP) framework. An MDP consists of a state space S, an action space A, a transition function, and a reward function that evaluates each action taken by the agent [68].

In the context of autonomous driving, the state typically represents sensor observations such as camera images, LiDAR measurements, or processed environment features. The action space corresponds to control commands such as steering angle, throttle, and braking. Through repeated interaction with the environment, the agent gradually improves its policy in order to achieve higher cumulative rewards.

Simulation environments play a critical role in reinforcement learning research because large amounts of training data are required. Training such policies in the real world would be impractical and unsafe. For this reason, high-fidelity simulators such as CARLA are widely used to evaluate reinforcement learning approaches for autonomous driving [21].

Several studies have demonstrated that reinforcement learning agents can learn driving behaviors including lane following, obstacle avoidance, and intersection navigation when trained in simulation. For example, Kendall et al. presented an approach where reinforcement learning was applied to urban driving tasks using visual observations from onboard cameras [35]. The results showed that reinforcement learning policies can learn complex driving behaviors through repeated interactions with simulated environments.

Despite these promising results, reinforcement learning still faces several limitations. Training deep reinforcement learning policies requires large amounts of data and computational resources. In addition, purely learned policies may lack interpretability and safety guarantees, which is problematic for safety-critical applications

such as autonomous driving. For this reason, reinforcement learning is often used in combination with traditional planning algorithms that enforce physical and safety constraints.

Hybrid Planning Architectures

Many modern autonomous driving systems combine learning-based methods with classical motion planning algorithms. This design leads to hybrid planning architectures. In such systems, machine learning modules support perception, prediction, or decision making. Classical planners remain responsible for generating safe and dynamically feasible trajectories.

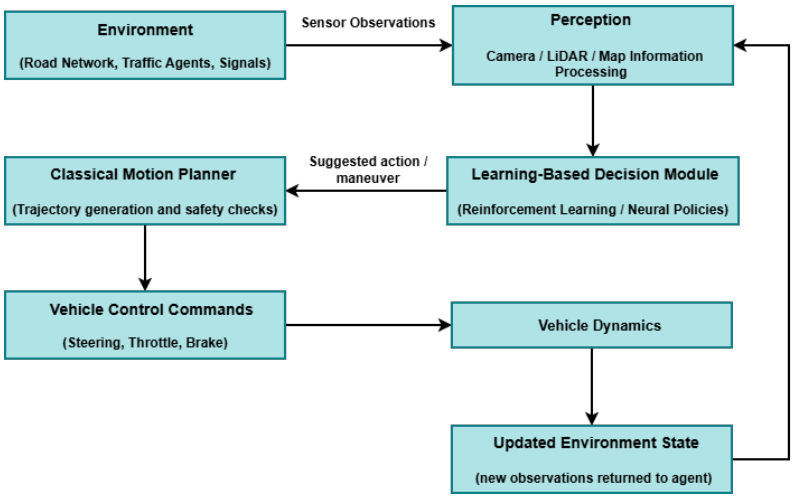


Figure 2.4: Conceptual architecture of a learning-assisted navigation system

Hybrid navigation systems often integrate learning-based decision modules with classical trajectory planners. As illustrated in Figure 2.4, environmental observations are processed by perception components, while learning modules provide high-level behavioral guidance. The motion planner then generates dynamically feasible trajectories that are executed through the vehicle control system [53].

One common hybrid strategy uses machine learning models to predict the future behavior of surrounding vehicles. These predicted trajectories are then incorporated into the planning process. The planner can adjust its trajectory in order to avoid potential conflicts with other traffic participants [53].

Behavior planning frameworks often rely on this combination of prediction and planning. For example, the behavioral planning architecture proposed by Wei et al. separates high-level decision making from low-level trajectory generation. The system first determines the appropriate driving maneuver. The trajectory planner then generates a feasible path that implements this maneuver while respecting vehicle constraints [74].

Hybrid approaches are particularly useful in dynamic traffic scenarios. Machine learning models can capture complex driving behaviors that are difficult to encode

using rule-based systems. At the same time, classical planning algorithms provide safety guarantees and enforce vehicle dynamics constraints. Simulation environments such as CARLA enable systematic evaluation of these hybrid architectures. The simulator provides configurable traffic agents, controllable weather conditions, and realistic sensor models [21].

2.3 Evaluation Metrics and Benchmarks for Autonomous Navigation

The evaluation of autonomous navigation systems requires clearly defined metrics that allow objective comparison between different planning approaches. In simulation-based research, these metrics are used to measure the performance, safety, and efficiency of navigation algorithms under controlled experimental conditions. Because autonomous driving involves interaction with complex environments and dynamic traffic participants, no single metric is sufficient to capture overall system performance. Instead, a combination of safety, efficiency, and computational performance indicators is typically used [53].

Simulation platforms such as CARLA provide structured benchmarking environments in which these metrics can be evaluated systematically. CARLA includes standardized scenarios, configurable traffic agents, and sensor models that allow researchers to analyze the behavior of planning algorithms under varying environmental conditions [21]. As a result, simulation-based benchmarks have become an important component of autonomous driving research. The following subsections describe the most commonly used metrics for evaluating navigation and planning algorithms.

2.3.1 Safety Metrics

Safety is the most critical requirement for autonomous navigation systems. Planning algorithms must ensure that the vehicle avoids collisions and respects traffic regulations. For this reason, safety metrics are typically the primary indicators used when evaluating navigation performance.

One of the most widely used metrics is the collision rate, which measures the number of collisions that occur during a navigation task. Collisions may involve other vehicles, pedestrians, or static objects such as roadside infrastructure. A lower collision rate indicates a safer navigation strategy [53].

Another important safety indicator is the traffic rule violation rate. This metric captures situations where the vehicle violates traffic regulations, such as running red lights or failing to yield at intersections. These violations are particularly relevant in urban driving scenarios where interactions with traffic signals and other road users are frequent.

Simulation platforms often provide additional safety-related indicators, including lane departures and unsafe distance violations. These metrics evaluate whether the

vehicle remains within its designated lane and maintains sufficient spacing from surrounding vehicles [21]. Together, these measurements provide a comprehensive view of the safety characteristics of a navigation algorithm.

2.3.2 Navigation Efficiency Metrics

Besides safety, autonomous navigation systems must also be assessed in terms of efficiency. A planner that avoids collisions but produces unnecessarily long or delayed routes is of limited practical value. For this reason, evaluation studies commonly include metrics that measure whether the vehicle reaches its goal in a timely and operationally effective manner. Such efficiency-oriented assessment is standard in motion-planning surveys and simulator-based benchmarking workflows [53].

One frequently used metric is the route completion rate. This metric measures how much of the assigned route is successfully completed by the autonomous agent. In CARLA-based benchmarking, route completion is a central evaluation quantity and is used together with infraction penalties to summarize overall task performance. The CARLA leaderboard framework explicitly reports route completion as one of its core metrics [71].

Another relevant indicator is travel time. This metric captures the total time required to reach the destination or to finish the assigned route. In navigation studies, travel time is important because it reflects whether the planner behaves efficiently under dynamic traffic conditions rather than merely conservatively. Closely related to this is path length, which indicates whether the planner selects unnecessarily long detours or inefficient maneuvers when alternative feasible routes exist. These criteria are consistent with the broader motion-planning literature, where efficiency is evaluated not only by feasibility but also by the quality of the produced route [53].

In more detailed analyses, efficiency is also linked to the smoothness and regularity of vehicle motion. Trajectories that involve excessive braking, abrupt acceleration, or frequent steering corrections may still complete the route, yet they indicate reduced operational quality. For this reason, some studies complement route completion and travel time with comfort-related measures such as acceleration smoothness or control stability. These quantities are especially relevant when comparing planners that differ in how often they replan or how aggressively they respond to dynamic obstacles [53].

When dynamic navigation is studied in CARLA, these efficiency metrics become particularly useful because the simulator enables repeated execution of the same route under controlled variations in traffic density, weather, and scenario difficulty. This makes it possible to compare not only whether a planner reaches the destination, but also how efficiently it does so across different runtime conditions [21].

2.3.3 Computational Performance Metrics

Autonomous navigation algorithms must operate under real-time constraints. In dynamic traffic scenarios, a planner has only a limited time window to update its decision before the surrounding environment changes again. For that reason, computational performance is not a secondary concern. It is a core evaluation dimension in autonomous driving research. Motion-planning surveys therefore treat runtime behavior and algorithmic tractability as essential criteria when comparing planning methods [53].

A central metric in this context is planning latency. Planning latency measures the time required to compute a route, trajectory, or control update after new environmental information becomes available. In dynamic navigation, low planning latency is important because route adjustments must be produced quickly enough to remain valid under changing traffic conditions. A method that yields high-quality solutions but cannot respond within the available reaction time may be unsuitable for online use [53].

A related measure is replanning frequency. Dynamic navigation functions often need to update the planned route repeatedly in response to blocked lanes, traffic participants, or traffic-light states. A system that supports frequent replanning can adapt more effectively to runtime changes. At the same time, frequent replanning increases the computational burden. This creates a trade-off between responsiveness and processing cost, which must be analyzed carefully in simulation studies [53], [21].

Another important aspect is scalability. A planner may perform well in sparse environments but degrade significantly as the number of vehicles, constraints, or route alternatives increases. Scalability analysis therefore examines how runtime and memory demand grow with scenario complexity. This issue is particularly relevant in simulation platforms such as CARLA because traffic density, map size, and sensor configuration can be varied systematically across repeated experiments [21], [64].

Benchmarking frameworks also support the systematic recording of runtime-related outputs. CARLA provides benchmark tooling and leaderboard evaluation protocols that make it possible to compare autonomous agents under standardized conditions. Although leaderboard scores primarily summarize driving performance, the benchmark structure itself is valuable for studying whether an agent can sustain stable decision making across multiple routes and scenario types [44]. CARLA's benchmark documentation also includes performance-oriented summaries of agent behavior during evaluation runs [20].

For a thesis on dynamic navigation, these computational metrics are especially relevant because the contribution is not limited to generating a valid route. The system must also update that route when the environment changes. Therefore, the evaluation should not only ask whether the planner succeeds, but also whether it can respond fast enough and robustly enough to support online navigation in a realistic simulator environment [53], [21].

Figure 2.5 represents the pipeline for autonomous navigation algorithms in simulation environments. Navigation planners are executed in the CARLA simulator

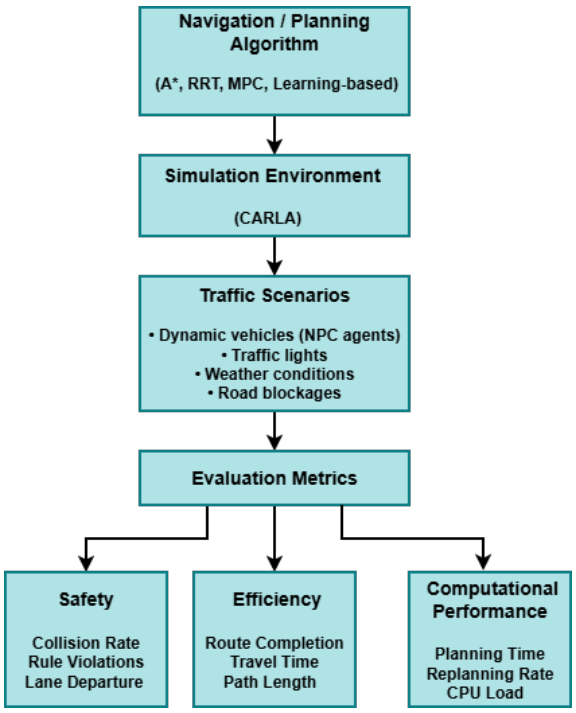


Figure 2.5: Evaluation pipeline for autonomous navigation algorithms in simulation environments

under configurable traffic scenarios including dynamic agents, traffic signals, and environmental conditions. System performance is evaluated using safety, efficiency, and computational performance metrics [53], [21].

2.4 NPC and Traffic Control Mechanisms in Simulation

Autonomous navigation systems operate in complex environments where multiple traffic participants interact simultaneously. In real-world driving scenarios, vehicles must continuously adjust their behavior in response to surrounding traffic, road infrastructure, and traffic regulations. For this reason, the evaluation of autonomous navigation algorithms requires simulation environments that can realistically represent the behavior of other road users. Within simulation platforms, these simulated traffic participants are commonly referred to as non-player characters (NPCs) or traffic agents [21].

Unlike static obstacles, traffic agents exhibit dynamic behavior that depends on road geometry, traffic rules, and interactions with nearby vehicles. These interactions may include lane changes, yielding behavior, car-following dynamics, or responses to traffic signals. Modeling such interactions is an important aspect of traffic

simulation and has been studied extensively in transportation research. Microscopic traffic simulators such as SUMO incorporate behavioral models that describe how individual vehicles react to their surroundings and to other road users [40], [4].

For autonomous driving research, the behavior of surrounding traffic agents directly affects the difficulty of the navigation problem. Planning algorithms must generate trajectories that remain safe while interacting with other vehicles that may behave unpredictably. This challenge has been recognized in the motion planning literature, where interaction with dynamic agents is considered a key component of real-world autonomous navigation [53].

Modern simulation platforms provide several mechanisms to generate and control traffic behavior. Built-in traffic management systems allow simulators to coordinate large numbers of vehicles while enforcing traffic rules such as speed limits, lane usage, and traffic signal compliance. In addition, scenario description frameworks enable researchers to design structured traffic situations that reproduce specific driving conditions [36]. These tools allow navigation algorithms to be evaluated under controlled yet realistic traffic environments.

The behavior of simulated traffic agents is particularly important when evaluating dynamic navigation strategies. In scenarios involving congestion, road blockages, or intersections with multiple vehicles, navigation algorithms may need to update their route or trajectory in response to evolving traffic conditions. Consequently, the mechanisms used to generate and control NPC behavior influence both the difficulty of navigation tasks and the validity of simulation-based evaluations [36].

This section reviews the main mechanisms used in autonomous driving simulators to model traffic participants and regulate their behavior. First, built-in traffic control systems provided by major simulation platforms are discussed. Next, scenario description frameworks that enable the creation of structured traffic situations are introduced. Finally, behavioral interaction models that describe negotiation between vehicles are examined, together with their impact on navigation algorithm evaluation.

2.4.1 Built-in Traffic Control Systems in Simulation Environments

Modern autonomous driving simulators have built-in mechanisms for creating and managing traffic behavior. These simulators have been created to manage a high number of vehicles yet adhering to road rules and creating realistic driving patterns. Built-in traffic control mechanisms enable realistic interactions between multiple agents, allowing researchers to test their navigation solutions under realistic traffic scenarios.

Microscopic traffic simulation models simulate each vehicle individually using driver models.These are common method for controlling traffic. These driver models determine how vehicles behave, how they accelerate, decelerate, and follow other vehicles. This type of microscopic traffic simulation is a well-studied area in transportation engineering, and many simulation tools are based on this idea [36].

CARLA Traffic Manager

The CARLA simulator includes a built-in traffic control module known as the Traffic Manager (TM). This module is responsible for coordinating the behavior of non-player vehicles within the simulation. The Traffic Manager assign control policies to individual vehicles including lane following, speed regulation, and collision avoidance.

A combination of waypoint-based navigation and rule-based decision making is used by the system. Vehicles follow predefined road waypoints extracted from the map. The local decisions such as lane changes or braking are handled through heuristic rules. This method allows the Traffic Manager to simulate realistic traffic flow without complex global coordination [21].

Another feature of the CARLA Traffic Manager is the ability to handle large numbers of vehicles in real time. This feature uses centralized coordination combined with local decision modules. The Traffic Manager also supports configurable parameters, for instance desired speed, following distance, and lane change probability, that allows researchers to create different traffic densities and driving behaviors [11].

From the perspective of navigation algorithm evaluation, the Traffic Manager provides a controlled as well as dynamic environment. The behavior of surrounding vehicles is deterministic to a certain extent, but still has variability due to interactions between agents. This makes it suitable for testing navigation strategies that must adapt to traffic flow and avoid collisions in real time [11].

SUMO and TraCI Interface

The Simulation of Urban Mobility (SUMO) is a widely used open-source traffic simulator, mainly focuses on large-scale microscopic traffic modeling. SUMO simulates individual vehicles and uses well-established car-following and lane-changing mod-

els. It reproduces realistic traffic behavior. These models are mainly based on traffic flow theory and empirical observations of driver behavior [40], [4].

An important feature of SUMO is the Traffic Control Interface (TraCI). It enables external programs to interact with the simulation in real time. Using TraCI, it is possible to control individual vehicles, modify traffic signals, and retrieve simulation data during the runtime. This interface is particularly useful as well for integrating SUMO with other simulation environments [40], [4].

In autonomous driving research, SUMO is often attached with high-fidelity simulators such as CARLA. In such simulation setups, SUMO handles traffic flow and vehicle behavior and CARLA provides detailed sensor simulation and physics modeling. This separation helps researchers to combine a large scale traffic simulation with realistic perception and control environments.

The use of TraCI also used for more flexible experimental setups compared to built-in traffic managers. Researchers can implement custom traffic control strategies, modify vehicle behavior dynamically, as well as create complex interaction scenarios. For this reasons, SUMO and TraCI are widely used for studying traffic-level effects on autonomous navigation algorithms.

Virtual Test Drive (VTD)

Virtual Test Drive (VTD) is a commercial simulation platform that is widely used in the automotive industry. VTD is used for testing advanced driver assistance systems and autonomous driving functions. It provides a dynamic simulation environment with traffic simulation, sensor modeling, and scenario generation [27].

VTD notably integrates traffic simulation directly into its architecture by using advanced driver behavior models. These models can simulate realistic interactions between vehicles with cooperative and non-cooperative behaviors. VTD also supports standardized scenario descriptions and enables precise control over traffic situations [51].

A notable feature of VTD is its focus on industry-grade validation. The platform is designed to support large-scale testing and certification processes. It includes interfaces for integrating external control systems as well as supports real-time simulation requirements. For this reason, VTD is often used in industrial settings where reproducibility and validation are critical [51].

From a research perspective, VTD illustrates the integration of traffic control systems inside a simulation framework. Although it is less accessible than open-source tools, it provides insights into how commercial simulators handle complex traffic scenarios and large-scale validation tasks.

Comparative Perspective

The discussed systems illustrate different approaches to traffic control in simulation environments. CARLA's Traffic Manager focuses on flexibility of use and integration within a dynamic simulator. SUMO provides detailed traffic flow modeling as well as

flexible external control through TraCI. VTD offers a a roboust industrial solution with integrated traffic and scenario management.

These differences have direct impact for navigation algorithm evaluation. Built-in traffic managers provide convenience and reproducibility. On the other hand, external traffic simulators offer greater flexibility and scalability. For this reason, the choice of traffic control system depends on the specific requirements of the research problem, including the level of realism, control, and computational efficiency required.

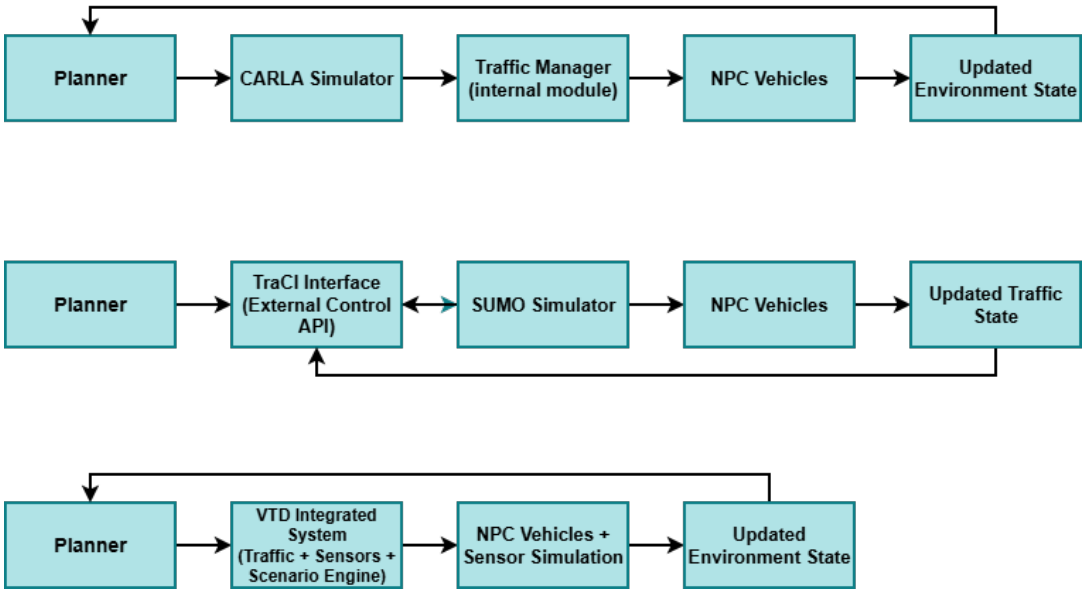


Figure 2.6: The architectural differences between traffic control mechanisms in CARLA, SUMO, and VTD

Comparison of traffic control architectures in simulation platforms are illustrated in Figure 2.6. In CARLA, traffic behavior is managed internally through the Traffic Manager, enabling direct interaction between the planner and simulated agents. In contrast, SUMO relies on an external control interface (TraCI), which facilitates bidirectional communication between the planner and the traffic simulator. VTD provides a fully integrated framework combining traffic simulation, sensor modeling, and scenario execution. These architectural differences determine how dynamic traffic interactions are generated and how navigation systems receive updated environment information.

2.4.2 Scenario Description Frameworks and Scripted Traffic Generation

In addition to built-in traffic control mechanisms, modern simulation environments provide structured frameworks that define reproducible and controllable traffic situations. These frameworks are important for evaluating navigation systems under well-defined and repeatable conditions. This is particularly important for dynamic navigation functions that must respond to changing environments. Scenario-based approaches allow researchers to explicitly specify interactions such as vehicle merging, intersection conflicts, or road blockages unlike purely stochastic traffic generation. As a result, they enable systematic testing of planner robustness and adaptability under controlled variations of traffic behavior [25].

OpenSCENARIO is a widely adopted standard for scenario description. It is developed by the Association for Standardisation of Automation and Measuring Systems (ASAM). OpenSCENARIO usually provides a formal language for defining dynamic elements within a simulation, which includes vehicle trajectories, triggers, and interaction conditions. It separates scenario logic into declarative components, for instance, actions, events, and conditions, that allows complex behaviors to be constructed in a modular and reusable manner. This well structured representation supports interoperability across different simulators, that includes CARLA and VTD. This enabling consistent benchmarking of navigation algorithms across platforms [22].

From the perspective of navigation planning, scenario frameworks play a crucial role in generating trigger-based environmental changes. For example, a scenario may define that a leading vehicle suddenly decelerates when the ego vehicle reaches a certain distance, or that a pedestrian enters the street at a predefined time. These events introduce environmental changes that require immediate replanning. These mechanisms are particularly important for evaluating dynamic navigation functions because they create situations where static path planning is not sufficient and adaptive behavior becomes necessary [21].

Scenario-based simulation is also closely related to the concept of corner case generation. It focuses on rare but safety-critical situations. Random traffic simulation alone is not sufficient most of the times to reliably produce such events, because their probability of occurrence may be extremely low. Scenario frameworks address this limitation by explicitly encoding these situations, ensuring that navigation systems are exposed to challenging conditions such as near-collision events, occlusions, or unexpected agent behavior. Fremont et al. introduced the SCENIC language, that enables probabilistic scenario specification. It allows systematic exploration of complex driving situations . This approach bridges the gap between deterministic scripting and stochastic simulation as well as provides a flexible tool for testing planner performance under uncertainty [24].

A further important element of scenario frameworks is their integration with traffic simulation engines. In CARLA, scenarios can be executed through the ScenarioRunner module. It interprets OpenSCENARIO definitions and coordinates agent

behavior accordingly. This enables the simultaneous control of multiple actors, including vehicles, pedestrians, and traffic lights, and that all within a unified execution environment. In SUMO-based workflows, scenario definitions can be combined with route files and traffic demand models to create structured experiments. These integrations makes sure that scenario definitions descriptive as well as executable within the simulation loop [46].

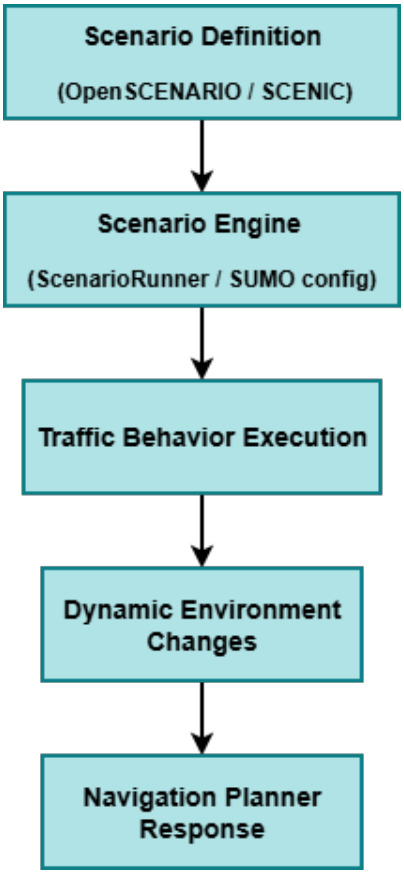


Figure 2.7: Role of scenario description frameworks in simulation-based evaluation

The scenario description frameworks introduce an additional layer between the planner and the simulation environment. Traffic managers govern low-level agent behavior and scenario engines define high-level temporal and spatial constraints that shape the evolution of the environment. This distinction is important for dynamic navigation, because it highlights that environmental changes may originate from both continuous traffic interactions as well as discrete scenario triggers. That is why a robust navigation function must be capable of responding to both types of dynamics.

Finally, the use of standardized scenario descriptions contributes to the reproducibility and comparability of experimental results. Researchers can ensure that navigation algorithms are evaluated under identical conditions by encoding traffic situations in a formal and simulator-independent manner. This is particularly rel-

evant for benchmarking and validation along with differences in scenario execution can significantly affect performance outcomes. As a result, scenario frameworks have become a central component in modern autonomous driving research. It supporting both controlled experimentation and large-scale evaluation campaigns.

The role of scenario description frameworks in generating dynamic traffic conditions is illustrated in Figure 2.7. Scenario definitions specify structured events, such as vehicle interactions or environmental triggers, which are executed by a scenario engine within the simulation environment. These events lead to dynamic changes in traffic behavior, including sudden braking, lane changes, or obstacle appearance. As a result, the navigation planner must continuously adapt its trajectory in response to updated environmental conditions. This process forms a closed-loop interaction between the planner and the simulation, which is fundamental for evaluating dynamic navigation functions.

2.4.3 Interaction Between Planner and Traffic Agents: Social and Negotiation Models

Navigation is not solely determined by static road structures or predefined routes in dynamic traffic environments. It is also determined by the interaction between the ego vehicle and surrounding agents. These interactions produces an additional layer of complexity, because each agent operates according to its own objectives, constraints, as well as behavioral patterns. As a result, modern navigation systems must adapt multi-agent dynamics. Meaning the actions of one vehicle influence and are influenced by others. This perspective shifts navigation planning to a decision-making process from a purely geometric problem. This is particularly relevant in urban driving scenarios [61].

A fundamental difficulty within these environments is that traffic participants demonstrate implicit social behaviors. For example yielding, merging, or right-of-way negotiation. These actions are not directly encoded in road networks but they arise from established conventions and driver expectations. Various approaches are proposed to model these interactions incorporating elements of game theory. agents are treated as rational decision-maker who seeks to optimize its own objective, considering the actions of others. Schwarting et al. describe autonomous driving as a multi-agent decision-making problem. Here planning must anticipate the responses of other agents to ensure safe and efficient navigation [61].

Interactive motion prediction constitutes a specific methodological approach, wherein the future paths of nearby agents are forecasted, incorporating the effects of their interactions. Interactive approaches consider that the behavior of one agent may change in response to the planned actions of another which is different than the independent prediction models. For example, a vehicle approaching an intersection may slow down if it scence that another vehicle intends to cross first. Using recurrent neural networks, Deo and Trivedi propose a framework for multi agent trajectory prediction. It capture interaction-aware behavior and demonstrates improved pre-

diction accuracy in complex traffic scenarios [18].

Another important direction is the use of game-theoretic planning models, where interactions are explicitly formulated as strategic games. In these models, each agent selects actions based on expected outcomes and potential responses from others. This framework allows the planner to reason about cooperative and competitive behaviors, such as merging into traffic or negotiating lane changes. Sadigh et al. introduced a planning approach that models interactions as a game between the autonomous vehicle and human drivers, enabling the system to influence the behavior of others through its own actions [58]. Such approaches highlight that navigation is not only reactive but can also be interactive and anticipatory.

In simulation environments, interaction models are usually approximated using either rule-based or probabilistic behavior models. For example, traffic managers in simulators like CARLA implement simplified interaction rules as collision avoidance, car-following, and lane-changing behaviors. These models do not capture the full complexity of human decision-making. instead they provide a reasonable approximation of traffic flow. It also enable the evaluation of navigation strategies under multi-agent conditions [21]. Similarly, microscopic traffic simulators such as SUMO employ car-following and lane-changing models in a way that the Intelligent Driver Model (IDM) and MOBIL, to simulate interactions between vehicles [40].

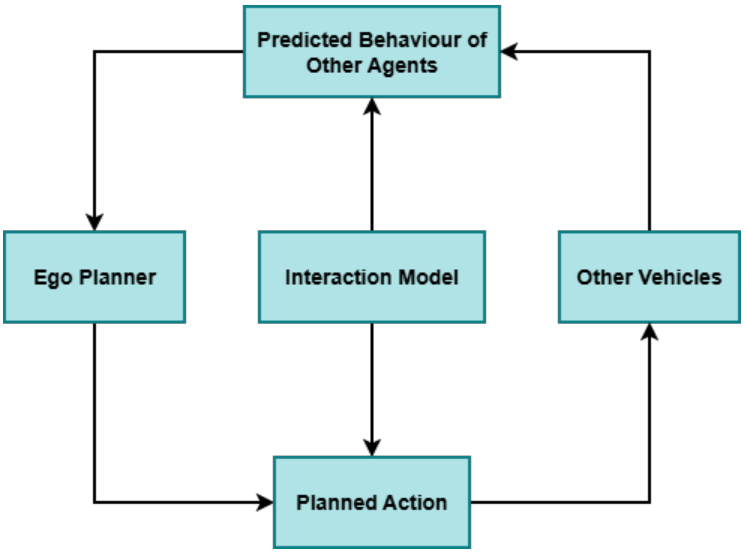


Figure 2.8: Conceptual representation of interaction-aware navigation in multi-agent environments

Figure 2.8 illustrates the interaction between the ego vehicle and surrounding traffic agents in a multi-agent environment. The planner estimates the future behavior of other vehicles and generates actions accordingly, while these actions in turn influence the behavior of surrounding agents. The interaction model captures this bidirectional dependency, forming a coupled decision-making loop.

From the perspective of dynamic navigation functions, the interaction between

the planner and surrounding agents introduces uncertainty and variability into the environment. Even if the global route remains fixed, local traffic interactions can influence the feasibility and efficiency of the chosen path. For example, congestion, aggressive driving behavior, or unexpected maneuvers by other agents may alter the effective cost of a route. For this reason, navigation systems must be evaluated in terms of geometric optimality aswell as in their ability to operate robustly in the presence of interacting agents.

Furthermore, the integration of social and negotiation models into simulation-based evaluation allows for more realistic testing of navigation strategies. Scenario frameworks can be used to create situations where interaction is critical, such as merging lanes, unprotected left turns, or dense urban traffic. In such cases, the planner must operate within an environment shaped by both deterministic rules and emergent behaviors. This reinforces the importance of considering interaction dynamics as part of the overall navigation function, even when the planning process itself is performed at the route level.

2.4.4 Impact on Planner Evaluation

The presence of dynamic traffic agents and complex interaction patterns significantly influences how autonomous navigation systems are evaluated in simulation environments. Classical path planning algorithms are often assessed based on geometric optimality or computational efficiency. This criteria are insufficient when planners operate in environments that is populated with multiple interacting agents. In these scenarios evaluation must consider safety, adaptability, and robustness under dynamic and uncertain conditions.

A special impact of traffic control system lies in the introduction of non deterministic behaviors which challenges the reproducibility and consistency of planner performance. Systems like the CARLA Traffic Manager or externally controlled simulators like SUMO generates a traffic patterns that depend on stochastic elements. That includes randomized vehicle spawning, behavioral variability, and interaction outcomes. For this, identical planning algorithms may exhibit different performance across simulation runs, necessitating evaluation over multiple scenarios and statistical aggregation of results rather than single-run analysis [21], [40].

An important aspect is the interaction-induced complexity that arises when planners must account for the anticipated behavior of other agents. In these cases, evaluation metrics extend beyond simple trajectory tracking or success of goal reaching. Instead, planners are assessed based on their ability to handle negotiation scenarios. It includes merging, yielding, or navigating through intersections with competing traffic flows. This requires the incorporation of metrics. These capture safety and social compliance, including collision rates, rule adherence, and interaction smoothness [58], [61].

In dynamic environments, scenario-based evaluation has become a standard methodology in order to assess planner performance. Instead of relying solely on continuous simulation runs, planners are tested against predefined scenarios. These represent

critical or challenging situations like sudden obstacles, dense traffic, or adversarial behaviors. Frameworks such as OpenSCENARIO and related benchmarking initiatives enable systematic comparison of planning strategies. this is done by controlling initial conditions and traffic configurations. This approach enhances reproducibility, still capturing the variability inherent in multi-agent interactions [22], [25].

Beyond safety and interaction metrics, computational performance under dynamic conditions becomes an important factor. Planners that are operating in environments with active traffic agents must frequently update their decisions in response to changing conditions. This introduces additional computational load because there are repeated collision checking, trajectory replanning, and prediction of other agents' behavior. As aresult, evaluation must consider solution qualityas well as runtime efficiency. That includes planning latency and the ability to meet real-time constraints [53].

the importance of robustness and generalization in planner evaluation is highlighted by the integration of traffic control mechanisms. A navigation system that performs well in static or low-density environments may fail when it is exposed to complex traffic interactions. That is why comprehensive evaluation requires testing across a diverse set of traffic densities, with behavioral profiles, and environmental conditions. This represents recent benchmarking. That emphasize large-scale, scenario-diverse evaluation protocols for autonomous driving systems [9].

In summary, the inclusion of traffic control mechanisms transforms planner evaluation from a purely algorithmic assessment into a system-level analysis that considers interaction dynamics, environmental variability, and computational constraints. This shift is essential for accurately assessing navigation systems intended for real-world deployment and directly supports the development of dynamic navigation functions capable of adapting to changing traffic conditions.

2.5 Simulation Fidelity vs Real-Time Performance

Simulation environments for autonomous driving serve as an Important platform for the development as well as evaluation of navigation algorithms. Still, these environments have a trade-off between simulation fidelity as well as real-time computational performance. High-fidelity simulations are able to accurately reproduce physical dynamics, sensor characteristics, and complex environmental interactions. On the other hand real-time performance needs efficient computation to enable responsive decision making and scalable experimentation. In order to balance these competing objectives there is a central challenge in the design and use of simulation frameworks for autonomous navigation.

From the perspective of dynamic navigation functions, this limitation directly affects the ability of a planner to react to changing conditions. High-fidelity simulations provide more accurate representation of real world driving scenarios. These features are particularly important when evaluating interaction aware planning. Here subtle differences in perception or motion prediction can significantly affect decision making outcomes. However, increased realism typically leads to higher computational costs that may limit the frequency of replanning or the number of agents that can be simulated simultaneously [56], [21].

On the other hand simplified simulation models give priority to computational efficiency. This enables faster execution than real time and large scale scenario testing. Traffic simulators such as SUMO uses abstracted vehicle dynamics and rule-based behavior models. It allows thousands of agents to be simulated efficiently. they may not capture fine-grained interactions which is required for validating motion level planning strategies or sensor-driven navigation systems. Even i they have such abstractions facilitate large-scale evaluation and reproducibility [40].

The interaction between fidelity and performance also affects the evaluation of replanning strategies in dynamic environments. Navigation systems that rely on frequent updates must operate within strict computational budgets. Specially when interacting with multiple agents or processing high dimensional sensor data. If simulation fidelity imposes excessive computational overhead, the planner may be unable to perform timely updates. As a result limiting its effectiveness in dynamic scenarios. This highlights the importance of considering both simulation realism and computational efficiency when navigation performance is assessed.

Modern simulation platforms like CARLA attempt to balance these requirements by offering configurable levels of detail in physics. Along with rendering as well as sensor modeling. This flexibility allows researchers to tailor the simulation environment according to the needs of specific experiments. It ranges from high-fidelity perception studies to large-scale traffic simulations. This trade-off remains a defining characteristic of simulation based research and must be explicitly considered when interpreting experimental results [21].

In the following subsections, this trade-off is examined in more detail. The main focus will be on physical realism, stochastic behavior, computational bottlenecks, and scalability limitations. These aspects are particularly relevant for dynamic

navigation functions. Here the ability to update routes and adapt to environmental changes depends on algorithmic design. It depend also on the capabilities and constraints of the simulation environment.

2.5.1 Physics and Rendering Trade-offs

Simulation fidelity in autonomous driving environments is largely determined by two core components, namely physical modeling and visual rendering. Both aspects contribute to the realism of the simulated environment. At the same time, they introduce significant computational demands. For this reason, simulation platforms must balance accuracy with efficiency, especially when used for evaluating dynamic navigation functions.

Physical modeling mainly points to the representation of vehicle dynamics, collision behavior, as well as environmental interactions. High fidelity simulators typically merge detailed dynamic models that responsible for forces like tire friction, inertia, and vehicle kinematics. These models allow for more realistic trajectory execution and it enable precise evaluation of motion planning algorithms. In particular, when planners operate under dynamic conditions, accurate physical responses are really essential. Small deviations in acceleration or steering behavior can lead to different interaction outcomes, especially in dense traffic scenarios [53].

However, increasing physical realism also increases computational cost. Complex dynamic models needs smaller simulation time steps and more frequent updates. This directly affects the ability of the system to perform real time replanning. In scenarios where navigation decisions must be updated repeatedly, such as in the presence of moving obstacles or traffic agents, computational overhead can limit responsiveness. Consequently, many simulation frameworks offer simplified kinematic models as an alternative. These models reduce computational load but may omit important physical effects, which can impact the validity of evaluation results [21].).

Rendering plays a similarly important role, specially for perception driven navigation systems. High quality rendering enables realistic simulation of camera sensors. That includes lighting conditions, shadows, reflections, and weather effects. This is essential for evaluating perception modules that rely on visual input. In platforms such as CARLA, rendering is tightly integrated with sensor simulation. It allows the generation of photorealistic images and depth information. These capabilities support the testing of end-to-end learning systems as well as vision based navigation approaches [21].

Despite having some benifits, high-fidelity rendering introduces significant GPU requirements. Realistic image generation needs complex shading models and large scene representations. Rendering can become a major bottleneck if it is combined with multiple sensors or high-resolution outputs. This is particularly relevant in multi agent simulations, where vehicles are equipped with sensors simultaneously. In those cases, rendering performance can limit the number of agents or the frequency of simulation updates [56].

To address these limitation, configurable levels of detail is provided to modern simulation platforms . Users can adjust parameters like physics accuracy, sensor resolution, and rendering quality. These depends on the requirements of the experiment. For example, large-scale traffic simulations has probablity to reduce rendering complexity or disable specific sensors to improve performance. On the other hand, perception focused studies may prioritize visual realism at the cost of simulation speed. This flexibility is really essential for supporting diverse research objectives within a single simulation framework.

In the study of dynamic navigation functions, the trade-off between physics and rendering has direct implications for system behavior. Accurate physics ensures that planned trajectories are executed reliably, having the realistic rendering supports robust perception and environment understanding. However, both aspects compete for computational resources. Therefore, the effectiveness of a navigation system in simulation depends not only on the planning algorithm but also on how these trade-offs are managed.

2.5.2 Determinism and Stochasticity in Simulation

Simulation environments for autonomous driving represents different levels of determinism and stochasticity that directly affect the evaluation of navigation systems. Deterministic simulations produce identical outcomes when executed under the same initial conditions. On the other hand stochastic simulations introduce variability through randomized elements. For example vehicle behavior, traffic generation, or sensor noise. Both approaches are widely used, and each serves a distinct purpose in the assessment of navigation performance.

Deterministic simulation is important for reproducibility. The same scenario can be executed repeatedly with identical outcomes when all parameters are fixed. This method allows direct comparison between different planning strategies under controlled conditions. It enables precise debugging as well as analysis of failure cases. Many simulation frameworks provide mechanisms to enforce determinism. It is done by fixing random seeds or controlling execution order. For example, CARLA allows reproducible experiments by synchronizing simulation steps and controlling randomness within the environment [21].

The variability observed in real world traffic is not capured by Fully deterministic environments. Human driving behavior is uncertain as well as environmental factors such as perception noise or unexpected agent actions introduce unpredictability. Stochastic simulation models incorporate randomness into agent behavior and environment dynamics. Traffic simulators such as SUMO use probabilistic models for vehicle insertion, route selection, and lane-changing decisions. That results in diverse traffic patterns across different simulation runs [40].

Including stochasticity is essential for evaluating the robustness and generalization of navigation systems. A planner that performs well in a single deterministic scenario may have failure when it is exposed to variations in traffic conditions. Stochastic simulation enables testing across a distribution of scenarios rather than a single

fixed configuration. This aligns with recent approaches in scenario based validation. Here, multiple variations of a base scenario are generated in order to assess system reliability that is under uncertainty [25].

Stochastic simulation introduces difficulties for evaluation. Across runs Performance metrics may vary which makes it difficult to draw conclusions from individual experiments. For this reason, evaluation must be conducted over multiple repetitions. Results must be aggregated using statistical measures. For example, mean performance and variance. This requirement increases computational cost and complicates experimental design. Moreover, differences in random sampling can lead to variability that is not directly related to the planning algorithm itself.

The balance between determinism and stochasticity is particularly relevant from the perspective of dynamic navigation functions. Deterministic scenarios allow for controlled analysis of how a planner responds to specific environmental configurations. Such as blocked roads or predefined traffic conditions. On the other hand, stochastic scenarios expose the planner to a wider range of interactions and unexpected events. These better reflect real world conditions. Both perspectives are necessary to obtain a comprehensive evaluation of navigation performance.

Recent research shows the importance of combining deterministic and stochastic approaches within a unified evaluation framework. Scenario definitions can be used to specify the overall structure of a situation. Stochastic variations are introduced to generate diverse realizations of that scenario. This hybrid approach ensures both reproducibility and variability. It enables systematic yet realistic evaluation of navigation systems [24].

In summary, determinism supports controlled experimentation and reproducibility. On the other hand, stochasticity enables robustness testing and realistic variability. Effective evaluation of navigation systems requires a careful balance between these two aspects. This balance is especially important in simulation-based studies where the behavior of traffic agents and environmental conditions directly influences the observed performance of the planner.

2.5.3 Computational Bottlenecks and System Constraints

The practical use of simulation environments for autonomous navigation is determined by modeling fidelity as well as by the available computational resources and system-level constraints. With the increment of simulation complexity computational bottlenecks also increase and that can directly affect the execution and evaluation of navigation algorithms. These challenges are particularly relevant in dynamic environment. Because the planner must process changing conditions and produce updated decisions within limited time budgets.

The simulation loop itself is one of the primary sources of computational load. It includes physics updates, traffic agent behavior, and environment synchronization. In platforms such as CARLA, the simulation operates in discrete time steps. Each step involves updating vehicle states, resolving collisions, and executing control commands. With the increment of the number of agents, the computational cost

grows significantl. Because, each additional vehicle introduces new interactions and state updates. This scaling effect has been observed in both game engine based simulators and traffic simulators, where performance decreases with increasing traffic density[21], [56].

Planning related computations contribute to system constraints beside simulation updates. Navigation algorithms usually depends on graph search, collision checking, and cost evaluation. In dynamic situation these operations may be repeated frequently as the environment changes. Even in cases where route planning is performed prior to execution, the planner must still evaluate environmental conditions to ensure that the selected path remains feasible. This causes computational overhead, specially when large road networks or complex constraints are involved. Paden et al. mentioned that motion planning in autonomous driving involves a combination of global routing and local feasibility checks. both of these can become computationally demanding in real-time systems [53].

Another important factor is the prediction and modeling of other agents that adds further complexity to the planning process. Interaction aware systems mostly require estimating the future trajectories of surrounding vehicles. Even simplified prediction models introduce additional computation, because multiple potential behaviors must be considered. In more advanced methods like probabilistic or learning based prediction, the computational cost increases further due to the necessity of repeated inference or sampling. These requirements can limit the frequency at which planning updates are performed. It happens especially when combined with high fidelity simulation [61],[18].

Rendering and sensor simulation also cause a major computational bottleneck. High resolution camera streams, LiDAR point clouds, and depth maps require significant processing. Specially when multiple sensors are active. In GPU based simulators, rendering can dominate the total computation time. It reduces the available resources for planning and control. Studies on simulation platforms such as LGSVL and CARLA have shown that sensor simulation and rendering pipelines can become important factors. Specially in multi agent scenarios with multiple sensor equipped vehicles [56], [21].

System constraints are further caused by the need for synchronization between simulation components. Physics, rendering, and control modules must operate in a coordinated manner in many simulators. This often requires synchronization barriers. Here the simulation waits for all components to complete their updates before proceeding to the next time step. Althogh this ensures consistency it can introduce delays. Specially when one component becomes a bottleneck. For example, slow rendering or complex physics calculations can delay the entire simulation loop and affects overall performance and responsiveness.

Scalability requirements in large-scale experiments arises limitation. The total computational demand increases significantly during evaluating navigation systems across multiple scenarios. Running simulations sequentially may become impractical. It leads to parallelization or distributed execution. Parallel simulation introduces extra challenges like resource allocation, synchronization across processes,

and reproducibility of results. These difficulties documented in large scale simulation studies, where balancing throughput and consistency is a key concern [7], [14].

From the perspective of dynamic navigation functions, computational bottlenecks define the practical limits of adaptability. Even when a planning algorithm is capable of handling dynamic environments, its effectiveness depends on whether it can operate within the available computational budget. If updates are too slow, the planner may needs to rely on outdated information. It reduces its ability to respond to environmental changes. That is the reason, evaluation of navigation systems must consider algorithmic correctness as well as computational feasibility within the simulation environment.

In summary, simulation-based evaluation of navigation systems is constrained by multiple interacting factors like simulation updates, planning complexity, agent modeling, rendering, and system synchronization. These bottlenecks highlight the importance of designing both simulation environments and navigation algorithms with computational efficiency in mind. Specially when addressing dynamic and multi-agent scenarios.

2.5.4 Scalability and System-Level Limitations

The ability of simulation environments to scale is an Important factor in the evaluation of autonomous navigation systems. Although high-fidelity simulations provide detailed representations of traffic and environment dynamics, their ability to scale to large numbers of agents or extensive scenario sets is sometimes limited by computational resources. This Limitation becomes particularly relevant when navigation performance across multiple runs are evaluated, diverse traffic conditions, or in large geographic areas.

One of the primary challenges in scalability arises from the increase in interacting agents. With the growing number of vehicles in a simulation the complexity of the environment increases nonlinearly. Each newly added agent introduces new interactions and that must be resolved within the simulation loop. This affects both the simulation engine and the planning process. In microscopic traffic simulators scalability is achieved through simplified vehicle models and efficient update rules. This allows thousands of agents to be simulated simultaneously. However, this efficiency is often obtained by reducing physical realism and interaction complexity [40].

High-fidelity simulators like CARLA prioritize realism over scalability. Detailed physics, sensor simulation, alongside rendering pipelines limit the number of agents that simulated in real time. Empirical observations from simulation studies indicate that performance degrades as more vehicles and sensors are introduced. Specially when multiple agents require simultaneous perception as well as decision-making capabilities [21], [56]. This compromise between realism and scalability reflects a fundamental limitation of current simulation frameworks.

Another important consideration is the scalability of experimental evaluation. In dynamic navigation studies, it is necessary to evaluate planner performance across

multiple scenarios. Its so necessary for repeated runs to capture variability in traffic conditions. This leads to a combinatorial increase in simulation workload. Running these experiments sequentially can be time-consuming. On the other hand parallel execution requires careful management of computational resources. Distributed simulation approaches have been proposed to solve this issue. They introduce additional complexity in synchronization and result consistency [25], [9].

Scalability is also influenced by the level of abstraction used in the simulation model. Higher levels of abstraction allow larger environments and more agents to be simulated efficiently. But it may omit detailed interactions that are relevant for navigation evaluation. Lower levels of abstraction provide more realistic behavior. But it limit the scale of experiments. This trade-off requires careful selection of simulation parameters depending on the goals of the study. For example, large-scale traffic flow analysis may tolerate simplified dynamics but detailed interaction studies require higher fidelity.

Resource allocation plays an important role in determining scalability from a system perspective. Simulation performance depends on the availability of CPU as well as GPU resources. Also on memory bandwidth, and data transfer efficiency. Rendering, physics, and planning components compete for these resources. In multi-agent scenarios with several vehicles equipped with sensors, resource demands can increase further. In this case, scalability is a property of the simulation model as well as of the underlying hardware and system configuration.

In the context of dynamic navigation functions, scalability limitations define the boundaries. By this a system can be evaluated under realistic and diverse conditions. Although smaller scale simulations allow detailed analysis of specific scenarios, larger-scale simulations are important to assess robustness and generalization. That is why an effective evaluation strategy often consists of combination of both approaches, balancing detail as well as scale according to the objectives of the study.

The trade-off between simulation fidelity and scalability can be summarized by comparing key system characteristics. High-fidelity simulation environments prioritize physical realism and detailed sensor modeling, whereas scalable simulation approaches focus on computational efficiency and the ability to handle large numbers of agents. The main differences between these approaches are outlined in Table 2.5.

Aspect	**High-Fidelity Simulation**	**Scalable Simulation**
Number of Agents	Limited	High
Physics Modeling	Detailed	Simplified
Rendering	High-quality	Reduced
Computational Cost	High	Lower
Experiment Throughput	Low	High

Table 2.5: Trade-offs between simulation fidelity and scalability in autonomous driving environments.

In summary, scalability challenges arise from the interaction between simulation fidelity, computational resources, and experimental design. These constraints changes both the feasibility as well as the interpretability of simulation based evaluations. Understanding these limitations is essential for designing meaningful experiments. Also for interpreting the performance of navigation systems within the broader context of real-world applicability.

2.6 Analysis of Existing Approaches and Identified Gaps

The previous sections have reviewed fundamental approaches to navigation and path planning, traffic modeling, interaction aware navigation, and simulation based evaluation. Existing research provides a broad range of methods for generating feasible trajectories. Also, modeling multi-agent environments, and evaluating navigation performance under controlled conditions. Despite having these advances, various limitation remain when applying these methods within simulation environments. This simulation aim to reflect dynamic traffic conditions while maintaining computational feasibility.

A central observation from the literature is that many approaches focus either on algorithmic optimality or on highly detailed interaction modeling, often assuming continuous replanning or full access to real-time information. Although these assumptions are appropriate for certain research settings, they are not always aligned with practical simulation workflows. Here navigation decisions are frequently evaluated under scenario-driven conditions and limited computational resources. This creates a gap between theoretical formulations of dynamic navigation and their application in structured simulation environments [53], [61].

In scenario based simulation frameworks those based on OpenSCENARIO or related approaches, they emphasize the generation of controlled traffic situations to evaluate system behavior. Although these frameworks enable systematic testing of navigation strategies, they often treat planning and scenario execution as separate components. As a result, the interaction between environment configuration and route selection is not always explicitly addressed. Specially at the level of global navigation decisions [22], [25].

A further difficulty is the evaluation of navigation under varying environmental conditions. Several studies assess planner performance within fixed scenarios. These methods concentrate on generating local trajectory generation without considering how changes in the environment influence route level decisions. Factors like blocked roads, traffic density, agent behavior can alter the feasibility as well as efficiency of a route. This highlights the need for navigation approaches that account for environment dependent variability within simulation based experiments [21], [9].

Simulation environments introduce constraints related to computational performance and scalability as discussed in 1.5. These constraints reduces the applicability

of approaches that rely on frequent replanning or complex interaction modeling. So, there is a need for navigation strategies that can operate effectively within these constraints while still capturing essential aspects of dynamic behavior.

Based on these observations, the research gap can be characterized as the lack of approaches that systematically connect environment conditions, scenario-based simulation, and route-level navigation decisions within a unified framework. Existing methods either focus on detailed local planning or on large-scale traffic simulation, but fewer works address how global navigation strategies can adapt to environment configurations in a structured and computationally feasible manner.

The following subsections analyze this gap in more detail and position the contribution of this work within the context of existing research.

2.6.1 Limitations in Planning Approaches

Classical and learning based planning methods both provide a strong foundation for autonomous navigation. When it is considered in the context of simulation based evaluation under varying traffic conditions, several limitations appear. These limitations reflect assumptions that are not always aligned with the requirements of dynamic and scenario-driven environments.

One of the limitations concern the assumption of static or weakly dynamic environments in many global planning approaches. Algorithms like Dijkstra and A* operate on predefined graphs and compute optimal paths based on fixed edge costs. There are extensions exist to manage time dependent costs or dynamic updates. But it often assumes that the structure of the environment remains stable during planning. Practically, factors such as road blockages, traffic congestion, scenario-specific constraints can alter the effective topology of the road network. As a result, routes computed under static assumptions may not reflect the actual conditions that are obtained during simulation [29], [17].

A further limitation relates to the division between global route planning and local motion planning. Many navigation systems follow a hierarchical structure. On the other hand a global planner determines a route on a road network. Beside a local planner handles obstacle avoidance and trajectory generation. Although this decomposition improves computational efficiency it introduces a disconnect between route level decisions and local environmental conditions. For example, a globally optimal route has a chance to become suboptimal or infeasible due to local interactions with traffic agents. Existing methods often rely on local planners to compensate for such issues, rather than adapting the global route itself [53],[75].

Another important limitation is the reliance on continuous replanning assumptions in many advanced planning frameworks. Approaches based on model predictive control, sampling based planning, optimization techniques sometimes assume that the planner can update its decisions at high frequency. Although this is suitable for systems with sufficient computational resources, it may not be practical in simulation environments. Here physics updates, rendering, and multi-agent interactions already impose significant computational load. Consequently, the usuability of such

methods may be constrained when operating under limited computational budgets [42], [33].

Many navigation planning methods do not explicitly responsible for environment dependent variability at the route level. Although local planners often react to nearby obstacles or agents, fewer approaches indicates how global navigation decisions should adapt to changes. For example blocked roads or scenario-specific constraints. This is particularly relevant in simulation based studies where environmental conditions can differ between different runs. Navigation strategies remain fixed even when the environment changes without cosidering This variability into the planning process, . This limits their adaptability [61], [69].

Furthermore, learning based planning methods rises additional challenges related to generalization and interpretability. Deep learning models that are trained on specific datasets may not generalize well to unseen scenarios or different traffic conditions. In simulation environments scenario configurations can vary significantly. This lack of generalization can affect the reliability of learned navigation policies. Also, the decision making process of such models is mostly difficult to interpret. This complicates the analysis of failure cases and limits their suitability for systematic evaluation [14], [7].

Together these limitations indicate that existing planning approaches are designed with assumptions that do not fully align with the requirements of scenario based simulation as well as environment dependent navigation. particularly the gap lies in the integration of global route planning with changing environmental conditions in a structured and computationally feasible way. Addressing this gap requires reconsidering how planning strategies incorporate environment information at the route level.Rather than relying solely on local adaptation or continuous replanning.

2.6.2 Limitations in Simulation-Based Evaluation

Simulation environments are now a days become a standard tool for evaluating autonomous navigation systems. They allow controlled experimentation, reproducibility as well as safe testing of complex traffic scenarios. Despite these advantages, several challenges arise when simulation is used in order to assess navigation performance. This is particularly in dynamic and scenario-driven contexts.

One limitation concerns the gap between scenario definition and navigation behavior. Scenario based frameworks such as OpenSCENARIO provide structured descriptions of traffic situations. This includes agent configurations, trajectories as well as environmental conditions. These scenarios are typically designed to test specific behaviors. for example lane changes or obstacle avoidance. The navigation system itself is often evaluated within a fixed route or predefined mission without explicitly linking scenario conditions to route level decisions. because of that, the planner operates within the scenario rather than adapting its navigation strategy. This is based on the scenario configuration [22], [25].

A second limitation is related to the restricted variability of evaluation conditions. Although simulation allows repeated execution of scenarios, many studies

rely on a small set of predefined configurations. Even when stochastic elements are given, variations are often constrained to local behavior. For example vehicle speeds or lane changing probabilities. This can make the evaluation results sensitive to specific scenario setups. Rather than reflecting broader performance across diverse conditions. Benchmark datasets and simulation based evaluations have mentioned this issue. It showes that performance can vary significantly depending on scenario selection [9] [67].

Another important limitation arises from the abstraction of environment dynamics. Many simulation environments simplify aspects of real world driving. That includes sensor noise, driver behavior, as well as environmental uncertainty. Although such abstractions are necessary for computational efficiency, they can affect the realism of interactions and the validity of evaluation outcomes. For example, rule based traffic agents may not fully capture the variability of human driving behavior. That leades to interactions that are more predictable than those observed in real world conditions [40], [21].

Simulation based evaluation often assumes difference between perception, planning, and control modules. Although this modular structure simplifies system design and analysis, it may overlook dependencies between these components. Particularly, perception errors or delayed information can influence planning decisions. This in turn affect navigation performance. If this kind of interactions are not fully represented in the simulation, evaluation results may not accurately reflect real-world system behavior [13], [31].

Another big challenge is the limited consideration of long term navigation effects. Many evaluation setups focus on short scenarios or isolated events. For example intersection handling or obstacle avoidance. Although these tests are useful for analyzing specific behaviors, they do not capture how navigation decisions accumulate over longer routes. For example, a planner may successfully handle individual interactions. But it may choose inefficient routes when considered over an entire journey. This represent the need for evaluation strategies incorporating both local behavior and route-level outcomes [53].

Finally, the constraints discussed in Section 1.5, including computational limits and scalability those influence simulation based evaluation. High-fidelity simulations may restrict the number of scenarios or agents that can be tested. Scalable simulations may simplify important aspects of the environment. These limitations affect both the breadth and depth of evaluation. It makes difficult to achieve a balance between realism and coverage.

Taken together, these limitations indicate that current simulation based evaluation approaches do not fully capture the relationship between environment configuration. it includes traffic dynamics, and navigation decisions. In particular, the connection between scenario conditions and route level planning remains underexplored. Addressing this issue requires evaluation strategies. It integrate environmental variability more directly into the navigation process, rather than treating scenarios as external test conditions.

2.6.3 Identified Gap and Positioning

The analysis in previous two sections indicates that existing work addresses important aspects of autonomous navigation. But it does not fully integrate environment conditions, scenario configuration, as well as route level decision making within a unified framework. Planning methods often assume static network conditions or rely on frequent local replanning. On the other hand Simulation based evaluations emphasize controlled scenarios. But it typically decouple these scenarios from global navigation decisions. This separation limits the ability to study how changes in the environment influence route selection in a structured and reproducible manner.

A central gap can therefore be identified at the interface between scenario-based simulation and global navigation planning. While local planners adapt to nearby agents and obstacles, fewer approaches address how environment conditions—such as blocked roads or altered traffic configurations—should influence the initial route selection prior to execution. In many studies, the route is fixed and the planner reacts locally during simulation. As a result, the impact of environment variability on route-level decisions remains underrepresented in current evaluation practices [53], [61].

At the same time, scenario specification frameworks provide mechanisms to define structured traffic situations, including agent behavior and environmental constraints. These frameworks are used to test system responses within predefined conditions. However, the integration of such scenario information into the navigation process itself is limited. Particularly, the use of scenario specific information to influence route generation before execution is not systematically explored [22], [25].

Another aspect of the gap indicates computational feasibility within simulation environments. As discussed in previous sections, high frequency replanning and complex interaction modeling can be computationally demanding. Although many advanced planning methods assume continuous updates, simulation based methods often operate under constrained resources. This creates a necessity navigation strategies that incorporate environmental information without relying on constant replanning during execution.

Based on these observations, this work focuses on a navigation approach that integrates environment-dependent route generation within a scenario-driven simulation workflow. Instead of adapting the route continuously during execution, the navigation strategy considers the environmental conditions present in each simulation run. It generates a route accordingly before execution begins. This allows the planner to reflect scenario specific constraints. for example blocked roads, while maintaining a structured and computationally manageable process.

Within the CARLA simulation environment, this approach enables the evaluation of navigation behavior across different traffic configurations by systematically varying environmental conditions between runs. In this way, the navigation function can be considered dynamic at the level of scenario dependent adaptation, rather than continuous online replanning. This perspective aligns with the practical constraints of simulation based experimentation. It still capturing the influence of environmen-

tal variability on navigation outcomes.

The contribution of this work therefore lies in bridging the gap between scenario configuration and route level navigation. It a framework in which environmental conditions directly influence navigation decisions in a controlled and reproducible way. This positioning complements existing planning approaches by addressing a level of adaptation that is often overlooked in both planning and simulation literature. The following chapter presents the methodology and implementation details of the proposed navigation framework.

3 Methodology

This chapter presents the methodology developed in this thesis for dynamic navigation function in the CARLA simulation environment. Based on the research gap identified in Chapter 2, the proposed approach addresses the problem of how route generation can reflect the conditions of a given simulation run before execution begins. The focus is therefore placed on a structured planning workflow in which scenario dependent environmental constraints are incorporated into the navigation process at the route generation stage.

The following sections describe the methodology in detail. The chapter begins with an overview of the complete workflow, followed by the methodological problem setting, graph construction strategy, integration of environmental constraints, route computation logic, and route execution process.

3.1 Overview of the Proposed Methodology

The proposed methodology is based on the principle that route generation in simulation should reflect the conditions under which the navigation task is actually executed. In conventional navigation planning settings, the road network is often treated as a fixed structure from which a route is computed once a start and destination are given. In the context of this thesis, however, such a view is not sufficient. The validity of a route may depend on the scenario specific state of the environment, particularly when the simulated road network contains blocked or otherwise unusable segments. For this reason, the methodology does not separate route generation from environment configuration. Instead, it establishes a direct connection between the two.

At the highest level, the proposed methodology consists of a staged workflow. The process begins with the characterization of the relevant environment conditions of the active simulation run. These conditions define which parts of the nominal road network should be considered available for route generation and which parts should be excluded from the effective planning space. Based on this information, a planning representation of the drivable environment is constructed in a way that preserves the navigational structure of the map while allowing environment dependent restriction of the search space. Route generation is then performed on this adapted representation, after which the resulting path is transferred into an executable route for the ego vehicle.

The methodological significance of this workflow lies in the ordering of its stages. The environment is not treated as a secondary factor that only affects route execu-

tion after a route has already been assigned. Instead, environmental conditions are incorporated before route computation takes place. As a result, the generated route reflects the effective state of the scenario rather than only the nominal map topology. This is the central methodological idea of the thesis and the defining characteristic of the proposed navigation framework.

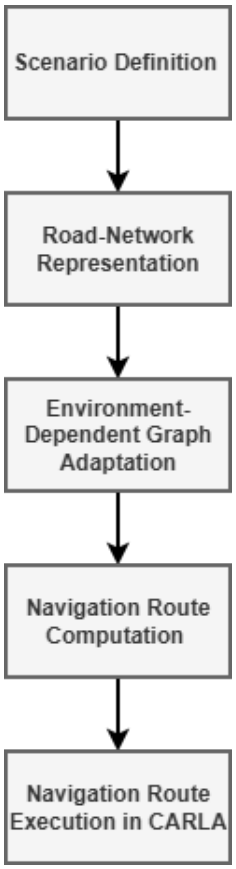


Figure 3.1: Conceptual workflow of the proposed navigation methodology in CARLA

Figure 3.1 presents the overall methodological workflow of the proposed navigation framework. The process begins with the definition of scenario dependent conditions, continues with the representation and adaptation of the road network for navigation, and then computes a route on the resulting planning space. The computed route is finally transferred to the ego vehicle for execution in the CARLA simulation environment.

3.2 Environment Aware Route Adaptation for Navigation

The central methodological principle of this thesis is that route generation should depend on the environmental conditions of the active simulation run. In a nominal route-planning setting, the road network is treated as fully available and the route is computed on that unchanged representation. Such an approach is sufficient only as long as the effective driving space remains identical across runs. In the context of this work, however, this assumption is not always valid. A simulation scenario may include blocked or otherwise undesired road segments, and these conditions directly affect whether a route remains suitable for execution. For this reason, the proposed methodology performs route generation on an environment dependent planning space rather than on the nominal road network alone.

This planning space is represented through a graph based abstraction of the drivable road network. The graph provides a structured representation of connectivity between reachable parts of the map and therefore serves as the basis for navigation oriented route computation. Methodologically, the graph is not treated as a fixed object. Its effective form depends on the conditions of the scenario under consideration. Segments that should not be used for the current run are excluded from the feasible planning space before route computation begins. As a result, the route is generated on the basis of what is effectively traversable in the scenario rather than what is only geometrically present in the underlying map.

The adaptation of the planning graph constitutes the core of the proposed navigation methodology. Instead of computing a baseline route first and resolving all conflicts later during execution, the method integrates relevant environmental constraints at the route-generation stage. This changes the role of the environment within the navigation process. The environment is no longer treated merely as an external execution context. It becomes an important factor in the generation of the route itself. In methodological terms, it creates a direct dependency between scenario configuration and route selection.

An important aspect of this approach is the interpretation of changing environmental conditions across simulation runs. The proposed methodology does not assume continuous global replanning during vehicle motion. Instead, it assumes that when the conditions of a new run differ from those of a previous run, the planning space is adapted accordingly and a new route is generated before execution starts. In this sense, a changed scenario condition can trigger a new planning outcome in the next run. This preserves the dynamic character of the navigation function at the level of run specific adaptation while remaining consistent with a structured and computationally manageable workflow.

From a navigation perspective, this methodology offers two main advantages. First, it improves the consistency between the scenario under evaluation and the route assigned to the ego vehicle. Second, it enables a controlled comparison between nominal and constrained route generation without requiring a fully reactive online

replanning system. The method therefore remains aligned with the practical scope of the thesis which to investigate how environmental conditions can influence route level navigation in CARLA in a transparent and reproducible way.

The output of this adaptation stage is a planning representation in which the feasible route space already reflects the relevant conditions of the active scenario. On this basis, route computation can proceed in a way that is both structurally clear and methodologically consistent with the navigation problem defined in Chapter 1. The next section therefore describes how the resulting route is prepared for execution and transferred to the ego vehicle within the simulation environment.

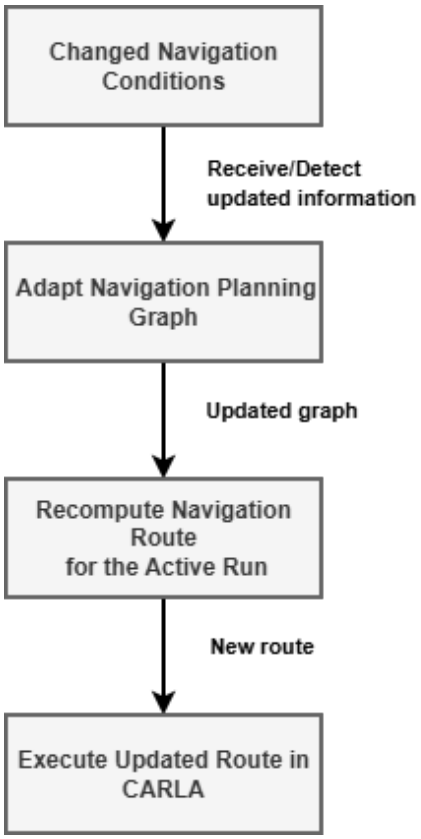


Figure 3.2: Environment-aware navigation route adaptation under changed scenario conditions

Figure 3.2 illustrates the adaptation logic of the proposed navigation methodology under changed scenario conditions. The process begins when the active run contains navigation-relevant environmental differences, such as blocked or undesired road segments. These changed conditions lead to an adaptation of the navigation planning graph so that the effective planning space reflects the current scenario. On the basis of this adapted graph, a new navigation route is computed for the active run. The resulting route is then transferred to the execution stage and followed by the ego vehicle in the CARLA simulation environment

3.3 Route Preparation and Execution

After the route has been generated on the environment-aware planning representation, it must be transformed into a form that can be executed within the simulation environment. In methodological terms, this stage connects abstract route computation to vehicle behavior in CARLA. The purpose of this step is not to change the route itself, but to prepare the planning result for downstream execution by the ego vehicle. The route generated in the planning stage therefore becomes the navigational reference that guides the subsequent movement of the vehicle in the simulator.

Within the proposed methodology, this preparation step serves as a bridge between planning and execution. The output of route computation is an ordered route description that captures how the vehicle should progress from the selected start position to the destination under the conditions of the active scenario. For execution in CARLA, however, this route must be expressed in a form that is compatible with the simulator's navigation and control framework. Methodologically, this means that the abstract planning result is translated into an executable sequence of route elements that can be followed by the ego vehicle during the simulation run.

This stage is important because the route-generation process would remain incomplete if the computed path were not linked to actual vehicle motion. The methodology proposed in this thesis is therefore not limited to identifying a theoretically valid route on an adapted graph. It also includes the transfer of this route into the execution layer of the simulator, where the ego vehicle follows the planned path toward the destination. In this way, the navigation framework maintains continuity between the planning logic and the realized driving behavior inside the simulation environment.

A further methodological aspect of this stage is that route execution remains consistent with the planning logic introduced earlier in the chapter. The vehicle does not receive a generic or nominal route after the planning phase. Instead, it follows the specific route that was generated for the active scenario after environmental constraints had already been incorporated into the planning space. This preserves the central idea of the thesis throughout the full workflow. The route executed in the simulator is the direct outcome of scenario-dependent navigation planning.

At the same time, the execution stage should be understood as distinct from continuous route adaptation during movement. The purpose of this step is to realize the planned route in the simulator, not to redefine the route repeatedly during the same run. If the conditions of a subsequent run change, a new route can be generated before that new execution begins. The execution process itself therefore remains tied to the route produced for the current scenario instance. This preserves methodological clarity and keeps the navigation framework consistent with the defined scope of the thesis.

From a structural point of view, this stage completes the methodology of the proposed navigation approach. The earlier sections define how the route is generated under environment-aware constraints, while the present section explains how this

route becomes operational within CARLA. The next chapter then moves from this methodological description to the concrete implementation of the framework in the developed system.

3.4 Methodological Summary

This chapter presented the methodology developed for dynamic navigation function in the CARLA simulation environment. The proposed approach was introduced as a structured navigation framework in which route generation is conditioned on the relevant constraints of the active simulation run. In contrast to a purely nominal planning process, the methodology treats the road network as a planning space whose effective usability depends on scenario dependent environmental conditions.

The chapter first outlined the overall methodological workflow and then described how changed navigation conditions influence the route-generation process. The central methodological principle of the proposed approach is that environmental constraints are incorporated into the planning representation before route computation begins. As a result, the generated route reflects the effective state of the scenario rather than only the nominal road topology. The chapter also explained how the computed route is prepared for execution and transferred to the ego vehicle within the CARLA simulation environment.

Taken together, the methodology establishes a direct connection between scenario configuration, planning space adaptation, route computation, and route execution. This provides the conceptual foundation for the implemented navigation system and ensures that the route followed by the ego vehicle is consistent with the environment conditions of the corresponding simulation run.

The next chapter presents the implementation of this methodology in detail. It describes how the proposed framework was realized in CARLA. Also how the planning logic was integrated into the developed Python system and how the navigation process was executed across the selected simulation scenarios.

4 Implementation

This chapter presents the implementation of the proposed navigation framework in the CARLA simulation environment. While Chapter 3 described the methodological logic of the approach at a conceptual level, the present chapter explains how that methodology was realized in practice. The focus is therefore placed on the concrete structure of the developed system, the realization of the planning pipeline in Python, and the integration of the navigation logic into CARLA for experimental execution.

The implementation was designed to preserve the central principle established in the previous chapter: route generation should reflect the environmental conditions of the active simulation run before execution begins. To realize this principle, a graph-based planning pipeline was developed that combines map based road network representation, environment aware graph adaptation, route computation, and execution by the ego vehicle. The final system was implemented and tested in different CARLA towns and under multiple scenario configurations, allowing the proposed approach to be evaluated under varied blocked road conditions.

Along with the core route generation logic, the implemented framework also includes supporting components required for simulation-based experimentation. These include scenario mode handling, traffic and blocker generation, camera based route visualization, and also CSV based run logging. All together, these components form an executable system that not only computes environment aware routes but also enables their observation and evaluation within the CARLA simulator. The following sections describe this implementation in a structured manner, beginning with an overview of the realized system.

4.1 Implementation Overview

The implemented system realizes the proposed navigation methodology as a structured planning and execution pipeline in CARLA. Its main function is to generate a route for a specific simulation run on the basis of the environmental conditions defined for that run and then to execute the resulting route with the ego vehicle inside the simulator. In this sense, the implementation provides an operational realization of the environment aware route generation concept developed in the previous chapter.

At a high level, the implementation consists of four main functional parts. The first part is scenario realization, in which the simulation run is configured through the selected town, the start and destination positions, and the scenario dependent road constraints. The second part is planning space construction, where the drivable

road network of the current map is transformed into a graph representation suitable for route computation. The third part is route generation, in which the graph is adapted according to the active scenario and a valid route is computed on the resulting planning space. The fourth part is route execution and observation, in which the generated route is passed to the ego vehicle, followed in CARLA, and observed through visualization and run logging. This structure is reflected in both the Town01 and Town02 implementations, although the specific scenario design differs between the two systems.

The implementation was developed in Python and relies on CARLA's simulation APIs together with navigation related components available inside the CARLA framework. In both towns, the navigation planning logic is embedded in a simulation loop that includes map loading, graph construction, actor spawning, route assignment, and execution control. The resulting system is therefore not a collection of isolated scripts, but a coherent pipeline in which scenario configuration, planning, and execution are directly connected. This is an important implementation characteristic because it ensures that the route followed by the ego vehicle is the route that was actually computed under the scenario specific navigation constraints of that run.

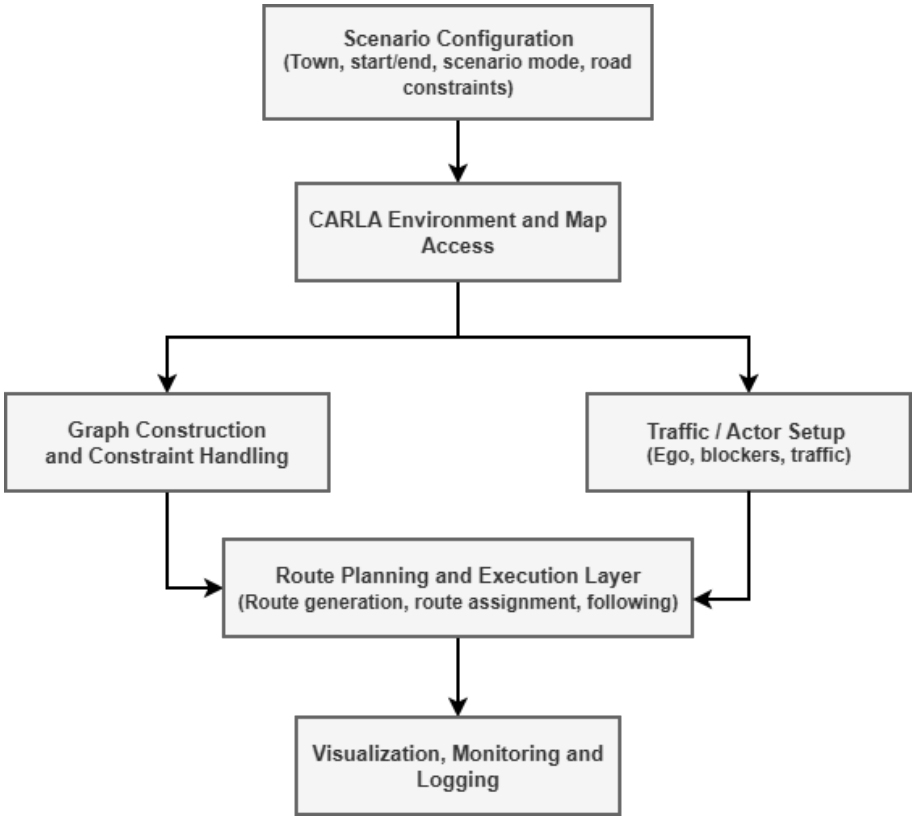


Figure 4.1: System-level architecture of the implemented navigation framework in CARLA

Although the same general logic is preserved across the developed system, the two town implementations are not identical. In Town02, the implementation emphasizes graph filtering through blocked coordinate sets and route computation under different blockage modes. In Town01, the system extends this logic by combining blocked road filtering with route shaping elements in selected scenarios, including the use of intermediate via locations and the additional comparison between baseline and adapted route behavior. These differences do not change the methodological foundation of the framework, but they do provide broader implementation coverage and allow the system to be examined under more than one type of constrained navigation setting.

A further characteristic of the implementation is that it includes not only route computation but also supporting components for execution analysis. The system uses camera based visualization to display route behavior in the simulator and CSV based logging to record execution related information over time. These additions are important from an implementation perspective because they transform the framework from a planning prototype into an executable and observable simulation system. As a result, the implementation supports not only route generation itself but also the later evaluation of navigation behavior under different scenario conditions.

Figure 4.1 illustrates the main implemented components of the navigation framework and their interaction within the CARLA environment. It shows how scenario configuration and map access provide the basis for graph construction, constraint handling, actor setup, route planning and execution, and the subsequent visualization, monitoring, and logging of the simulation run.

The following sections describe the implementation in greater detail. They explain how the system components were organized, how the navigation scenarios were realized in Town01 and Town02, how environment-aware route generation was implemented, and how execution behavior was visualized and recorded during the simulation runs.

4.2 System Components and Software Structure

The implemented navigation framework was organized as a structured software pipeline in which scenario handling, planning, execution, and observation were integrated into a single CARLA-based system. From an implementation perspective, the objective was not only to compute an environment-aware route, but also to realize a complete simulation workflow in which the route could be generated, executed, visualized, and recorded under reproducible scenario conditions. For this reason, the developed system consists of several interacting functional components rather than a single isolated planning script.

A first central component of the implementation is the scenario configuration layer. This part of the system defines the simulation context in which a run is executed. It includes the selected town, the start and destination positions, the scenario mode, and the set of road constraints associated with that mode. In practical

terms, this layer determines whether a run corresponds to a nominal case or to a constrained navigation case with blocked or undesired road segments. This component is especially important because it forms the connection between experimental scenario design and the actual navigation logic implemented later in the pipeline.

A second component is the environment and map access layer. This part of the implementation is responsible for loading the relevant CARLA world, accessing the map topology, and providing the road network information required for planning. The road topology provided by CARLA serves as the basis for constructing the internal graph representation that is used for route computation. In this way, the implementation does not rely on a manually defined map abstraction. Instead it derives the planning structure directly from the simulator environment. This contains a significant implementation property because it ensures that route generation remains grounded in the actual road layout of the selected town.

The third major component is the graph construction and constraint handling module. This module translates the drivable road structure into a graph based planning representation and integrates the navigation constraints associated with the active scenario. From a software perspective, this is one of the most important parts of the system because it determines the feasible search space on which the route is later computed. The implementation therefore combines graph construction with environment aware filtering rather than treating them as fully separate stages. This design keeps the planning representation closely tied to the effective road availability of the scenario.

A fourth component is the route planning and execution layer. Once the graph has been prepared, this part of the implementation is responsible for computing a route between the selected start and destination and transferring the resulting route into a form that can be executed inside CARLA. This layer therefore combines two tasks that are conceptually distinct but operationally connected. These are route generation and route following. In the final system, the planning result is not stored only as an abstract graph path. It is transformed into an executable navigation route that can be assigned to the ego vehicle within the simulator. This makes the route planning stage directly relevant for observable vehicle behavior during a run.

Another important part of the software structure is the actor and traffic setup component. In addition to the ego vehicle, the implemented system can also create traffic vehicles and static blockers depending on the selected scenario. This component supports the realization of the constrained navigation conditions that are needed for the experiments. It also ensures that route execution takes place in a simulation context that contains the necessary environmental elements for the scenario under investigation. In this sense, actor setup is not only a simulation convenience feature, but a direct implementation requirement of the overall framework.

The final major component is the visualization and logging layer. The implementation was extended beyond route computation itself by integrating route visualization and CSV based run logging. These additions serve an important role in the overall software structure because they make the behavior of the implemented system observable and analyzable during execution. The framework therefore supports

not only navigation in the strict sense, but also the monitoring and recording of the resulting simulation behavior. This is particularly relevant for the later evaluation chapter, where route outcomes and execution-related observations must be analyzed in a systematic manner.

Taken together, these components form a coherent implementation architecture. The overall system begins with scenario definition, proceeds through map access and planning-space construction, generates and executes a route, and finally records the outcome of the run through visualization and logging. Although the specific scenario logic differs between Town02 and Town01, the same general software structure is preserved across both realizations. This consistency is important because it allows the same navigation mechanism to be implemented and examined across more than one CARLA town, thereby supporting the broader validity of the developed framework.

4.3 Scenario Realization in Town02 and Town01

The proposed navigation framework was implemented and tested in two CARLA towns, namely Town02 and Town01. The purpose of using both environments was not to create two independent navigation systems, but to realize the same environment aware navigation mechanism across different urban layouts. The initial implementation was carried out in Town02, where the central route adaptation logic was first established. The framework was then applied to Town01 in order to examine whether the same navigation principle could also be realized in a second town with a different road structure. In this way, the use of two towns strengthens the implementation by showing that the developed mechanism is not limited to a single CARLA map.

From an implementation perspective, both towns follow the same general idea. In each case, a simulation world is loaded, a scenario specific set of navigation conditions is defined, the road network is represented in a planning compatible form, the feasible planning space is adapted according to the active scenario, and a route is then generated and executed by the ego vehicle. The difference between the two town implementations lies not in the underlying navigation concept, but in the way the scenario conditions are realized within the respective map layout.

Town02 as the Primary Implementation Environment

Town02 was used as the primary environment for the realization of the navigation mechanism. In this town, the implementation was structured around three scenario modes. The first mode represents a nominal case without blocked road segments. The second mode introduces constrained navigation conditions by defining undesired road regions that are excluded from the planning space. The third mode extends this setup by adding further blocked segments, thereby increasing the restriction of the feasible route space. Through these progressively constrained scenarios, Town02

provides the main implementation setting in which the effect of environment aware graph adaptation on route generation can be observed clearly.

The Town02 realization uses fixed start and destination configurations that depend on the selected scenario mode. The environmental constraints are defined through coordinate based blocked or undesired regions, which are used both for the realization of the scenario inside the simulation and for the adaptation of the planning graph. As a result, the generated route reflects the effective road availability of the active run rather than the nominal road topology alone. This makes Town02 the clearest implementation of the core planning principle developed in this thesis.

Town01 as an Additional Implementation Environment

After the mechanism had been realized in Town02, the same planning principle was applied to Town01. The purpose of this second implementation was to show that the proposed navigation framework could also be realized under a different town structure and was therefore not tied only to a single map layout. Town01 thus serves as an additional implementation environment that complements the primary Town02 realization and supports the broader applicability of the framework across different CARLA towns.

Like Town02, the Town01 implementation uses scenario specific conditions that influence the feasible planning space before route generation begins. The same environment aware logic is preserved. The route assigned to the ego vehicle is generated after the relevant scenario constraints have been incorporated into the planning representation. In selected constrained scenarios, additional scenario specific guidance was introduced in order to keep the generated route aligned with the intended feasible road corridor of the Town01 layout. This does not change the central navigation mechanism of the framework, which remains based on environment aware adaptation of the planning space followed by route generation and execution. Rather, it reflects an implementation adjustment required by the characteristics of the second town.

An additional benefit of the Town01 realization is that it allows the route behavior under constrained conditions to be examined in a second map context. This strengthens the implementation chapter because it shows that the navigation logic can be applied beyond its initial realization environment. The role of Town01 is therefore not to define a separate method, but to support the implementation level validation of the same route adaptation principle in another town structure.

Shared Implementation Principle Across Both Towns

Despite the differences in scenario realization, both town implementations are based on the same core principle. In both cases, the route followed by the ego vehicle is not treated as a fixed output of the nominal map. Instead, it is generated after the planning space has been adapted according to the environmental conditions of the active scenario. This ensures that the resulting navigation behavior remains directly

linked to the run specific constraints of the simulation. The use of both Town02 and Town01 is therefore meaningful because it demonstrates two realizations of the same environment aware route generation mechanism rather than two separate planning approaches.

The town specific scenario realization process used in the implemented framework is summarized in the algorithm box bellow.

Algorithm: Scenario realization in Town02 and Town01

Input: town T, scenario mode M, start configuration, destination configuration
Output: scenario-specific environment for navigation execution

load the selected CARLA town
access the corresponding map and available spawn points
determine the start and destination configuration
load the scenario specific environmental constraints

if $T =$ Town02 **then**
    define blocked or undesired road regions according to M
    prepare the scenario specific planning space
else if $T =$ Town01 **then**
    define the corresponding constrained road regions
    apply additional route guidance only if required by the scenario
end if

spawn ego vehicle and supporting actors
return the configured environment for planning and execution

Algorithm above summarizes how the implemented system prepares the simulation environment before route generation begins. Although the concrete scenario constraints differ between Town02 and Town01, both realizations follow the same environment-aware preparation logic. This ensures that route computation is carried out on a scenario specific planning space rather than on the nominal map alone.

4.4 Graph Construction and Constraint Integration

The graph construction process begins with access to the map topology of the selected CARLA town. This topology provides the structural road information from which the internal planning representation is derived. In the implementation, the drivable road network is transformed into a graph whose nodes represent reachable road positions and whose edges represent navigable connectivity between them. The

resulting graph serves as the computational basis for route generation. A key implementation decision is that this graph is not loaded as a fixed predefined structure from an external planning dataset. Instead, it is built directly from the CARLA map information at runtime, which keeps the planning representation closely tied to the active simulation environment.

To obtain a usable route planning structure, the implementation samples the road topology and converts the resulting drivable positions into graph nodes. Connectivity between successive nodes is then represented through weighted edges. In both town implementations, this graph building stage is designed to preserve the navigable structure of the selected map while also allowing the later integration of scenario specific restrictions. This makes the graph construction stage more than a simple map conversion step. It is the foundation on which the environment aware planning mechanism operates.

The second core aspect of this section is the integration of constraints into the graph. The implemented system uses scenario dependent blocked or undesired road regions to determine which parts of the graph should not remain available for route generation in the active run. In Town02, this is realized through predefined coordinate based lists representing blocked or undesired road segments. During graph construction, nodes located within these constrained regions are excluded from the feasible planning space. This means that the graph used for route computation is already adapted to the environmental condition of the selected scenario. As a result, the later route generation stage does not need to treat blocked roads as merely local execution disturbances, because the restrictions have already been integrated into the planning representation itself.

In Town01, the same principle is preserved, although the constraint handling is applied in a somewhat richer scenario context. The graph is again constructed from the CARLA road structure and then adapted according to the scenario specific road restrictions. The important implementation point is that Town01 does not replace the graph adaptation logic with another planning mechanism. Instead, the same environment aware planning principle is maintained in the second town realization, which supports the claim that the proposed framework is not confined to one isolated map configuration.

From an implementation perspective, the integration of constraints during graph preparation is the most direct realization of the thesis contribution. It is this step that transforms a nominal route planning pipeline into an environment aware one. If the graph were left unchanged and blocked roads were handled only after route assignment, the resulting system would no longer reflect the planning concept developed in the methodology chapter. By contrast, the implemented graph construction and filtering process ensures that route generation is carried out on a planning space that corresponds to the active scenario rather than to the unconstrained map.

A further advantage of this design is that it keeps the route generation stage transparent and interpretable. The adaptation of the planning graph can be directly linked to the scenario conditions defined for the run, and the resulting changes in the feasible route space can be traced systematically. This is useful not only from

a software perspective, but also from an evaluation perspective, because it makes it possible to analyze how route outcomes differ under nominal and constrained graph conditions. In this sense, graph construction and constraint integration form the central implementation mechanism that connects scenario design to navigation behavior in the developed system.

The implementation logic for graph construction and scenario dependent constraint integration is summarized in the algorithm box bellow.

Algorithm: Graph construction and constraint integration

Input: CARLA map topology, sampling resolution, scenario constraint set C
Output: environment-aware planning graph G

retrieve road topology from the selected CARLA map
initialize empty graph G

for each drivable road segment in the topology **do**
    sample reachable positions along the segment
    convert sampled positions into graph nodes
    connect consecutive nodes with weighted edges
end for

for each node in G **do**
    if node lies inside a blocked or undesired region in C **then**
    remove node from the feasible planning space
    end if
end for

return the filtered planning graph G

Algorithm above summarizes the central implementation step that enables environment aware route generation for navigation in the developed system. By integrating scenario specific restrictions into the graph before route computation begins, the implementation ensures that the planning space used by the route planner reflects the effective road availability of the active simulation run.

4.5 Route Generation and Execution Pipeline

After the navigation planning graph has been constructed, the next implementation stage is the generation of a route from the selected start position to the destination and the transfer of that route into executable vehicle behavior in CARLA. This stage forms the operational core of the implemented system because it connects the prepared planning space to the actual movement of the ego vehicle in the simulation. In the developed framework, route generation and route execution are implemented as consecutive stages of a single navigation pipeline rather than as isolated processes.

Route Generation

The route generation process begins by mapping the relevant start and destination locations to the planning graph. Since the graph is built as a discrete representation of the drivable road network, the selected simulation positions must first be associated with graph nodes that can serve as valid planning endpoints. Once these endpoints are established, the route planner computes a feasible path through the adapted graph. In both town implementations, the core objective of this stage is the same. The objective is to generate a route that is consistent with the constraints already embedded in the planning space and therefore suitable for execution under the active scenario conditions. The planner does not operate on the nominal road network independently of the scenario. Instead, it searches on the graph that has already been filtered or adapted according to the run-specific navigation conditions.

In Town02, this pipeline is realized in a comparatively direct form. After the graph has been filtered according to the selected scenario mode, the route is computed between the graph nodes corresponding to the chosen start and destination. The resulting path is then converted into a sequence of route elements suitable for execution by the ego vehicle. This implementation makes Town02 the clearest example of the route generation mechanism. Which is the scenario dependent graph adaptation followed by route computation on the feasible planning space.

In Town01, the same route generation principle is preserved, although the implementation includes a richer scenario realization. A baseline route can be computed on the unconstrained graph, while the active route is generated on the constrained scenario specific graph. This makes it possible to distinguish between nominal and adapted route behavior within the same town. The key implementation point, however, remains unchanged. Which is the route assigned to the ego vehicle is the route produced from the graph representation dynamically that corresponds to the active scenario, not a static route reused independently of the selected conditions.

Once the route has been computed, it must be transformed into an executable representation for use inside CARLA. In the implemented system, the graph based route is not left as a purely abstract path description. Instead, it is converted into a route format that can be assigned to the simulator's vehicle agent. This conversion stage is essential because it links planning output to simulator execution. Without this the planning process would remain disconnected from observable vehicle be-

havior. The execution pipeline therefore includes not only route generation itself, but also the preparation of the generated path for route following in the simulation environment.

Execution Pipeline

The execution stage begins after the ego vehicle has been spawned and the route has been assigned. From that point onward, the vehicle follows the computed route inside CARLA under the conditions of the active scenario. This execution stage is important because it ensures that the implemented framework does not stop at route computation alone. Instead, it realizes the complete path from planning space preparation to actual route following behavior by the ego vehicle. This is a defining implementation characteristic of the system, since the thesis is concerned not only with generating routes under scenario specific constraints, but also with observing how those generated routes perform during simulation runs.

A further implementation feature of this pipeline is that it remains compatible with the run specific interpretation of dynamic navigation developed throughout the thesis. The route is not repeatedly redefined at arbitrary points during the same drive. Rather, the implementation computes the route for the active scenario before execution begins and then carries that route into the execution stage. If the conditions of a later run differ, a new graph and a new route can be generated for that new run. In this way, the route generation and execution pipeline remains fully aligned with the scope of the work and does not overstate the system as a continuously online re planning architecture.

From a software perspective, this pipeline is especially important because it unifies the main functions of the implemented framework. That are scenario specific preparation, graph based route generation, executable route transfer, and simulated route following. It is therefore one of the most central sections of the implementation chapter, since it describes how the navigation mechanism developed in the thesis becomes observable as actual ego vehicle behavior in the simulation environment

The implemented route-generation and execution logic is summarized in the algorithm box bellow.

Algorithm: Route generation and execution pipeline

Input: planning graph G, start location S, destination D
Output: executed scenario-specific navigation route

map S and D to valid graph nodes
compute a feasible route on G between the selected nodes
convert the resulting graph path into an executable route format
spawn the ego vehicle in the selected scenario
assign the generated route to the vehicle agent
execute the route in CARLA
monitor route following behavior during the simulation run
return executed route behavior

The algorithm box above summarizes the implemented route generation for navigation and execution process used in the developed CARLA framework. The procedure begins by associating the selected start and destination positions with valid nodes in the environment aware planning graph. On this basis, a feasible route is computed under the constraints of the active scenario and then transformed into an executable route representation suitable for the simulator. After the ego vehicle is spawned, the generated route is assigned to the vehicle agent and executed inside CARLA. In this way, the algorithm captures the complete transition from graph-based route planning to observable route following behavior in the implemented navigation system.

4.6 Visualization and Logging

A central part of this support layer is the visualization of route execution in the simulation environment. In both town implementations, the system uses a split-screen pygame window that presents the current run from two complementary perspectives. The first view is a third-person camera that follows the ego vehicle during motion and provides a direct observation of local driving behavior. The second view is a fixed bird's-eye perspective that provides a static overview of the scenario, including the overall route geometry and the spatial relation between start position, destination, and surrounding road structure. This dual view arrangement allows route execution to be observed both locally and globally during the same run.

The implemented visualization is not limited to raw camera output. The generated route is projected visually onto the displayed views so that the intended path of the ego vehicle can be inspected during execution. In the bird's-eye view, the route is shown as a colored line that represents the path the vehicle is expected to follow, together with explicit markers for the start and end locations. In the vehicle following view, the route can also be visualized in relation to the current movement

of the ego vehicle. This makes it possible to compare the planned route with the actual vehicle behavior and to assess whether the navigation mechanism is producing the expected route following outcome under the active scenario conditions.

This visualization design is particularly useful in constrained scenarios. The bird's-eye view supports interpretation of the route at the level of the full scenario layout, while the third-person view makes it easier to observe how the vehicle progresses along the route in the presence of traffic, static blockers, or other scenario specific conditions. As a result, the visualization does not merely display the simulation for demonstration purposes. It functions as an implementation tool for route verification, qualitative assessment, and execution monitoring.

In addition to the route overlay, the pygame window also displays execution related information directly on top of the simulation views. These on-screen overlays provide compact status information during the run, including route related progress, vehicle state, and scenario context. As a result, the visualization interface does not only support geometric observation of the planned path, but also provides immediate feedback about the current execution state of the navigation process. This improves the interpretability of the simulation run and makes the visual monitoring of route execution more informative.

Alongside visualization, the framework also includes CSV-based logging of execution data during the simulation run. This logging mechanism records time dependent information that reflects the state and behavior of the ego vehicle while the route is being executed. The purpose of this logging is to preserve essential run data in a structured form so that route execution can later be inspected beyond real-time visualization alone. In the implemented system, the recorded values include simulation time, vehicle position, yaw angle, speed, lane information, road information, control inputs, goal-reached status, and cumulative driven distance. These data provide a useful basis for analyzing route progression and execution behavior across different runs and scenario modes.

The logging mechanism is important because visual observation alone is not sufficient for systematic evaluation. While the visual output helps to understand route behavior qualitatively, the CSV log preserves the execution history in a reproducible form that can later be processed for quantitative inspection. This gives the framework two complementary observation channels. These are direct visual interpretation during the run and structured data recording for later analysis. Together, these components support the later evaluation chapter by making both the route behavior and the execution state of the vehicle accessible after simulation.

From an implementation perspective, visualization and logging also serve as debugging and validation tools. They make it easier to verify that the generated route has been transferred correctly to the ego vehicle, that the vehicle progresses along the expected route, and that the scenario-specific navigation constraints lead to observable differences in route behavior. In this sense, these components are not only supportive additions to the implementation, but practical parts of the overall framework that contributed to verifying and analyzing the developed navigation mechanism during experimentation.

4.7 Implementation Summary

This chapter presented the implementation of the proposed navigation framework in the CARLA simulation environment. While the previous chapter defined the methodological basis of the approach, the present chapter showed how that methodology was realized as an executable system in Python and integrated into CARLA. The implementation was organized as a structured pipeline that connects scenario realization, map based planning space construction, environment aware constraint handling, route generation, route execution, visualization, and logging within a single simulation framework.

A central implementation result of this chapter is the realization of the same navigation mechanism across two different CARLA towns. Town02 served as the primary implementation environment in which the environment aware route adaptation logic was first established, while Town01 was used as an additional implementation environment to examine whether the same planning principle could also be realized under a different town layout. In both cases, the key implementation idea remained unchanged which is the route followed by the ego vehicle was generated only after the planning space had been adapted according to the active scenario conditions. This ensured that route execution was directly linked to the effective road availability of the corresponding run rather than to the nominal map alone.

The chapter also showed that the framework was implemented as more than a route planning prototype. In addition to graph construction and route computation, the system includes execution support through actor setup, a split-screen visualization interface, route overlays, on-screen status information, and CSV based logging of run-time data. These elements are important because they make the generated navigation behavior observable and analyzable during simulation, thereby providing the practical basis for the evaluation presented in the next chapter.

Taken together, the implementation chapter established how the proposed dynamic navigation function concept was transformed into a working CARLA system. The next chapter builds on this implementation and presents the results and evaluation of the framework under different scenario conditions in Town02 and Town01.

5 Results and Evaluation

This chapter presents the results of the implemented navigation framework and evaluates its behavior under different scenario conditions in the CARLA simulation environment. The purpose of the chapter is to determine whether the proposed dynamic navigation mechanism operates reliably under constrained navigation settings. Also, whether the resulting route behavior remains consistent with the objectives defined earlier in the thesis.

The evaluation is based on the implemented framework in Town02 and Town01 and considers multiple scenario modes in each town. In both environments, the system was executed repeatedly under varying road constraints in order to observe how the route-generation mechanism responds to changed navigation conditions. The analysis therefore focuses not only on whether a valid route can be generated, but also on how the resulting route affects execution-related properties such as travel distance, travel time, speed behavior, and overall route completion.

In addition to town specific evaluation, the chapter also evaluates the applicability of the framework across different map layouts. Since the same navigation principle was implemented in both Town02 and Town01, the evaluation can assess whether the proposed mechanism remains effective beyond one single simulation environment. In this way, the chapter does not only report isolated run outcomes, but uses the collected data to evaluate the practical behavior of the developed framework under repeated and scenario-dependent conditions.

The chapter is structured as follows. It first introduces the evaluation setup and the metrics used for analysis. It then presents the results for Town02 and Town01 separately before discussing the observations in a broader comparative context.

5.1 Evaluation Setup

The evaluation presented in this chapter was designed to assess the practical behavior of the implemented navigation framework under different scenario-dependent conditions in the CARLA simulation environment. Since the objective of the thesis is not to develop a continuously replanning online navigation architecture, but rather a route-generation mechanism that adapts to the environmental conditions of each simulation run before execution begins, the evaluation focuses on whether this mechanism remains effective, robust, and computationally manageable across nominal and constrained scenarios. The central question is therefore not only whether a route can be generated, but whether the generated route remains suitable for execution under the active scenario conditions and whether this behavior can be observed

consistently across different simulation settings.

The evaluation was conducted in two CARLA towns, Town02 and Town01. Town02 serves as the primary implementation environment in which the environment aware navigation mechanism was first realized. Town01 was then used as an additional implementation environment in order to examine whether the same navigation principle could also be applied under a different town layout. The purpose of using both towns is therefore not to compare them as competing benchmark cases, but to evaluate whether the same implemented mechanism remains applicable across more than one simulated urban environment. This is important for the thesis because it strengthens the claim that the proposed framework is not limited to a single town specific configuration.

In both towns, the framework was evaluated under multiple scenario modes that represent different levels of route constraint. These scenarios range from a nominal case without relevant blocked road conditions to progressively more constrained cases in which blocked or undesired road segments alter the effective planning space. Each scenario mode was executed repeatedly in order to reduce the dependence of the analysis on isolated individual runs. The evaluation is therefore based on both detailed run logs and run summary data. The timestep level logs preserve dynamic execution information such as position, speed, control values, and cumulative driven distance, while the run summaries provide compact indicators such as route completion, route length, travel time, planning time, graph size after filtering, and the number of blockers or constraint points associated with the scenario.

The analysis is organized around four main evaluation dimensions. The first dimension is route feasibility and completion, which examines whether a valid path could be generated under the selected conditions and whether the destination was successfully reached. The second dimension is navigation efficiency, represented by quantities such as planned route length, executed distance, travel time, and speed related measures. The third dimension is computational performance, which considers planning time and graph related characteristics in order to assess whether the route adaptation mechanism remains practically manageable under increasingly constrained conditions. The fourth dimension is execution robustness, which is reflected through indicators such as recovery behavior where relevant. Taken together, these dimensions provide a structured basis for examining both the effectiveness and the practical behavior of the implemented navigation framework.

The chapter combines quantitative and qualitative evaluation. The quantitative analysis is based on the recorded summary metrics and repeated runs in both towns. The qualitative analysis is based on the observed route behavior during simulation, supported by the implemented split-screen visualization interface, route overlays, and representative execution screenshots. This is useful because the developed framework is not only a numerical planning system but also a visible route execution system whose adaptation behavior can be interpreted directly in the simulator. The combination of both perspectives therefore makes it possible to evaluate not only how the mechanism performs numerically, but also how the resulting route behavior appears under different navigation conditions.

The overall evaluation setup used for the experimental analysis in Town02 and Town01 is summarized in Table 5.1.

Town	Scenario Mode	Runs	Scenario Description
Town02	Mode 1	3	Nominal navigation case without blocked-road constraints.
Town02	Mode 2	3	Constrained case with road blockage caused by a construction site, affecting the feasible planning space.
Town02	Mode 3	3	More strongly constrained case with additional blocked-road conditions caused by an accident.
Town01	Mode 1	3	Nominal navigation case in the second town environment.
Town01	Mode 2	3	Constrained case in Town01 with traffic congestion affecting the feasible route.
Town01	Mode 3	3	More complex constrained case with stronger route restriction effects due to heavy traffic congestion.

Table 5.1: Overview of the evaluated scenario configurations in Town02 and Town01

Table 5.1 summarizes the scenario configurations used for the evaluation of the implemented navigation framework. In both Town02 and Town01, the analysis was conducted under three scenario modes representing different levels of route constraint. Mode 1 corresponds to the nominal case without relevant blocked road conditions, while Modes 2 and 3 introduce increasingly constrained navigation settings that alter the effective planning space before route generation. Each mode was executed three times in order to reduce the dependence of the evaluation on isolated runs and to provide a more stable basis for the analysis of route completion, navigation efficiency, computational performance, and execution robustness.

5.2 Evaluation of the Navigation Framework in Town02

Town02 serves as the primary implementation environment of the proposed navigation framework and therefore forms the starting point of the evaluation. It is in this town that the environment aware route generation mechanism was first realized and tested under nominal as well as progressively constrained scenario conditions. The Town02 evaluation is therefore particularly important for assessing whether the implemented framework behaves consistently with the central objective of the thesis, namely to generate a route that reflects the environmental conditions of the active simulation run before execution begins.

The evaluation in Town02 was conducted across three scenario modes representing different levels of route restriction. Mode 1 corresponds to the nominal case without blocked road constraints, while Modes 2 and 3 introduce increasingly constrained navigation conditions through blocked or undesired road regions. In the implemented Town02 system, these constraints are realized through scenario dependent actor placement and graph filtering, so that route generation is carried out on a planning graph that already reflects the feasible road space of the selected scenario. This design is directly visible in the Town02 implementation, where scenario mode, blocked road coordinates, filtered graph construction, route generation, split-screen visualization, and CSV-based execution logging are integrated into a single CARLA based pipeline. The main quantitative results obtained for Town02 are summarized in table 5.2 and 5.3 .

Mode	**Runs**	**Path Found Rate**	**Avg. Planned Route Length (m)**	**Avg. Travel Time (s)**
Mode 1	3	100.0%	199.82	41.08
Mode 2	3	100.0%	252.23	69.55
Mode 3	3	100.0%	330.88	49.50

Table 5.2: Planning feasibility and main navigation performance results for the Town02 evaluation

Mode	**Avg. Executed Distance (m)**	**Avg. Speed (m/s)**	**Avg. Planning Time (s)**
Mode 1	170.42	3.951	0.1069
Mode 2	228.19	3.680	0.0901
Mode 3	321.05	6.239	0.1042

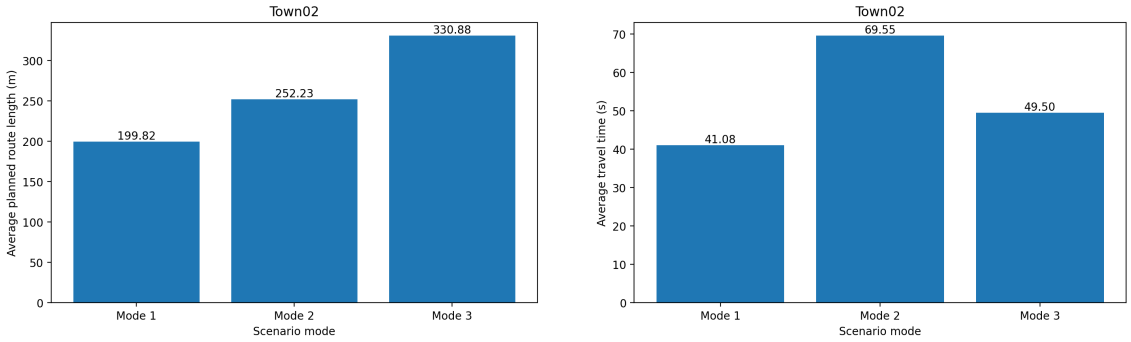
Table 5.3: Execution related and computational results for the Town02 evaluation

As shown in Table 5.2 and 5.3, a valid path could be generated in all evaluated Town02 runs across all three scenario modes. This indicates that the proposed planning mechanism remained feasible not only in the nominal case, but also under increasingly constrained conditions. In this sense, the path-found rate provides the most direct planning-related evidence that the environment aware route generation approach remained operational when the effective planning space was altered through blocked or undesired road regions.

At the same time, the quantitative results show that the increasing scenario constraints had a visible influence on navigation efficiency. The average planned route length increased from 199.82 m in Mode 1 to 252.23 m in Mode 2 and reached 330.88 m in Mode 3. This progression indicates that the route generation mechanism responded to the increasing restriction of the planning space by selecting longer feasible routes when necessary. A similar tendency can be observed in the executed travel distance, which also increases from the nominal case to the constrained cases. These changes are consistent with the intended behavior of the framework, since route adaptation under blocked road conditions is expected to alter the geometry of the resulting path.

The travel-time values further support this interpretation. The average travel time increases markedly from Mode 1 to Mode 2, while Mode 3 also shows a comparatively high execution duration despite its different average speed profile. These observations suggest that the constrained scenarios did not merely affect the abstract planning graph, but also had a measurable impact on the practical route following behavior of the ego vehicle during execution. The effect of scenario dependent adaptation is therefore visible both at the planning level and at the execution level.

Planning time remains low across all three modes, with values close to one tenth of a second. This is an important observation because it indicates that the graph adaptation and route-generation mechanism remained computationally lightweight even under the more constrained scenarios. In the context of the present thesis, this supports the claim that the proposed approach is not only functionally effective, but also practically manageable within the implemented CARLA workflow.



(a) Average planned route length

(b) Average travel time

Figure 5.1: Average planned route length and average travel time across the evaluated scenario modes in Town02.

Figure 5.1 shows that the increasing scenario constraints in Town02 led to a clear rise in average planned route length from Mode 1 to Mode 3. This indicates that the progressive restriction of the feasible planning space required the framework to generate longer adapted routes under more constrained conditions. A related effect can also be observed in the travel time values. Although the increase is not perfectly monotonic across all modes, the constrained scenarios show a clear deviation from the nominal case, which suggests that the scenario-dependent route adaptation influenced not only the geometry of the generated route but also the practical execution behavior during the simulation runs.

Mode	Avg. Blockers	Avg. Constraint Points	Avg. Graph Nodes	Avg. Stuck Events	Avg. Recovery Events
Mode 1	0	0	1342	0	0
Mode 2	12	23	1322	0	0
Mode 3	19	50	1295	0	0

Table 5.4: Scenario structure and robustness metrics for the Town02 evaluation

Table 5.4 provides additional context for interpreting these results. The number of blockers and constraint points increases substantially from Mode 1 to Mode 3, while the graph size after filtering decreases from 1342 nodes in the nominal case to 1295 nodes in the most constrained case. This confirms that the effective planning space was progressively reduced across the scenario modes. At the same time, no stuck events or reverse-recovery events were recorded in the evaluated Town02 runs. This suggests that the generated routes remained sufficiently stable for execution even when the route geometry was altered by stronger scenario constraints.



Figure 5.2: Representative split-screen execution view of the implemented navigation framework in Town02

In addition to the quantitative results, the Town02 evaluation also provided a clear qualitative indication of the dynamic navigation behavior of the implemented framework. Through the split-screen visualization interface, the generated routes could be observed both from the vehicle following third person view and from the fixed bird's eye perspective with route overlay. In the constrained scenarios, the bird's eye view made it possible to observe that the generated route was redirected away from the blocked or undesired road regions and remained aligned with the feasible road corridor of the active scenario. This visual behavior is consistent with the quantitative increase in route length and supports the interpretation that the environment aware planning mechanism did not merely alter abstract graph properties, but produced visibly different route geometries under constrained conditions.

5.3 Evaluation of the Navigation Framework in Town01

Town01 was used as the additional implementation environment in order to examine whether the proposed navigation mechanism remains applicable beyond its primary realization in Town02. The purpose of this section is therefore not to compare Town01 against Town02 as if both towns were independent benchmark systems, but rather to evaluate whether the same environment aware navigation principle could also be realized successfully in a second CARLA town with a different road layout. In this sense, the Town01 results are important because they support the claim that the implemented navigation framework is not limited to a single map-specific configuration.

As in Town02, the Town01 evaluation was conducted across three scenario modes representing increasing levels of route restriction. Mode 1 corresponds to the nominal case without relevant route constraints, while Modes 2 and 3 introduce scenario dependent traffic or route restrictions that alter the feasible planning space before route generation begins. The evaluation is based on repeated runs and summarized through the recorded planning and execution metrics. In Town01, these results are particularly useful because they show whether the implemented mechanism remains stable when transferred to a different map topology and a somewhat different constrained scenario context.

The main Town01 evaluation results are presented in Tables 5.5 and 5.6.

Mode	Runs	Path Found Rate	Avg. Planned Route Length (m)	Avg. Travel Time (s)
Mode 1	3	100.0%	210.50	30.33
Mode 2	3	100.0%	217.61	29.52
Mode 3	3	100.0%	416.63	99.54

Table 5.5: Planning feasibility and main navigation performance results for the Town01 evaluation

Table 5.5 summarizes the planning feasibility and the most central route level performance indicators for Town01. A valid path was found in all evaluated runs across all three scenario modes, which indicates that the proposed navigation mechanism remained operational in the second town environment as well. The results further show that the route length remains relatively similar between Modes 1 and 2, while Mode 3 produces a substantially longer route. A similar pattern can be observed in the travel time values, where the first two modes remain close to one another, but the most constrained scenario leads to a markedly higher execution duration. This suggests that the stronger scenario restrictions in Mode 3 had a clearly visible effect on both route geometry and practical navigation behavior.

Table 5.6 presents additional execution related and computational indicators for the Town01 evaluation. The increase in executed distance from Mode 1 to Mode 3 is consistent with the increase in planned route length and supports the interpretation that the stronger scenario restrictions in Mode 3 required a substantially more

Mode	Avg. Executed Distance (m)	Avg. Speed (m/s)	Avg. Planning Time (s)
Mode 1	198.94	5.922	0.3443
Mode 2	214.47	5.790	0.5338
Mode 3	409.38	3.902	0.5624

Table 5.6: Execution-related and computational results for the Town01 evaluation

extensive adapted route. In contrast, the average vehicle speed remains relatively high in the first two modes and decreases clearly only in Mode 3, which indicates that the more strongly constrained route also influenced the practical driving behavior during execution. Planning time remains below one second in all evaluated cases, although it is somewhat higher than in Town02. This suggests that the environment aware navigation mechanism remained computationally manageable in Town01, even though the graph structure and route geometry differ from the primary implementation environment.

A quantitative interpretation of the Town01 results shows that the navigation mechanism remained feasible in all evaluated runs. As shown in Table 5.5, the path found rate remained at 100% across all three scenario modes. This is an important result because it indicates that the route generation logic remained operational when applied to a second CARLA town with a different road structure. The framework was therefore able to construct a valid route not only in the nominal case, but also under scenario-dependent route restrictions in Town01.

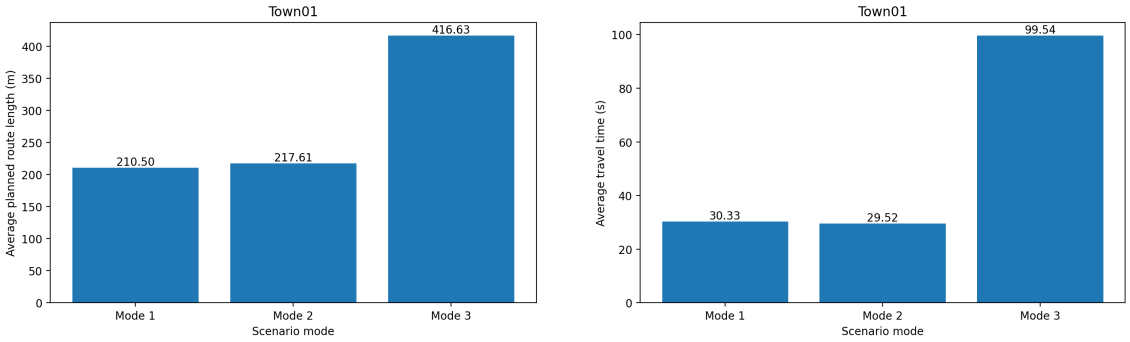
At the same time, the numerical results reveal that the effect of the scenario restrictions was not uniform across all modes. Modes 1 and 2 remain relatively close to one another in terms of route length, travel time, and average speed. The average planned route length increases only slightly from 210.50 m in Mode 1 to 217.61 m in Mode 2, while the average travel time changes only marginally from 30.33 s to 29.52 s. The executed distance and average speed also remain close, with 198.94 m and 5.922 m/s in Mode 1 compared with 214.47 m and 5.790 m/s in Mode 2. This suggests that the constrained conditions in Mode 2 influenced the route generation process, but did not produce a major deviation from the nominal navigation behavior.

A clearly different picture emerges in Mode 3. Here, the average planned route length rises to 416.63 m, the executed distance increases to 409.38 m, and the average travel time reaches 99.54 s. At the same time, the average speed decreases to 3.902 m/s. These values indicate that the more strongly constrained Town01 scenario required a substantially more extensive route adaptation and also influenced the practical execution behavior of the ego vehicle much more strongly than the lighter constrained case. The results therefore suggest that the implemented mechanism remained functional in Town01, while still producing visibly different route outcomes under stronger scenario restrictions.

The planning-time values further support the interpretation that the proposed mechanism remained computationally manageable in the second town environment. The average planning time increased from 0.3443 s in Mode 1 to 0.5338 s in Mode 2

and 0.5624 s in Mode 3. Although these values are somewhat higher than those observed in Town02, they remain low in absolute terms and do not suggest a problematic computational burden. In the context of this thesis, this is important because it indicates that the same environment aware navigation principle could be realized in a second CARLA town without turning route generation into a computational bottleneck.

The influence of the Town01 scenario modes on route geometry and execution time is illustrated in Figure 5.3.



(a) Average planned route length

(b) Average travel time

Figure 5.3: Average planned route length and average travel time across the evaluated scenario modes in Town01.

Figure 5.3 shows that the average planned route length remains nearly unchanged between Modes 1 and 2, increasing only slightly from 210.50,m to 217.61,m. In contrast, Mode 3 leads to a substantial increase to 416.63,m, which indicates that the stronger scenario restrictions in this mode required a significantly longer adapted route. A similar pattern can be observed in the travel-time values. While the average travel time remains close in Modes 1 and 2, at 30.33,s and 29.52,s respectively, it rises sharply to 99.54,s in Mode 3. This confirms that the most strongly constrained Town01 scenario did not only influence the abstract planning graph, but also produced a clear execution-level effect during the route-following process

Additional scenario related and robustness-oriented metrics for Town01 are listed in Table 5.7.

Mode	Avg. Blockers	Avg. Constraint Points	Avg. Graph Nodes After Filtering	Avg. Stuck Events	Avg. Reverse Recovery Events
Mode 1	0	0	3031	0	0
Mode 2	10	76	3004	0	0
Mode 3	14	80	3004	0	0

Table 5.7: Scenario-structure and robustness metrics for the Town01 evaluation

Table 5.7 provides additional context for the interpretation of the Town01 results by describing the structure of the evaluated scenarios and the observed robustness of execution. The number of blockers and constraint points increases substantially from the nominal case to the constrained modes, while the graph size after filtering

is reduced from 3031 nodes in Mode 1 to 3004 nodes in Modes 2 and 3. Although this reduction is smaller in absolute terms than the route length increase observed in Mode 3, it still confirms that the feasible planning space was altered systematically by the scenario conditions. At the same time, no stuck or reverse recovery events were recorded in the evaluated Town01 runs, which suggests that the generated routes remained stable enough for execution in the selected scenarios.



Figure 5.4: Representative split-screen execution view of the implemented navigation framework in Town01

A short qualitative observation also supports the quantitative interpretation. Through the split-screen visualization interface, the route behavior in Town01 could be observed both from the third-person chase view and from the fixed bird's-eye perspective with route overlay. In the more strongly constrained scenarios, the bird's-eye view showed that the generated route deviated visibly from the nominal path and followed a longer feasible corridor through the scenario. This visual behavior is consistent with the substantial increase in planned route length and travel time observed in Mode 3 and therefore supports the interpretation that the framework remained environment aware not only at the level of graph adaptation, but also in the visible route geometry during execution.

5.4 Cross-Town Applicability of the Navigation Framework

The purpose of the cross-town analysis is not to compare Town02 and Town01 as competing benchmark environments, but to examine whether the same environment-aware navigation mechanism remained applicable across different CARLA town layouts. This distinction is important because the thesis does not propose two separate navigation systems. Instead, the same planning principle was first realized in Town02 and then transferred to Town01 in order to assess whether the framework could be implemented successfully beyond its primary realization environment. The relevance of this section therefore lies in the applicability of the mechanism across two different simulated urban contexts rather than in a direct performance ranking between the towns.

The results presented in Sections 5.2 and 5.3 show that a valid path could be generated in all evaluated scenario modes in both towns. From the perspective of route feasibility, this is the most important common result of the chapter. It indicates that the graph based route generation mechanism remained operational not only in the nominal cases, but also under progressively constrained conditions in both Town02 and Town01. In this sense, the cross-town analysis supports the claim that the proposed navigation framework is not restricted to a single CARLA map, but can be realized in more than one town while preserving the same environment-aware planning logic.

A compact cross-town overview of the main evaluation metrics is provided in Table 5.8.

Town	Mode	Path Found Rate	Planned Route Length	Travel Time	Planning Time
Town02	Mode 1	100.0%	199.82	41.08	0.1069
Town02	Mode 2	100.0%	252.23	69.55	0.0901
Town02	Mode 3	100.0%	330.88	49.50	0.1042
Town01	Mode 1	100.0%	210.50	30.33	0.3443
Town01	Mode 2	100.0%	217.61	29.52	0.5338
Town01	Mode 3	100.0%	416.63	99.54	0.5624

Table 5.8: Cross-town applicability of the implemented navigation framework across Town02 and Town01

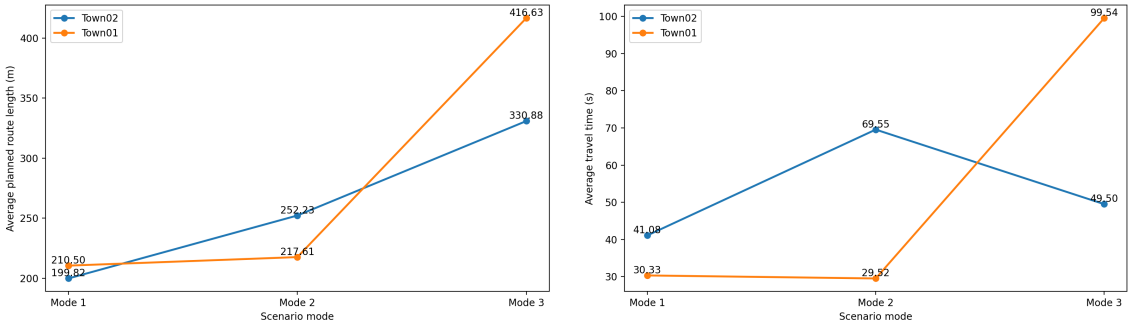
Table 5.8 summarizes the main mode wise indicators for both towns. The most important common result is that the path-found rate remained at 100% across all evaluated scenarios in both Town02 and Town01. This indicates that the navigation mechanism remained operational in both environments and was able to produce valid routes not only in the nominal case, but also under progressively constrained conditions. In this sense, the table supports the claim that the developed navigation framework is transferable across different CARLA towns.

At the same time, the concrete numerical effects of route adaptation differ between the two towns. In Town02, the average planned route length increases progressively

from 199.82 m in Mode 1 to 252.23 m in Mode 2 and 330.88 m in Mode 3. In Town01, by contrast, Modes 1 and 2 remain relatively close to one another, with average planned route lengths of 210.50 m and 217.61 m respectively, while Mode 3 shows a much larger increase to 416.63 m. This indicates that the environment aware navigation mechanism remained functional in both towns, but that the extent of route adaptation depended on the specific map structure and on the concrete realization of the constrained scenarios.

A similar observation can be made with respect to travel time. In Town02, the constrained modes differ clearly from the nominal case, although not in a strictly monotonic manner. In Town01, the first two modes remain close to one another, whereas Mode 3 leads to a much larger travel time of 99.54 s. These values suggest that the practical execution behavior of the ego vehicle was influenced not only by route length, but also by town specific route geometry and the local conditions encountered during the simulated runs. The average planning times also remain low in both towns, although they are somewhat higher in Town01 than in Town02. This indicates that the route generation mechanism remained computationally lightweight even when transferred to a second town with a different graph structure.

The relative behavior of the route lengths nad travel time across towns is illustrated in Figure 5.5.



(a) Average planned route length

(b) Average travel time

Figure 5.5: Cross-town comparison of average planned route length and average travel time across the evaluated scenario modes in Town02 and Town01..

Figure 5.5 illustrates the cross-town behavior of the main route-level and execution-level indicators across the evaluated scenario modes. Figure 5.5 (a) shows that the average planned route length increases in both towns as the scenario conditions become more restrictive, although the pattern differs between the two environments. In Town02, the increase is more gradual from Mode 1 to Mode 3, whereas in Town01 the route length changes only slightly from Mode 1 to Mode 2 and rises sharply in Mode 3. This indicates that the same navigation planning principle remained applicable in both towns, while the extent of route adaptation depended on the underlying map structure and on the realization of the constrained scenario. Figure 5.5 (b) presents the corresponding travel time behavior and shows that the execution

level effects were not identical to the route length trends. In Town02, travel time increases from Mode 1 to Mode 2 and then decreases in Mode 3, while in Town01 the first two modes remain close and Mode 3 produces a substantial increase. This suggests that the practical execution behavior was influenced not only by the length of the generated route, but also by town-specific route geometry, local traffic interaction, and scenario dependent driving conditions. Taken together, the figure supports the conclusion that the proposed framework remained functional and applicable across both towns, even though the quantitative effects of route adaptation were not numerically identical.

The scenario structure metrics further support the applicability of the framework in both environments. The constrained modes in both towns show an increase in blockers or constraint points together with a reduction in the filtered planning graph. At the same time, no stuck or reverse recovery events were recorded in the summarized runs used for the main analysis. This suggests that the generated routes remained sufficiently stable for execution in both environments. Even though the scale of the graph and the route geometry differ between Town02 and Town01, the same underlying planning principle produced feasible and executable routes in both cases.

Taken together, the cross-town evaluation supports the conclusion that the proposed navigation framework remained applicable across the two CARLA towns considered in this thesis. The route adaptation mechanism preserved its core functionality in both environments, while the concrete numerical effects varied according to map structure and scenario realization. This is a meaningful result because it shows that the developed framework is not only effective in its primary realization environment, but can also be transferred to another town without changing the underlying navigation concept.

5.5 Interpretation of the Evaluation Results

The results presented in this chapter show that the proposed navigation framework remained operational in both Town02 and Town01 under nominal as well as constrained scenario conditions. In all evaluated modes, a valid path could be generated on the navigation planning graph. This indicates that the environment aware route generation mechanism was able to preserve its basic planning functionality even when the feasible route space was altered by blocked or undesired road regions.

A second central observation is that stronger scenario constraints generally led to longer routes and, in several cases, to longer travel times. This behavior is consistent with the intended logic of the framework. Since the route is generated after the planning graph has been adapted to the active scenario, increased route restriction is expected to alter the geometry of the resulting path. The observed increase in planned route length is therefore an indication that the mechanism responded to the scenario conditions rather than relying on a fixed nominal route.

At the same time, the results also show that the magnitude of these effects differed

between Town02 and Town01. This suggests that the practical outcome of route adaptation depends not only on the planning mechanism itself, but also on the structure of the town and on the way the constrained scenario is realized in the environment. The cross-town analysis therefore supports the conclusion that the same planning principle remained applicable in both towns, while its quantitative effects were shaped by town specific route geometry and scenario context.

A further important result is that the planning times remained low in both environments. Even in the more constrained scenarios, route generation stayed computationally lightweight in absolute terms. This supports the idea that the proposed framework achieved environment-aware route adaptation without requiring a heavy continuously replanning architecture. Within the scope of this thesis, the evaluation therefore indicates that the implemented mechanism was both functionally effective and practically manageable.

5.6 Scope and Limitations of the Evaluation

The evaluation presented in this chapter provides meaningful evidence for the applicability of the proposed navigation framework within the scope of the implemented simulation setup. At the same time, the results should be interpreted in relation to the conditions under which they were obtained. The present analysis is based on two CARLA towns, a limited number of scenario modes, and repeated simulation runs under manually structured route-restriction conditions. These choices are sufficient to evaluate whether the developed mechanism remains functional under nominal and constrained settings and whether it can be realized in more than one town layout. However, they do not represent an exhaustive validation across a broad range of map structures or scenario classes.

A first scope related limitation therefore concerns the simulation environments themselves. Although the use of Town02 and Town01 strengthens the evaluation beyond a single-town demonstration, the results remain tied to these two selected CARLA maps. The findings show that the same environment aware navigation principle can be applied successfully in both environments, but they should not be interpreted as evidence that the exact same behavior would necessarily remain unchanged across all possible CARLA towns or more complex map topologies.

A second limitation lies in the scenario design. The evaluated scenarios were deliberately structured to represent nominal, constrained, and more strongly constrained route conditions. This allows the effects of route space restriction to be examined in a controlled manner, but it also means that the evaluation reflects a specific set of blocked road and traffic related cases rather than the full variety of situations that could arise in broader autonomous driving studies. The presented results are therefore best understood as a focused evaluation of the proposed mechanism under representative constrained navigation settings rather than as a comprehensive benchmark over all possible environmental variations.

A further scope related point concerns the distinction between planning behavior

and full execution outcome. In this thesis, the primary focus of the evaluation is the behavior of the route generation and navigation mechanism, including path feasibility, route adaptation, and the measurable consequences of scenario dependent planning on route length, travel time, and execution related indicators. Individual simulation runs may also be affected by traffic interaction or other local execution side effects that are not always attributable solely to the planning mechanism itself. For this reason, the results are most appropriately interpreted as evidence for the applicability and practical behavior of the proposed planning framework rather than as a complete end-to-end benchmark of autonomous-driving performance.

The number of repeated runs per scenario also defines the scope of the analysis. Repeated runs improve the robustness of the evaluation compared with single run observations and make it possible to identify meaningful tendencies across the scenario modes. At the same time, the sample size remains limited, so the evaluation is not intended as a statistically extensive study of run-to-run variation. Instead, it provides a structured simulation based assessment that is appropriate for the scope of a Master's thesis and sufficient for identifying the main trends relevant to the developed framework.

Finally, the evaluation is limited to a navigation approach that adapts the route before execution begins. As emphasized throughout the thesis, the proposed framework is not a continuously re-planning online system. This does not weaken the relevance of the results, but it defines the range of conclusions that can be drawn from them. The findings presented in this chapter therefore apply to a scenario dependent pre-execution route-generation mechanism and should not be interpreted as evidence for real-time reactive re-planning under continuously changing road conditions.

Taken together, these aspects define the scope within which the evaluation results should be understood. Within this scope, the chapter provides clear evidence that the proposed navigation framework remained applicable in two CARLA towns and responded meaningfully to scenario dependent route restrictions.

6 Discussion and Future Work

This chapter discusses the results of the thesis in a broader context and reflects on the significance of the proposed navigation framework beyond the direct evaluation results presented in Chapter 5. While the previous chapter focused on the measured behavior of the implemented system under different scenario conditions in Town02 and Town01, the present chapter considers what these results imply for the overall contribution of the work, how the developed framework should be interpreted within the broader context of simulation-based navigation research, and which limitations and future extensions arise from the chosen system design.

The discussion is particularly important because the contribution of this thesis does not lie in the development of a full autonomous driving stack or a continuously replanning online navigation architecture. Instead, the work addresses a more focused problem thet is, how route generation in CARLA can be made responsive to the environmental conditions of a given simulation run before execution begins. The significance of the developed framework must therefore be assessed in relation to this defined scope. From this perspective, the key question is not whether the system solves all aspects of dynamic autonomous navigation, but whether it provides a meaningful, technically interpretable, and experimentally useful route generation and navigation mechanism for scenario dependent simulation settings.

A second purpose of this chapter is to distinguish between the concrete evaluation findings and the broader implications of the work. Chapter 5 has already shown that the framework remained operational across two CARLA towns and produced visibly and quantitatively different route behavior under constrained conditions. The present chapter builds on that basis by discussing what these findings mean for the positioning of the thesis, where the developed approach offers value, where its methodological boundaries remain, and how it could be extended in future research.

The chapter is structured as follows. It first discusses the developed framework and its contribution in relation to the objectives of the thesis. It then addresses the limitations of the proposed approach at the system level before outlining possible directions for future work.

6.1 Discussion

The central objective of this thesis was to develop a navigation framework in CARLA in which route generation reflects the environmental conditions of the active simulation run before execution begins. In this sense, the work addressed a more specific problem than that considered in many broader autonomous driving architectures. The contribution was not intended to lie in the development of a complete perception planning control stack, nor in a continuously re-planning online navigation system. Instead, the thesis focused on the question of how scenario dependent environmental constraints can be integrated into route level planning in a structured, interpretable, and computationally manageable way. From this perspective, the results of the thesis show that the developed framework fulfilled its intended purpose within the defined scope of the work.

A first important discussion point concerns the methodological relevance of linking scenario configuration directly to route generation. In many simulation workflows, traffic conditions and environmental changes are used to evaluate the behavior of a vehicle during execution, while the route itself remains based on a nominal road network representation. The framework developed in this thesis takes a different position. It incorporates blocked or undesired road information into the planning graph before route computation begins, so that the generated route already reflects the effective road availability of the active scenario. This is a meaningful contribution because it improves the consistency between the simulation environment and the navigation process. The route followed by the ego vehicle is no longer disconnected from the configured scenario conditions, but instead becomes a direct outcome of them.

A second important point is the interpretability of the developed approach. The proposed framework relies on a graph based planning representation whose adaptation can be traced directly through the constrained planning space and the resulting route. This gives the system a level of transparency that is valuable both methodologically and practically. The route changes observed in the evaluation are not the result of an opaque internal model whose reasoning remains hidden. Rather, they arise from clearly identifiable changes in the feasible road graph of the active scenario. This interpretability is particularly important in a thesis context, because it allows the planning behavior to be understood not only through numerical outputs, but also through the structural relationship between environmental constraints, graph adaptation, and resulting route geometry.

The results also show that the proposed framework achieved a useful balance between adaptability and computational simplicity. The route generation and navigation mechanism responds to changed scenario conditions. This is significant because many advanced planning approaches assume frequent online updates and correspondingly high computational demand. In contrast, the framework developed here remains comparatively lightweight by performing route adaptation before execution begins. The evaluation indicates that this design was sufficient to produce meaningful scenario dependent route changes while keeping planning times low in

both Town02 and Town01. In the context of simulation based experimentation, this represents a practical advantage, since it supports reproducibility and controlled analysis without introducing unnecessary architectural complexity.

Another discussion point concerns the meaning of "dynamic" navigation in the context of this thesis. The work does not interpret dynamic navigation as continuous global replanning during vehicle motion. Instead, the dynamic character of the framework lies in the fact that route generation changes from run to run according to the environmental conditions of the active scenario. This distinction is important because it defines the actual contribution of the work more precisely. The results show that such run-specific adaptation is already sufficient to produce meaningful and measurable differences in route behavior. In this sense, the thesis contributes to a relevant understanding of dynamic navigation in simulation.

The cross-town results further strengthen the significance of the developed framework. The fact that the same planning principle could be implemented successfully in both Town02 and Town01 suggests that the contribution is not limited to a single map-specific realization. At the same time, the numerical differences between the two towns show that the practical effect of route adaptation depends on the geometry and structure of the environment in which the mechanism is applied. This is not a weakness of the framework. On the contrary, it supports the central idea of environment aware navigation, because it shows that the system does not behave independently of its context. Instead, it responds differently when the town structure and scenario realization differ.

From a broader research perspective, the thesis can also be understood as a contribution to simulation based navigation evaluation. The developed framework does not only produce routes, it also provides a reproducible way of studying how route level planning changes when the effective road space of a scenario is modified. In this sense, the work contributes not only a navigation mechanism, but also an evaluation perspective. It highlights that route selection should itself be considered part of the scenario-dependent adaptation process, rather than treating route planning as a fixed background step that remains unaffected by the experimental conditions.

Taken together, the discussion indicates that the developed framework offers value in three main respects. First, it establishes a clear connection between scenario conditions and route generation. Second, it provides a transparent and graph based implementation that is methodologically interpretable. Third, it demonstrates that meaningful environment aware route adaptation and navigation can be achieved within a computationally manageable simulation workflow. Within the defined scope of the thesis, these points represent the main significance of the work and explain why the developed approach is a relevant contribution to the study of navigation in CARLA under structured environmental constraints.

6.2 Future Work

Although the developed navigation framework achieved its intended objective within the scope of this thesis, several directions for future work can be identified. These extensions concern both the navigation mechanism itself and the broader simulation framework in which it was evaluated. The present work focused on scenario dependent route generation before execution begins, and this already provided a useful basis for examining how environmental constraints influence navigation planning in CARLA. At the same time, the developed system also reveals a number of opportunities through which the approach could be extended further in future research.

A first natural extension would be the incorporation of continuous route adaptation during execution. In the present thesis, the route is generated before the run begins and then followed by the ego vehicle during the simulation. This design was intentionally chosen in order to study environment aware navigation and route generation in a structured and computationally manageable way. A future system could build on this foundation by adding a mechanism that monitors the route during execution and updates it whenever new relevant environmental changes occur. Such an extension would move the framework closer to a real-time adaptive navigation architecture and would allow the relation between pre-execution route adaptation and online re-planning to be studied more directly.

A second direction for future work concerns the range and complexity of the evaluated scenarios. The present thesis focused on blocked or undesired road conditions and on structured constrained scenarios in two CARLA towns. Future research could extend this by introducing a wider variety of environment changes, including more diverse traffic disturbances, dynamic lane closures, partially obstructed roads, or more complex multi-agent interaction settings. This would make it possible to assess whether the route-generation principle developed here remains effective under a broader set of navigation-relevant constraints.

A further extension would be the integration of richer traffic aware or interaction aware planning signals into the route generation process. In the current framework, route adaptation is driven primarily by structured road-space restrictions represented in the planning graph. Future work could investigate how additional information, for example, estimated congestion levels, predicted traffic-agent behavior, or dynamically changing route costs. These could be incorporated into the planning space before or during route computation. This would allow the framework to move beyond blocked-road adaptation toward a more general environment aware navigation model.

Another promising direction lies in the expansion of the evaluation setup. Although the use of Town02 and Town01 already demonstrates that the framework is not limited to a single map, future studies could extend the experiments to additional CARLA towns or even to other simulators with compatible map structures. This would allow a broader assessment of how strongly the navigation behavior depends on specific town geometry and whether the same planning principle remains

transferable across a wider set of simulation environments.

Future work could also improve the evaluation methodology itself. In the present thesis, the system was assessed primarily through route feasibility, route length, travel time, planning time, and qualitative route observation. A more advanced evaluation could include larger scale repeated runs, richer statistical analysis, and a broader set of execution related indicators. In addition, a future study could compare the developed framework more explicitly against other navigation baselines, for example nominal graph planning without constraint integration or alternative graph adaptation strategies. This would allow the practical benefits of the proposed method to be quantified more systematically.

Another relevant extension concerns the integration of the route generation framework into a more complete autonomous driving architecture. Future research could examine how the developed mechanism interacts with downstream behavior planning, motion planning, or prediction modules. Such integration would be especially valuable for studying how scenario-dependent route adaptation at the global level influences tactical and local decision-making during the drive.

Finally, future work could address the visualization and analysis pipeline in greater depth. The current system already provides split-screen route observation and CSV-based logging, which proved useful for evaluation. These capabilities could be extended further by adding automated result dashboards, richer summary metrics, or direct visual comparison between nominal and adapted routes across scenario modes. This would improve not only the analysis of route behavior, but also the reproducibility and interpretability of future simulation studies based on the developed framework.

Taken together, these future directions show that the proposed navigation framework provides a useful foundation rather than a closed final solution. The present thesis demonstrates that environment-aware route generation in CARLA can be realized in a structured and meaningful way under scenario-dependent constraints. Future work can build on this foundation by extending the method toward richer environmental adaptation, broader evaluation, and tighter integration with more advanced autonomous driving architectures.

7 Conclusion

This thesis addressed the problem of route generation under scenario-dependent environmental conditions in the CARLA simulation environment. The main objective was to develop a navigation framework in which the generated route reflects the effective road availability of a specific simulation run before execution begins. In contrast to a purely nominal route planning process, the proposed approach incorporates blocked or undesired road information into the planning graph and computes the route on this adapted representation. In this way, the work investigated a structured form of dynamic navigation at the level of run-specific route generation rather than continuous online re-planning during vehicle motion.

To achieve this objective, a graph-based navigation framework was developed and implemented in CARLA. The system combines scenario realization, map-based graph construction, environment aware constraint integration, route generation, route execution, visualization, and logging within a single simulation workflow. The framework was first realized in Town02 as the primary implementation environment and was then applied to Town01 in order to examine whether the same navigation principle remained applicable in a second town with a different road layout. This two-town realization was important because it allowed the work to move beyond a single-map demonstration while preserving the same underlying planning logic.

The evaluation results showed that the proposed navigation and route generation mechanism remained operational under nominal as well as constrained scenario conditions in both towns. In all evaluated modes, valid paths could be generated on the scenario dependent planning graph. The results also showed that stronger route restrictions led to visible and measurable changes in route behavior, including increased route length and, in several cases, increased travel time. These findings are consistent with the intended logic of the framework, since the adaptation of the planning space is expected to alter the geometry and practical behavior of the generated route when the scenario conditions become more restrictive.

A further important outcome of the thesis is that the framework remained computationally manageable despite the introduction of scenario dependent graph adaptation. The planning times remained low in both Town02 and Town01, which indicates that the developed approach achieved environment aware route generation without requiring a computationally heavy continuously re-planning architecture. From a practical perspective, this is an important result because it shows that meaningful route adaptation can be integrated into a reproducible simulation workflow while preserving transparency and implementation simplicity.

Beyond the numerical evaluation, the thesis also demonstrated that the route adaptation behavior was visible qualitatively in the simulator. Through the split-

screen visualization interface and the route overlays, it could be observed that the generated routes changed in accordance with the active scenario conditions. This qualitative evidence complements the numerical results and strengthens the interpretation that the developed framework remained environment-aware not only at the graph level, but also in the visible route behavior of the ego vehicle during execution.

Taken together, the thesis makes a focused contribution to simulation based navigation research in CARLA. It shows that scenario dependent environmental constraints can be integrated directly into route generation in a way that is methodologically clear, technically interpretable, and suitable for reproducible experimental evaluation. Its contribution is narrower and more specific: it establishes and evaluates a structured framework for environment aware route generation and navigation before execution begins. Within the defined scope of the thesis, this represents a meaningful contribution to the study of navigation under structured simulation constraints.

Bibliography

[1] Ahmed, N., Afyouni, I., Dabool, H., Al Aghbari, Z.: A systemic survey of the omniverse platform and its applications in data generation, simulation and metaverse. Frontiers in Computer Science 6, 1423129 (2024)

[2] Ajanovic, Z., Lacevic, B., Shyrokau, B., Stolz, M., Horn, M.: Search-based optimal motion planning for automated driving. in 2018 ieee. In: RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4523–4530

[3] Becker, D., Küppers, G., Eckstein, L.: Simulative performance analysis of an ad function with road network variation. arXiv preprint arXiv:2308.04446 (2023)

[4] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: Sumo–simulation of urban mobility: an overview. In: Proceedings of SIMUL 2011, the third international conference on advances in system simulation. ThinkMind (2011)

[5] Betts, J.T.: Practical methods for optimal control and estimation using nonlinear programming. SIAM (2010)

[6] Biagiola, M., Stocco, A., Riccio, V., Tonella, P.: Two is better than one: digital siblings to improve autonomous driving testing. Empirical Software Engineering 29(4), 72 (2024)

[7] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)

[8] Brunner, M., Ghignone, E., Baumann, N., Magno, M.: R-carla: High-fidelity sensor simulations with interchangeable dynamics for autonomous racing. arXiv preprint arXiv:2506.09629 (2025)

[9] Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020)

[10] CARLA: Release 0.10.0 — carla blog. `https://carla.org/2024/12/19/release-0.10.0/` (dec 2024), accessed: 2025-07-03

[11] CARLA: Advanced traffic manager — carla documentation. `https://carla.readthedocs.io/en/latest/adv_traffic_manager/` (2025), accessed: 2025-07-03

[12] CARLA: Core map — carla documentation. `https://carla.readthedocs.io/en/latest/core_map/` (2025), accessed: 2025-07-03

[13] Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: Learning affordance for direct perception in autonomous driving. In: Proceedings of the IEEE international conference on computer vision. pp. 2722–2730 (2015)

[14] Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: 2018 IEEE international conference on robotics and automation (ICRA). pp. 4693–4700. IEEE (2018)

[15] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms (3-rd edition). MIT Press and McGraw-Hill (2009)

[16] Cragun, M.: Nvidia drive sim: Omniverse early access. `https://blogs.nvidia.com/blog/nvidia-drive-sim-omniverse-early-access/` (2021), accessed: 2025-07-03

[17] Delling, D., Sanders, P., Schultes, D., Wagner, D.: Engineering route planning algorithms. In: Algorithmics of large and complex networks: design, analysis, and simulation, pp. 117–139. Springer (2009)

[18] Deo, N., Trivedi, M.M.: Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In: 2018 IEEE intelligent vehicles symposium (IV). pp. 1179–1184. IEEE (2018)

[19] Dijkstra, E.W.: A note on two problems in connexion with graphs. In: Edsger Wybe Dijkstra: his life, work, and legacy, pp. 287–290 (2022)

[20] Documentation, C.S.: Driving benchmark performance metrics. `https://carla.readthedocs.io/en/0.8.4/benchmark_metrics/` (2024), accessed: 2026-03-16

[21] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning. pp. 1–16. PMLR (2017)

[22] e.V., A.: Asam openscenario: User guide. `https://www.asam.net/fileadmin/Standards/OpenSCENARIO/ASAM_OpenSCENARIO_BS-1-2_User-Guide_V1-2-0.html` (May 2022), accessed: 2026-03-17

[23] Falcone, P., Borrelli, F., Asgari, J., Tseng, H.E., Hrovat, D.: Predictive active steering control for autonomous vehicle systems. IEEE Transactions on control systems technology 15(3), 566–580 (2007)

[24] Fremont, D.J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: a language for scenario specification and scene generation. In: Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation. pp. 63–78 (2019)

[25] Fremont, D.J., Kim, E., Pant, Y.V., Seshia, S.A., Acharya, A., Bruso, X., Wells, P., Lemke, S., Lu, Q., Mehta, S.: Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). pp. 1–8. IEEE (2020)

[26] Gambi, A., Maul, P., Mueller, M., Stamatogiannakis, L., Fischer, T., Panichella, S.: Soft-body simulation and procedural generation for the development and testing of cyber-physical systems. Technical report, BeamNG GmbH (2022), published in the context of ICSE-SEIP 2023 Companion Volume

[27] Gangel, K., Hamar, Z., Háry, A., Horváth, Á., Jandó, G., Könyves, B., Panker, D., Pintér, K., Pataki, M., Szalai, M., et al.: Modelling the zalazone proving ground: a benchmark of state-of-the-art automotive simulators prescan, ipg carmaker, and vtd vires. Acta Technica Jaurinensis 14(4), 488–507 (2021)

[28] Güllü, A., Shah, F.A., Pfahl, D.: An lstm-based test selection method for self-driving cars. arXiv preprint arXiv:2501.03881 (2025)

[29] Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)

[30] Hexagon: Virtual test drive. `https://hexagon.com/products/virtual-test-drive` (2025), accessed: 2025-07-03

[31] Janai, J., Güney, F., Behl, A., Geiger, A.: Computer vision for autonomous vehicles: Problems, datasets and state of the art. Foundations and Trends in Computer Graphics and Vision 12(1-3), 1–308 (2020)

[32] Kalra, N., Paddock, S.M.: Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Transportation research part A: policy and practice 94, 182–193 (2016)

[33] Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The international journal of robotics research 30(7), 846–894 (2011)

[34] Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE transactions on Robotics and Automation 12(4), 566–580 (2002)

[35] Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.M., Lam, V.D., Bewley, A., Shah, A.: Learning to drive in a day. In: 2019 international conference on robotics and automation (ICRA). pp. 8248–8254. IEEE (2019)

[36] Kesting, A., Treiber, M.: Traffic flow dynamics: data, models and simulation. No. Book, Whole)(Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[37] Kong, J., Pfeiffer, M., Schildbach, G., Borrelli, F.: Kinematic and dynamic vehicle models for autonomous driving control design. In: 2015 IEEE intelligent vehicles symposium (IV). pp. 1094–1099. IEEE (2015)

[38] Konthala, S., Becker, D., Eckstein, L.: Open-source tool based framework for automated performance evaluation of an ad function. In: 2023 International Conference on Electrical, Computer and Energy Technologies (ICECET). pp. 1–7. IEEE (2023)

[39] Koren, M., Nassar, A., Kochenderfer, M.J.: Finding failures in high-fidelity simulation using adaptive stress testing and the backward algorithm. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 5944–5949. IEEE (2021)

[40] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent development and applications of sumo-simulation of urban mobility. International journal on advances in systems and measurements 5(3&4) (2012)

[41] Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., How, J.P.: Real-time motion planning with applications to autonomous urban driving. IEEE Transactions on control systems technology 17(5), 1105–1118 (2009)

[42] LaValle, S.M.: Planning algorithms. Cambridge university press (2006)

[43] LaValle, S.M., Kuffner Jr, J.J.: Randomized kinodynamic planning. The international journal of robotics research 20(5), 378–400 (2001)

[44] Leaderboard, C.: Evaluation criteria for the leaderboard 2.0. `https://leaderboard.carla.org` (2020), accessed: 2026-03-16

[45] Li, Y., Yuan, W., Zhang, S., Yan, W., Shen, Q., Wang, C., Yang, M.: Choose your simulator wisely: A review on open-source simulators for autonomous driving. IEEE Transactions on Intelligent Vehicles (2024)

[46] Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., Wießner, E.: Microscopic traffic simulation using sumo. In: 2018 21st international conference on intelligent transportation systems (ITSC). pp. 2575–2582. Ieee (2018)

[47] Ma, C., Yang, J., Xue, J., Liu, Y., Ma, L.: State-statistical model based trajectory-band planning in urban environment. In: 2015 IEEE Intelligent Vehicles Symposium (IV). pp. 400–405. IEEE (2015)

[48] Maierhofer, S., Althoff, M.: Commonroad-carla interface: Bridging the gap between motion planning and 3d simulation. In: 2024 IEEE Intelligent Vehicles Symposium (IV). pp. 2702–2708. IEEE (2024)

[49] Martelli, A.: On the complexity of admissible search algorithms. Artificial Intelligence 8(1), 1–13 (1977)

[50] Micheli, F., Bersani, M., Arrigoni, S., Braghin, F., Cheli, F.: Nmpc trajectory planner for urban autonomous driving. Vehicle system dynamics 61(5), 1387–1409 (2023)

[51] Nguyen, M.T., Pitz, J., Krantz, W., Neubeck, J., Wiedemann, J.: Subjective perception and evaluation of driving dynamics in the virtual test drive. SAE International Journal of Vehicle Dynamics, Stability, and NVH 1(2017-01-1564), 247–252 (2017)

[52] OSSDC: Running svl simulator — svl documentation. `https://race.ossdc.org/docs/running-simulations/running-simulator/` (2025), accessed: 2025-07-03

[53] Paden, B., Čáp, M., Yong, S.Z., Yershov, D., Frazzoli, E.: A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles 1(1), 33–55 (2016)

[54] Pivtoraiko, M.N.: Differentially constrained motion planning with state lattice motion primitives (2012)

[55] Ratliff, N., Zucker, M., Bagnell, J.A., Srinivasa, S.: Chomp: Gradient optimization techniques for efficient motion planning. In: 2009 IEEE international conference on robotics and automation. pp. 489–494. IEEE (2009)

[56] Rong, G., Shin, B.H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., et al.: Lgsvl simulator: A high fidelity simulator for autonomous driving. In: 2020 IEEE 23rd International conference on intelligent transportation systems (ITSC). pp. 1–6. IEEE (2020)

[57] Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 627–635. JMLR Workshop and Conference Proceedings (2011)

[58] Sadigh, D., Sastry, S., Seshia, S.A., Dragan, A.D.: Planning for autonomous cars that leverage effects on human actions. In: Robotics: Science and systems. vol. 2, pp. 1–9. Ann Arbor, MI, USA (2016)

[59] Schmidt, J., Khan, Q., Cremers, D.: Lidar view synthesis for robust vehicle navigation without expert labels. In: 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). pp. 2835–2842. IEEE (2023)

[60] Schulman, J., Ho, J., Lee, A.X., Awwal, I., Bradlow, H., Abbeel, P.: Finding locally optimal, collision-free trajectories with sequential convex optimization. In: Robotics: science and systems. vol. 9, pp. 1–10. Berlin, Germany (2013)

[61] Schwarting, W., Alonso-Mora, J., Rus, D.: Planning and decision-making for autonomous vehicles. Annual Review of Control, Robotics, and Autonomous Systems 1(1), 187–210 (2018)

[62] Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and Service Robotics: Results of the 11th International Conference. pp. 621–635. Springer (2018)

[63] Shimanuki, G.K.G., Nascimento, A.M., Vismari, L.F., Junior, J.B.C., Junior, J.R.d.A., Cugnasca, P.S.: Navigating the edge with the state-of-the-art insights into corner case identification and generation for enhanced autonomous vehicle safety. arXiv preprint arXiv:2503.00077 (2025)

[64] Simulator, C.: Carla simulator github repository. `https://github.com/carla-simulator/carla` (2026), accessed: 2026-03-16

[65] Song, Z., He, Z., Li, X., Ma, Q., Ming, R., Mao, Z., Pei, H., Peng, L., Hu, J., Yao, D., et al.: Synthetic datasets for autonomous driving: A survey. IEEE Transactions on Intelligent Vehicles 9(1), 1847–1864 (2023)

[66] SUMO: Interfacing traci from python — sumo documentation. `https://sumo.dlr.de/docs/TraCI/Interfacing_TraCI_from_Python.html` (feb 2025), accessed: 2025-07-03

[67] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2446–2454 (2020)

[68] Sutton, R.S., Barto, A.G., et al.: Reinforcement learning: An introduction, vol. 1. MIT press Cambridge (1998)

[69] Talebpour, A., Mahmassani, H.S.: Influence of connected and autonomous vehicles on traffic flow stability and throughput. Transportation research part C: emerging technologies 71, 143–163 (2016)

[70] Thalapanane, S., Kumar, S.S.S., Peethambari, G.N.A.D., SriHari, S., Zheng, L., Poveda, J., Lin, M.C.: Traverse: Traffic-responsive autonomous vehicle experience & rare-event simulation for enhanced safety. In: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 5513–5520. IEEE (2024)

[71] V2, C.A.D.L.: Evaluation criteria for the leaderboard 2.0. `https://leaderboard.carla.org/evaluation_v2_0/` (2025), accessed: 2026-03-16

[72] Wang, J., Wang, Y., Zhang, D., Yang, Y., Xiong, R.: Learning hierarchical behavior and motion planning for autonomous driving. In: 2020 IEEE/RSJ

International Conference on Intelligent Robots and Systems (IROS). pp. 2235–2242. IEEE (2020)

[73] Wang, T., Qu, D., Song, H., Dai, S.: A hierarchical framework of decision making and trajectory tracking control for autonomous vehicles. Sustainability 15(8), 6375 (2023)

[74] Wei, J., Snider, J.M., Gu, T., Dolan, J.M., Litkouhi, B.: A behavioral planning framework for autonomous driving. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings. pp. 458–464. IEEE (2014)

[75] Werling, M., Ziegler, J., Kammel, S., Thrun, S.: Optimal trajectory generation for dynamic street scenarios in a frenet frame. In: 2010 IEEE international conference on robotics and automation. pp. 987–993. IEEE (2010)

[76] Wikipedia, W.: A* search algorithm. `https://en.wikipedia.org/wiki/A%2A_search_algorithm` (jun 2025), accessed: 2025-07-28

[77] Wikipedia, W.: Dijkstra's algorithm. `https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm` (jul 2025), accessed: 2025-07-28

[78] Xia, T., Chen, H.: A survey of autonomous vehicle behaviors: Trajectory planning algorithms, sensed collision risks, and user expectations. Sensors 24(15), 4808 (2024)

[79] Yang, K., Guo, Z., Lin, G., Dong, H., Zuo, D., Peng, J., Huang, Z., Xu, Z., Li, F., Bai, Z., et al.: Natural-language-driven simulation benchmark and copilot for efficient production of object interactions in virtual road scenes. arXiv preprint arXiv:2312.04008 (2023)

[80] Ye, C., Xiong, G., Shang, J., Dai, X., Gong, X., Lv, Y.: Sumo-mcp: Leveraging the model context protocol for autonomous traffic simulation and optimization. arXiv preprint arXiv:2506.03548 (2025)

[81] Zhao, D.: Accelerated Evaluation of Automated Vehicles. Ph.D. thesis (2016)



This report - except logo Chemnitz University of Technology - is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this report are included in the report's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the report's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.

Chemnitzer Informatik-Berichte

In der Reihe der Chemnitzer Informatik-Berichte sind folgende Berichte erschienen:

CSR-25-01 Md. Ali Awlad, Hasan Saadi Jaber Aljzaere, Wolfram Hardt, AUTOSAR Software Component for Atomic Straight Driving Patterns, März 2025, Chemnitz

CSR-25-02 Billava Vasantha Monisha, Hasan Saadi Jaber Aljzaere, Wolfram Hardt, Automotive Software Component for QT Based Car Status Visualization, März 2025, Chemnitz

CSR-25-03 Zahra Khadivi, Batbayar Battseren, Wolfram Hardt, Acoustic-Based MAV Propeller Inspection, Mai 2025, Chemnitz

CSR-25-04 Tripti Kumari Shukla, Ummay Ubaida Shegupta, Wolfram Hardt, Time Management Tool Development to Support Self-regulated Learning, August 2025, Chemnitz

CSR-25-05 Ambu Babu, Ummay Ubaida Shegupta, Wolfram Hardt, Development of a Retrieval Model based Backend of a Tutoring Agent, August 2025, Chemnitz

CSR-25-06 Shahid Ismail, Ummay Ubaida Shegupta, Wolfram Hardt, Development of a Generative Model based Backend of Tutoring Agent, August 2025, Chemnitz

CSR-25-07 Chaitanya Sravanthi Akula, Ummay Ubaida Shegupta, Wolfram Hardt, Integration of Learning Analytics into the ARC-Tutoring Workbench, August 2025, Chemnitz

CSR-25-08 Jörn Roth, Reda Harradi, Wolfram Hardt, Implementation of a Path Planning Algorithm for UAV Navigation, Dezember 2025, Chemnitz

CSR-25-09 Alhassan Khalil, Reda Harradi, Stephan Rupf, Wolfram Hardt, Development of an Automation Framework for 1D Measurement, Dezember 2025, Chemnitz

CSR-26-01 Vismay Gunda, Shadi Saleh, Wolfram Hardt, Cloud-Based AI Solutions for Ensuring Data Quality in Predictive Models, Februar 2026, Chemnitz

CSR-26-02 Sami Mansoor Alavi, Shadi Saleh, Wolfram Hardt, Continuous Integration for Cloud-Based Swarm Farming Applications, Februar 2026, Chemnitz

Chemnitzer Informatik-Berichte

CSR-26-03 Sarah Onyinyechi Obasi, Shadi Saleh, Wolfram Hardt, Cloud-Based AI for Data Completeness Analysis and Improvement in Predictive Modeling, März 2026, Chemnitz

CSR-26-04 Atul Chandra Nath, Reda Harradi, Wolfram Hardt, GPS-based UAV Precision Landing, April 2026, Chemnitz

CSR-26-05 Josey Mol Sibi, Batbayar Battseren, Wolfram Hardt, Real Time Multi Stream Video Transmission in Autonomous UAV, Mai 2026, Chemnitz

CSR-26-06 Silpa Geetha Harshakumar, Batbayar Battseren, Wolfram Hardt, Simultaneous Video Streaming and Recording with Geotagging in Autonomous UAV, Mai 2026, Chemnitz

CSR-26-07 Monica Chambial, Batbayar Battseren, Wolfram Hardt, Implementation and Enhancement of a Mission Scheduling Component for Autonomous UAV, Mai 2026, Chemnitz

CSR-26-08 Ganapathi Teja Chirravuri, Batbayar Battseren, Wolfram Hardt, Dynamic Model Switching for Real-Time UAV Inspection, Oktober 2026, Chemnitz

CSR-26-09 Hariharan Chandrasekaran, Batbayar Battseren, Wolfram Hardt, LLM-Powered Novel Scenario Extraction and Reasoning in Autonomous Driving, Oktober 2026, Chemnitz

CSR-26-10 Anh Tu Nguyen, Julkar Nine, Wolfram Hardt, Integration of Deep learning Algorithms in a Cloud Server-Based Carla Simulation Environment, November 2026, Chemnitz

CSR-26-11 S M Tanvir Abid, Julkar Nine, Wolfram Hardt, Dynamic Navigation Function in CARLA Simulation Environment, November 2026, Chemnitz

Chemnitzer Informatik-Berichte

ISSN 0947-5125

Herausgeber: Fakultät für Informatik, TU Chemnitz
Straße der Nationen 62, D-09111 Chemnitz