RoadGPT: AI-Based procedural generation of virtual roads

Bachelor Thesis

Submitted to
IMC University of Applied Sciences Krems

Bachelor Programme Informatics

by

Benedikt Steininger

for the award of academic degree
Bachelor of Science in Engineering (BSc)

under the supervision of
Alessio Gambi, PhD

Submitted on 15.01.2024

DECLARATION OF HONOUR

I declare on my word of honour that I have written this Bachelor Thesis on my own and that I have not used any sources or resources other than stated and that I have marked those passages and/or ideas that were either verbally or textually extracted from sources. This also applies to drawings, sketches, graphic representations as well as to sources from the internet. The Bachelor Thesis has not been submitted in this or similar form for assessment at any other domestic or foreign post-secondary educational institution and has not been published elsewhere. The present Bachelor Thesis complies with the version submitted electronically.

Benedikt Steininger
15.01.2024

ABSTRACT

The emergence of self-driving vehicles marks a significant change in the automotive industry, with companies like Uber and Waymo leading the way in this new technology. To ensure road safety, it is essential to conduct rigorous testing of these vehicles, especially as they are introduced into live traffic. Field testing can be costly and risky, so most of the testing is done in simulations.

This study aimed to simplify the generation of virtual roads, which is typically reliant on domain models and heuristics. It utilized generative AI to transform textual road descriptions into detailed machine-readable formats. These outputs were then used to create roads automatically within a simulator.

The findings indicated that generative AI holds promise for this application. However, since the model used in this study was not specifically trained for this task, it encountered challenges in certain areas. These issues might be resolved by fine-tuning the current model or developing a new one.

Keywords: Prompt Engineering, Generative AI, Autonomous Vehicles

Table of Contents

List of Tables

List of Figures

Chapter 1
INTRODUCTION

This section will discuss the research questions and the research approach of the thesis, as well as the motivation behind it.

1.1 Motivation

Artificial intelligence (AI) has entered various industries during the past few years, including the automotive sector. From advanced driver assistance systems to Tesla's autopilot, AI is used to improve traffic safety. Testing autonomous vehicles in actual traffic situations would be risky, defeating the goal of assuring safety.

As a result, simulations are used to test autonomous vehicles. These simulations are created by generating roads and environments to analyse the cars' behavior. Most existing approaches for road generation rely on domain models and metaheuristics. They are limited to generating 2D roads with fixed layouts. These problems could be solved using generative AI.

Generative AI is a branch of Artificial Intelligence that focuses on creating new content. With the use of large language models like ChatGPT [15], there is potential to generate virtual roads for testing by providing verbal descriptions such as "mountain road" or "highway". The output of ChatGPT can then be automatically translated and passed into driving simulators.

1.2 Research Questions

The primary objective of this thesis is to answer the following research question.

Research Question 1 (RQ1): Can generative AI be used to automatically create three-dimensional roads for testing autonomous vehicles?

By harnessing the capabilities of generative AI, the tool streamlines the process of generating roads for self-driving vehicles, eliminating the necessity for complex algorithmic coding.

1.3 Research Method

The research approach consists of four main phases: a review of the state-of-the-art project, implementation of the project, and data gathering and analysis of the data. The findings of the state-of-the-art will serve as a basis for the implementation of the project.

The implementation can be divided into four parts that build on each other. The first part consists of prompt engineering (see section 2.1.3) and receiving the output in a format that can be used in the next part. In the second part, a Python script will translate the output into road nodes and interpolate the road. The third part consists of validating the correctness of the road (steepness, sharpness of turns, etc.). The fourth part is building the road in the simulator. The roads will then be evaluated in the context of a user study.

The quantitative data gathered through the validation script and the user study will then be used for a statistical analysis of the results.

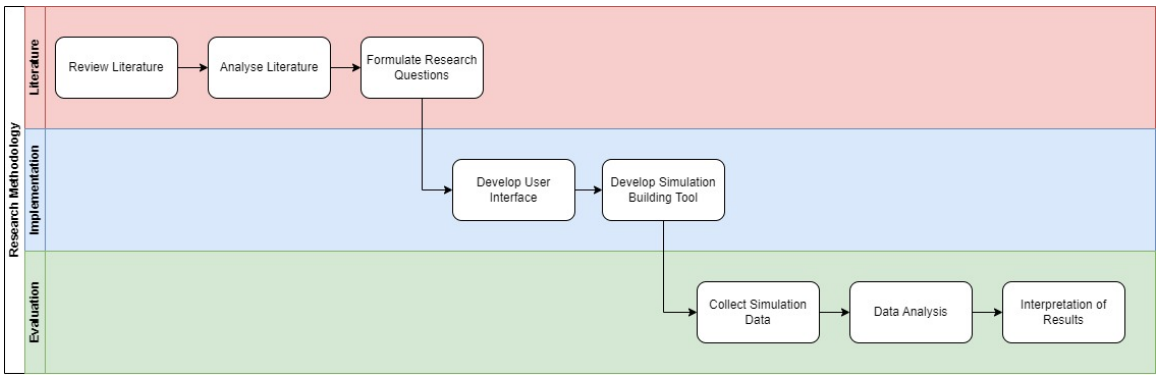

Figure 1.1: Research Methodology

1.4 Structure

The bachelor thesis is structured as follows:

- Chapter 1: Introduction

  The opening chapter of this bachelor thesis holds significant importance in setting the stage for the research. Within this section, I provide a thorough outline of the motivation driving the study, articulate the fundamental research questions, and present a concise overview of the chosen research methodology. By elaborating on the motivation, the significance and relevance of the study in a broader context is established.

The formulation of research questions provides a clear roadmap, guiding readers through the central inquiries shaping the investigation. Furthermore, a comprehensive discussion of the selected research methodology offers insights into the tools and approaches used to address the research questions. This chapter serves as an essential guide, laying the foundation for subsequent chapters and contributing to the overall coherence and clarity of the thesis.

- Chapter 2: Background

  This chapter provides a thorough background that explains the domains and technologies related to the topic. Its primary objective is to supply readers with an overview and the essential vocabulary necessary for understanding the following chapters. By exploring relevant domains, this section seeks to place the research within a broader context, providing the foundation of the study. Additionally, an examination of related technologies is conducted to acquaint the reader with the tools and methodologies essential to this research.

- Chapter 3: Related Work

  The "Related Work" chapter forms a critical component of this bachelor thesis, offering a comprehensive analysis of existing literature and research relevant to the subject matter. In this section, a thorough review of the current state of the art within the field is conducted, examining scholarly contributions, methodologies, and findings that are directly connected to the research objectives. By critically evaluating the existing body of work, gaps, trends, and debates in the literature can be identified. This structured examination of the related work is crucial for establishing the academic foundation of the thesis and demonstrating the unique contributions of the research.

- Chapter 4: Methodology

  This chapter provides a comprehensive overview of the methodology used in this thesis. It offers detailed insights into the systematic approach taken to address the research questions. The chapter covers the research design, data collection methods, and analytical techniques employed, aiming to bring clarity to the procedural aspects of the study. By outlining the methodology, this section not only serves as a guide for potential replication but also establishes the credibility and robustness of the research, laying a solid foundation for the subsequent analysis and interpretation of findings.

- Chapter 5: Implementation

  Chapter 5 focuses on the practical implementation of the project, explaining how the code works and which technologies were used. It provides a detailed explanation of

the project's technical aspects, including the software tools and programming language utilized. This chapter aims to offer readers a clear understanding of how the project was implemented.

- Chapter 6: Evaluation

  The "Evaluation" chapter serves as a comprehensive exploration of the methods used for assessing the tool's effectiveness. It extensively discusses the procedures used for data collection, analysis, and interpretation. Additionally, it provides a detailed exploration of the user study used to evaluate the tool, covering its setup and execution. Through a thorough examination of these elements, the chapter aims to offer a comprehensive understanding of the evaluation process and provide insights into the tool's performance.

- Chapter 7: Summary

  This chapter presents the main findings of the study. The insights obtained from the results are presented by systematically analyzing and interpreting the collected data. The findings are discussed in detail, highlighting their relevance to the research questions and objectives.

Chapter 2
BACKGROUND

In this section, the background of the most important technologies used will be presented, setting up grounds for discussion.

2.1 Artificial Intelligence

IBM defines Artificial Intelligence as the use of technology to enable computers and machines to replicate human intelligence and problem-solving abilities.

Artificial intelligence (AI), whether used independently or combined with other technologies like sensors and robotics, accomplishes tasks that are typically performed by humans. Examples include digital assistants, GPS navigation, autonomous vehicles, and sophisticated AI tools like OpenAI's ChatGPT, which are widely used in today's society.

In the field of computer science, AI includes areas like machine learning and deep learning, which aim to mimic human thinking through algorithmic systems. These algorithms can "learn" from the data they receive, getting better at making decisions over time through continuous improvement processes.

Artificial intelligence can be categorized into two groups: Weak AI and Strong AI.

Weak AI, also called narrow AI or artificial narrow intelligence (ANI), is designed and trained for specific tasks, driving much of today's AI applications like Siri, Alexa, IBM Watson, and self-driving cars. On the other hand, strong AI consists of artificial general intelligence (AGI) and artificial super intelligence (ASI). AGI aims to match human intelligence, possessing self-awareness and problem-solving abilities, while ASI would surpass human intelligence. Although strong AI remains theoretical, ongoing research explores its potential [10].

2.1.1 Natural Language Processing

Natural Language Processing (NLP) refers to a set of computational techniques designed for automatically analyzing and representing human languages. It involves tasks such as accessing and acquiring lexical, semantic, and episodic characteristics of language, creating and propagating dynamic bindings, manipulating recursive structures, coordinating processing modules, identifying language constructs, and representing abstract concepts. The ultimate goal of NLP is to advance from basic language processing to achieving natural language understanding, which involves grasping context and meaning beyond mere syntactic structures. Presently, NLP heavily relies on syntactic representations, which are constrained by their inability to incorporate background knowledge similar to human comprehension. The focus of NLP centers on replicating human cognitive processes and leveraging semantic features that are implicit in the text. These computational models support both theoretical exploration and practical implementations, enabling efficient communication between humans and machines [4].

2.1.2 Transformers

Transformer architectures leverage a self-attention mechanism to understand relationships within sequences, allowing them to capture long-range dependencies more effectively compared to recurrent networks. Unlike traditional models that rely on recursion, Transformers process entire sequences at once, improving scalability and efficiency. They are built solely on attention mechanisms, with a unique implementation known as multi-head attention, optimized for parallelization. Transformers excel in handling complex models and large datasets, offering advantages over alternatives like hard attention, which relies on stochastic sampling. Due to their minimal prior knowledge assumption, Transformers are typically pre-trained on large-scale unlabeled datasets using pretext tasks, enabling them to encode rich and generalizable representations. These representations are then fine-tuned on supervised tasks to achieve optimal performance [11].

2.1.3 Generative and Conversational AI

Generative AI is a type of Artificial Intelligence that uses deep learning models to create human-like content in response to complex prompts, such as languages, instructions, or questions. Examples of this technology include OpenAI's ChatGPT [15] and DALL-E [16]. These AI models generate responses that go beyond their explicit programming, distinguishing them from other AI models. Conversational AI is an Artificial Intelligence subset that focuses on human-like interactions, often relying on predefined responses. However, some Conversational AI models can also generate content. Models like ChatGPT use both Generative and Conversational AI to enhance their capabilities, allowing them to better simulate human conversation and generate more dynamic responses. [12]

ChatGPT

ChatGPT, an AI-driven service accessible online, is designed for tasks such as text organization, summarization, and composition. GPT, short for "Generative Pre-trained Transformer", describes the manner in which ChatGPT processes requests and generates responses. It is built to understand and respond to user queries and commands by analyzing extensive text data to grasp word relationships. By predicting the most probable next word, similar to search engine auto-complete features, ChatGPT refines its responses iteratively. During training, exemplified by completing sentences such as "instead of turning left, she turned ...", ChatGPT evolves from random word suggestions to more accurate predictions. Because of the wide variety of possible answers, the model's numerical settings, called "weights" or "parameters," slowly change as it learns, capturing what it learns without saving or copying the original sentences [17].

In response to natural language inputs, also called "prompts", ChatGPT provides text output. One example of such a prompt would be: "Explain to me in one paragraph what ChatGPT is." ChatGPT provided the following answer (Fig 2.1).

ChatGPT is an advanced language model developed by OpenAI that uses deep learning techniques to generate human-like text responses. Trained on a vast amount of internet text, it has the ability to understand and generate coherent and contextually relevant responses to user queries, making it a powerful tool for conversational interactions, information retrieval, and providing assistance across a wide range of topics. [18]

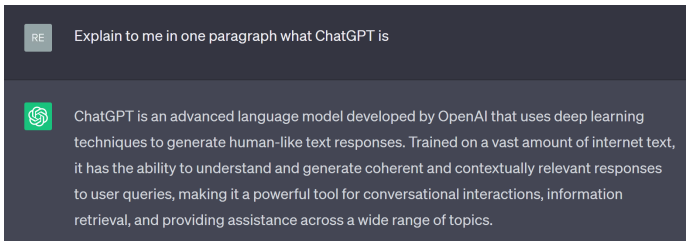

Figure 2.1: ChatGPT - prompt and answer

Prompt Engineering

Prompt engineering is the process of using prompts to program Large Language. Models (LLMs), like ChatGPT. An LLM is given a set of instructions called a prompt to help it tailor its output and user interactions. Engineering prompts to enforce rules, automate procedures, and guarantee particular output qualities and output quantities, which is known as prompt engineering. It is an ever-more-important skill set required to communicate with LLMs and can be used to address a variety of issues, such as automating software development tasks when speaking with LLMs. [31]

Microsoft outlined a set of best practices in their guide to prompt engineering.

- **Be specific**: Provide clear and precise instructions, minimizing room for interpretation and narrowing the operational scope.
- **Be descriptive**: Use descriptive language and analogies to enhance understanding.
- **Double Down**: Reinforce key points by repeating instructions, both before and after presenting primary content, employing cues for better comprehension.
- **Order Matters**: Consider the order of information presentation, as it can influence the model's output. Be mindful of whether instructions precede or follow content, and recognize the impact of recency bias in the order of few-shot examples.
- **Give the model an "out"**: Offer the model an alternative option or an "out" in case it struggles with the assigned task. For instance, suggest a default response like "not found" to help the model avoid generating inaccurate information.[13]

2.2 Autonomous Vehicles

There are six levels of driving automation, ranging from no driving automation (Level 0) to full driving automation (Level 5). At Level 0, vehicles are entirely manually controlled, with no automation in the dynamic driving task.

Level 1 involves a single automation feature, like cruise control or steering assistance, while the driver retains control over other driving tasks. At Level 2, the car takes control over steering, acceleration, and braking. However, it still requires supervision from a human driver, who must take control when necessary.

Level 3 marks conditional driving automation, in which vehicles have environmental detection capabilities and can make informed decisions. However, human supervision and intervention are still necessary.

Level 4 introduces high driving automation, allowing vehicles to intervene in the event of system failures. While these cars can operate in self-driving mode, their use is limited to specific areas.

Level 5 represents full driving automation, where vehicles no longer require human attention. These fully autonomous cars have the ability to operate without restrictions, mimicking the capabilities of experienced human drivers.

Autonomous vehicles are vehicles that operate on Level 5 of the driving automation levels, which means they do not require human interaction[28][23][30].

2.3 Computer Simulation

The Cambridge Dictionary defines a simulation as *a model of a real activity, created for training purposes or to solve a problem* [21].

A computer simulation is a program that imitates the operations of a real-world system or process over time. This program, referred to as a computer simulation model, uses algorithms to calculate the system's states over a specified time period, effectively creating a numerical representation of the system's evolution. The simulation generates data that can be visualized, often in a manner resembling scientific instrument readings. Simulations are typically used when a model's mathematical equations are complex and cannot be solved analytically, necessitating step-by-step computational methods. It's worth noting that the term "computer simulation", in the narrowest sense, refers to a specific implementation of an algorithm on a particular digital computer using a specific language and compiler. Variations in these elements can result in different outcomes from the simulation [32].

2.3.1 Beamng.tech

BeamNG.tech is an adapted version of the simulation game BeamNG.drive, created for academic and industrial purposes. While sharing many similarities, BeamNG.tech extends its functionality to support driver training and advanced driver-assistance systems (ADAS). It utilizes a soft-body physics engine for vehicle simulation [1].

Soft-body physics, as implemented in BeamNG.drive, is a fundamental aspect of the game's physics model. Unlike traditional "Rigid Body" physics simulation found in most games, BeamNG.drive utilizes a "Soft Body" physics simulation, which means that objects such as cars are deformable, allowing for more realistic and dynamic interactions [3].

Additionally, BeamNG.tech incorporates tools such as BeamNGpy, a Python interface that automates testing and data gathering. Although its primary focus is on simulating ground-based vehicles, BeamNG.tech also offers simulations for air and maritime scenarios. Its research-oriented features include a sensor suite equipped with cameras, LIDAR, ultrasonic, electric, and IMU sensors tailored specifically for autonomous driving applications. BeamNGpy streamlines testing scenarios for learning systems and ensures the validation of autonomous driving software. While BeamNG.tech is not open-source, researchers can apply for a free license on their website [1].

2.4 Procedural content generation

In the realm of video games, procedural content generation (PCG) is the practice of using algorithms instead of manual design to generate elements like levels, characters, or environments. PCG has become increasingly significant in game development and technical game research because of its ability to efficiently create diverse and dynamic content. This approach involves the methodical application of algorithms and rules to produce content, providing developers with an automated and adaptable method to create a wide range of unique gaming experiences [8][27].

Procedural generation in the context of virtual roads involves generating roads by seamlessly connecting individual road segments. This approach ensures a continuous and gapless network, where each segment's front line serves as the back line for the subsequent one, resulting in a smooth transition. The procedural generation process also ensures that roads with impossible configurations are not created in order to maintain validity [7][8].

2.5 Testing of autonomous vehicles

Testing is essential to make sure autonomous vehicles are safe, dependable, and compliant with the law. It helps to reduce risks and improve algorithms for making decisions. Such testing increases public confidence in the technology by demonstrating safe and efficient functionality in a variety of driving scenarios.

2.5.1 Field Testing

Autonomous vehicles can be tested in real-world scenarios, like Google driverless cars, which are tested mainly in real traffic, according to Huang et al. [9]. Other real-world testing approaches include testing facilities. This form of testing is particularly beneficial because it mirrors the environment where the autonomous vehicle will eventually operate, providing a realistic assessment of its performance.

While tests in real-world settings are always the final step in the validation chain, according to Huang et al., self-driving cars would have to drive hundreds of millions of kilometers without accidents to prove their safety statistically [9]. Since that is not feasible because of time and cost, simulation-based testing offers an alternative [25].

2.5.2 Simulation-based testing

The simulation-based testing of autonomous cars is carried out in a virtual setting, where the vehicles are evaluated against a variety of simulated scenarios that resemble real-world events. These scenarios can be repeated and altered as many times as necessary, incorporating different weather conditions, pedestrian movements, and traffic conditions. This allows the testing of uncommon or risky events without any real risk. Simulation testing is very valuable because it enables thorough testing in less time and at a lower cost than real-world testing. However, there are limitations since it depends on how accurately the simulations capture the complexity and unpredictability of real-world driving [7][9][20].

2.6 Context in Domain

In the practical implementation, the prompts provided to ChatGPT will consist of high-level descriptions of roads. ChatGPT will generate more detailed descriptions of those roads as output. Subsequently, a Python program will translate these detailed road descriptions into nodes containing x, y, and z-coordinates. These translated road nodes will then be passed into the simulator using the BeamNGpy library. Within the simulator, a scenario will be constructed based on the road nodes received from the Python code, allowing for the visualization and simulation of the generated roads. By combining ChatGPT's language generation capabilities, Python programming, and the simulation environment provided by BeamNG.tech, this system enables the generation, translation, and simulation of roads based on textual descriptions.

Chapter 3
Related Work

In this chapter, related works will be discussed.

3.1 Generation of Virtual Roads

Gambi et al. point out that the conventional methods of testing autonomous vehicles in actual traffic situations pose significant risks, are expensive, and have resulted in fatal incidents. In response to these challenges, the authors devised a solution known as AsFault. This automated tool tests software for self-driving cars by creating virtual road networks and simulating driving scenarios. It utilizes a combination of a genetic algorithm and procedural content generation to create road networks that expose potential safety issues. The primary objective is identifying bugs and problematic environmental conditions that can cause self-driving cars to fail. AsFault focuses explicitly on testing the software's lane-keeping capability. By evolving the road networks based on a fitness function that considers the distance between the car and the center of the lane, AsFault generates test cases where the car deviates from its intended path. The tool can be installed as a Python package and relies on the BeamNG.research driving simulator. Test cases generated by AsFault are saved in JSON format and can be re-executed. It also provides visualizations of the results, including the number of instances where the car goes out of bounds and the coverage of road segments achieved by the generated test suite [7].

In their research, Tang et al. showed that High-definition (HD) maps, which are critical for evaluating ADSs in simulated environments, come with certain limitations. They include features such as road networks, traffic lights, and signs, which essentially influence the range of testing scenarios, directly affecting the tests' effectiveness. However, several issues arise with these HD maps. Firstly, commercial HD maps often have a significant quantity of repeated elements, such as junctions of similar shapes, leading to a lack of

diversity in the testing scenarios. Secondly, the HD maps included with open-source simulators are usually relatively small, possessing a restricted number of roads and junctions, further limiting the diversity of testing scenarios. Thirdly, the unique junction structures found in different cities or regions make it time-consuming to test ADSs thoroughly across various HD maps from different locations. Lastly, the process of manually creating diverse HD maps using specific tools is quite demanding and labor-intensive, especially considering the myriad of scenario requirements [29].

To address these issues, Tang et al. introduced an innovative solution called Feat2Map. This is an automated framework for generating HD maps based on essential features, tailored for simulation testing of autonomous driving systems. It operates by taking existing maps, extracting crucial features, and then utilizing combinatorial sampling to generate grid-layout HD maps. Feat2Map is also designed to accept manual feature configurations, providing a degree of customization flexibility. The goal of this framework is to eradicate duplicated intersections, while also ensuring a maximum diversity of intersection structures and potential scenarios. The end result is a concise HD map that maintains a comparable level of scenario diversity [29].

The Search-Based and Fuzz Testing (SBFT) workshop, previously focused on Search-Based Software Testing (SBST), aims to bring together researchers and practitioners from SBST, Fuzzing, and Software Engineering to discuss and advance the automation of software testing. The workshop aims to explore the application of search and fuzzing techniques in testing, as well as their integration with other software engineering areas. The workshop includes a research track, keynotes, testing tool competitions, and a panel discussion to facilitate knowledge sharing and foster new advancements in SBFT research [24].

One of the competitions inside the SBFT workshop is the CPS (Cyber-Physical Systems) Testing Tool Competition, which aims to encourage researchers to investigate the problem of testing safety-critical CPSs (such as self-driving cars) and provide a shared framework for benchmarking test generators. The competition involves generating the highest number of diverse failure-inducing inputs, i.e., valid virtual roads that cause the autonomous vehicle to drive out of the lane. The infrastructure detects a failure each time the ego-car (partially) drives outside the lane, and the competitors are ranked based on various aspects of test generation, including their ability to generate valid test cases and trigger out-of-bounds episodes [6].

Pathrudkar et al. introduced the SceVar (Scenario Variations) database to enhance the realism of simulations. This database improves autonomous vehicle testing by generating realistic variations of scenarios based on real-world driving data. These variations enable a wide range of scenarios for simulation-based regression testing. The SceVar database significantly advances autonomous vehicle testing by incorporating key features such as semantic data models, real-world statistical analysis, statistical insights, and smart sce-

nario variations. Leveraging semantic data models, it comprehensively represents static and dynamic entities in the road-traffic ecosystem, fostering intuitive interactions. Through the analysis of real-world driving data, the database extracts authentic traffic patterns, enabling the creation of numerous realistic scenario variations for simulation-based testing. The integration of statistical analysis allows for insights, pattern identification, and a focused reduction of the simulation state space. Moreover, the database facilitates the creation of smart scenario variations based on statistical insights, covering a wide range of test cases while minimizing the number of simulation runs. Collectively, these features contribute to the efficiency, accuracy, and cost-effectiveness of autonomous vehicle testing in simulation-based environments [20].

Chapter 4
METHODOLOGY

The research methodology carried out in this study is depicted in Fig. 1.1. This chapter will further discuss the implementation and evaluation of the thesis.

4.1 Methods

In terms of practical application, a Python script was implemented that accepts user input and transfers it to ChatGPT. The response from ChatGPT will then be transformed into road nodes, which will subsequently undergo interpolation processes. Following this, the program will evaluate the validity of the designed road, ensuring they do not contain overly steep gradients or excessively sharp turns and do not overlap. If the road is validated successfully, it will be integrated into BeamNG.tech, a physics-based vehicle simulation environment. The roads were then evaluated in a user study to identify whether they aligned with the description specified by the initial user input.

4.2 Implementation

A collection of Python libraries was used to implement this project. The OpenAI library enabled communication with ChatGPT, which is essential for user-AI model interaction. The BeamNGpy library was used to construct detailed road simulations. The task of interpolating road nodes is handled by the NumPy and SciPy libraries, which are renowned for their superior mathematical and scientific computation capabilities. As I built this program, I drew inspiration from the tool AsFault [7], as well as the methodologies from SBFT's Cyber-Physical Systems (CPS) testing competition [24]. This integrated and inspired approach will help ensure an efficient, thorough implementation and analysis of the resulting roads.

4.3 Evaluation

For the evaluation of the project, I gathered data corresponding to the generated roads and conducted a statistical analysis to determine the extent to which they meet the specified requirements. This data collection involved assessing multiple factors such as:

- validity of the road
- alignment between the generated road and the given prompt
- diversity of generated outputs using identical or similar prompts

The assessment of output validity was systematically acquired during the generation of the roads through an implemented script. Quantitative data regarding the alignment between the generated output and the given prompt, as well as the diversity of the generated outputs, was collected through a user study.

By evaluating these parameters, I was able to quantify the proportion of roads that fulfill the given requirements versus those that do not. The chi-squared test, as explained in 4.3.1, was used to determine whether there is a statistically significant difference in the number of valid roads generated as the prompts become more detailed. Furthermore, to assess the performance of ChatGPT in generating road descriptions, I will analyze how frequently it produces identical or similar roads when provided with vague descriptions. This analysis will shed light on the model's ability to generate diverse and contextually appropriate road-related responses.

4.3.1 Chi-Squared Test

The chi-square test is a statistical method used to determine whether there is a significant association between two categorical variables. It assesses whether the observed distribution of categories for one variable differs from what would be expected if the two variables were independent [5].

To conduct a chi-square test, categorical data is organized into a contingency table, with rows representing one variable and columns representing the other. The observed frequencies of categories in the table are then compared to the expected frequencies, which are calculated based on the assumption of independence between the variables [5].

The test statistic is computed by summing the squared differences between observed and expected frequencies, divided by the expected frequencies [5].

The dimensions of the contingency table determine the degrees of freedom for the chi-square test. Once the test statistic is calculated, its significance level (p-value) is assessed. If the computed chi-square statistic is greater than a critical value at a chosen significance level, the null hypothesis of independence is rejected, suggesting a significant relationship

between the variables. On the other hand, if the computed statistic does not exceed the critical value, the null hypothesis is accepted [5].

Chapter 5
IMPLEMENTATION

This section covers the implementation of the tool, the technologies used, and the code's organization. It also outlines the tool's purpose and its target users.

5.1 Purpose and Intended Audience

The tool targets researchers and developers involved in testing self-driving vehicles. Its purpose is to streamline road creation for testing these cars, eliminating the need for users to write road generation algorithms. Instead, they can simply provide textual descriptions of the roads they want to create.

5.2 Technologies

5.2.1 Python

RoadGPT is written in Python, an *"interpreted, object-oriented, high-level programming language with dynamic semantics"* [22]. Python enables rapid development across different domains, from web development to data analysis and machine learning. With its extensive standard library and numerous third-party packages, it helps programmers solve complex problems efficiently [22]. The main libraries used for the tool are the following:

- **OpenAI**: provides easy access to the OpenAI API and is used to communicate with ChatGPT [18].
- **NumPy**: an open-source library used for working with numerical data in Python. It includes a wide variety of mathematical functions [14].
- **SciPy**: a collection of mathematical algorithms and functions built on NumPy [26].

- **BeamNGpy**: a package that provides a Python API to BeamNG.tech [2].

5.2.2 OpenAI API

The OpenAI API is used to enhance user road descriptions by returning more detailed descriptions in a standardized format, simplifying further processing. I selected the GPT-3.5 turbo model for this task because it supports enforced JSON responses. At the time of writing, the model is priced at $0.0005 per 1000 tokens for inputs and $0.0015 per 1000 tokens for outputs. Tokens represent pieces of words, with 1000 tokens approximately equaling 750 words [19].

5.2.3 BeamNG.tech

BeamNG.tech has been chosen as the simulator for several reasons. Firstly, the BeamNGpy package simplifies scenario creation and data extraction for visualization. Secondly, the use of mesh roads enables the generation of 3D roads. Lastly, BeamNG.tech features a realistic physics simulation, ensuring accurate vehicle behavior and environmental interactions. Together, these attributes make BeamNG.tech a comprehensive and reliable option for simulation tasks.

5.3 RoadGPT

RoadGPT is a command line tool that can be called with different parameters. The code is based on the SBFT tool competition [24].

- **beamng-home**: The location of the BeamNG.tech executable.
- **beamng-user**: A directory where levels and other BeamNG-related data will be copied
- **model**: A parameter specifying whether to use OpenAI's assistants or the completions API. Assistants use instructions to tune their personalities and capabilities [18]. Once created, users can access existing assistants by the latter's ID. Completion models take a list of messages as input and return an answer generated by the model [18]. In the case of RoadGPT, these messages are just the system message, giving the model some context, ground rules, and the user's road description.

5.3.1 OpenAI API Connection

After the user provides the road description and specifies the generation budget (the number of road descriptions to be obtained from ChatGPT), RoadGPT initiates an API call using the provided parameters. This API call includes a system prompt, a prompt specified

by the user, and defines the response format as JSON. The system prompt serves as the initial instruction for the model, guiding it on what to generate based on the user's input.

The system prompt used is:

"You are a road designer, who designs roads to test the lane keeping functionality of self driving vehicles. People give you descriptions of roads and you create more detailed descriptions of novel roads, where you split the road into segments. These descriptions are then turned into coordinates, so keep the coordinate values in mind. Since you are testing the lane keeping functionality of self driving vehicles, you want to stress it.

Here are some ground rules:

- *The roads should be diverse, that means given the same description don't create the same road (Start in different directions, etc.)*
- *The car should cover as many directions on the map as possible / The car should face as many directions as possible while using your road.*
- *before the first segment give me a starting point (x, y, z), consider your starting point when you build the road since the x and y values are not allowed to be less than 0 or greater than 200*
- *Give me an angle between 0 and 360 to show which way the road is starting. (0 = along the y axis). the value has to be an integer*
- *make sure that no point is out of bounds / every x and y value is greater than 0 and less than 200*
- *the z axis can't be lower than -28.0 every segment needs the distance of the end point of the segment to the end point of the previous segment in meters, direction (e.g. right turn), the incline in % and the degrees of the turn!*
- *Only return 'left', 'right' or 'straight' for the direction*
- *Only return numbers for the distance, incline and the degrees of the turn.*
- *Write declines in height and the degrees of right turns as negative numbers!*
- *The maximum incline is 15%*
- *The turn degrees of one segment should not exceed 70 degrees. If you want to create a turn with more degrees split it into two segments!*
- *Return the road description in json format of {'starting_point': (x, y, z), 'theta': theta, 'road_segment1': {'distance': int, 'direction': str, 'incline': int, 'turn_degrees': int}, 'road_segment2': {'distance': int, 'direction': str, 'incline': int, 'turn_degrees': int}}, etc.*

- *Given the same prompt you should never return the same road description and your descriptions should be as diverse as possible (don't start with the same theta every time, don't start with a right turn every time, don't start at the same point, etc.)*
- *Make the roads as diverse as possible (longer turns, shorter turns, incline, decline, etc.)*
- *Keep track of where you are going since you are not allowed to go outside of the map boundaries!*"

The first paragraph provides the necessary context for ChatGPT by explaining the task and delineating the intended usage of the generated output. It also states the requirement for output in JSON format to prevent the model from adding unnecessary whitespaces until it reaches the token limit. Following this paragraph are guidelines for the model, outlining crucial details such as the maximum slope and presenting a structure for the JSON output. Additionally, the guidelines specify the types of values inside the JSON structure, which is important to ensure the correct data types for the next steps of the process. The system prompt and the user prompt are merged and forwarded to the GPT-model 3.5 turbo. The model generates output resembling the following example:

{ "starting_point": [50, 30, 0], "theta": 90, "road_segment1": {"distance": 50, "direction": "right", "incline": 5, "turn_degrees": 40}, "road_segment2": {"distance": 40, "direction": "left", "incline": 3, "turn_degrees": -20}, "road_segment3": {"distance": 30, "direction": "straight", "incline": 2, "turn_degrees": 0} }

The *starting_point* denotes a random location on the map selected by ChatGPT as the initial point of the road. *Theta* represents the initial direction in which the road is oriented.

Every *road_segment* holds data essential for determining the subsequent node. *Distance* indicates the length to the next node in meters, *direction* specifies left, right, or straight, *incline* represents the road slope in percentage, and *turn_degrees* denotes the angle of the turn in degrees.

The output is used in another part of the tool to translate the dictionary information into road nodes.

5.3.2 Translation to Nodes

In the road construction process, each segment's description is used to calculate the subsequent node. Beginning from the starting point and orientation (theta), a new node is computed at a distance of 5 meters ahead, maintaining the same altitude to prevent potential issues with cars glitching and spawning within the road.

Following this, the new orientation of the road is calculated by adding the turn angle to the current road orientation for a left turn or subtracting it for a right turn. The road orientation is then converted to radians for node calculation. These computations are then

applied to determine the x- and y-coordinates of the next road node using the equations:

$$x_n = x_p + distance * cos(orientation)$$

and

$$y_n = y_p + distance * sin(orientation)$$

Additionally, the altitude of the next node is determined by the following formula:

$$z_n = z_p + distance * (incline/100)$$

The variables x_n, y_n and z_n correspond to the coordinates of the new node, while x_p, y_p and z_p represent those of the previous node. $distance$ denotes the distance between nodes in meters, $orientation$ signifies the current road orientation in radians, and $incline$ indicates the slope of the road segment in percentage. Following this, the road nodes are interpolated before the validation stage.

5.3.3 Validation of the Road

Before the roads are constructed in the simulator, they undergo validation and must meet the following criteria:

- **The road is within the map boundaries**: The created road must stay within the map's bounds, which, even though the prompt indicates a map size of 200 x 200, is actually 250 x 250 to offer the model a bit of flexibility. Two checks are conducted for this: First, the distance between the two furthest points must be under 250. Second, no part of the road is permitted to intersect the map boundaries (0 and 250).

- **The road is not self-intersecting**: To identify if a road is self-intersecting, it is divided into polygon shapes. These shapes are then examined to ensure that there are no overlaps between non-adjacent ones.

- **The turns are not too sharp**: During this evaluation, the tool calculates the radius of each road turn and verifies if the smallest radius is greater than a specified threshold.

- **The slopes are not to steep**: In evaluating slope steepness, the tool analyzes the altitude of each node and its distance from the previous node. It then computes the average change in altitude over a set window size to determine if the average change in altitude is within a predefined incline threshold. For instance, using a window size of 5, the tool computes the average incline of 5 consecutive nodes and verifies whether it remains under the specified incline threshold.

Should a road fail to meet any of the criteria, the tool generates an error specifying the failed test and proceeds to generate the next road, provided the total number of roads to generate has not been reached yet.

If a road meets all the criteria, the tool proceeds to construct the simulation.

5.3.4 Simulation and Visualization of the Road

Once all validation steps have been completed, a scenario is constructed within the simulator using the custom map from the SBFT tool competition, which features a flat terrain with a grass texture. The road itself is generated using a Meshroad object, which are *strips of rectangular mesh segments defined by a 3D spline* [1] that align with the road's orientation. Meshroads are used because they can float in the air without any restrictions. Additionally, a Decalroad is placed on top of the Meshroad. BeamNG defines a Decalroad as *a strip-shaped decal defined by a 2D spline. This strip is projected onto the terrain, similarly to how decals are projected on collision meshes* [1]. However, there is a risk of fragmentation, or sections of the decal road rendering beneath the Meshroad, which usually occurs when the x- and y-coordinates of the Decal- and Meshroads' nodes are identical. To avoid this issue, random noise is introduced to the x- and y-coordinates of the Meshroad's nodes.

Once the roads are generated, a car is positioned on the road to assess the performance of the self-driving agent. Using the BeamNGpy API, the tool continuously retrieves updates on the vehicle's position. If the car drives out of its lane or reaches the end of the road, the scenario ends, and two plots are generated to visualize the road layout. The first plot, as seen in Figure 5.1, gives a bird's-eye view of the road, showing its curves. The second plot, as seen in Figure 5.2, displays the altitude of the road at a given distance from the starting point.

Upon reaching the generation budget limit, two CSV files are created. The first file contains information regarding the test generation process, such as the total number of generated tests, the count of valid and invalid tests, and the number of tests that passed or failed. The second CSV file provides details about instances where the car drove out of the lane, specifying the occurrences on both the left and right sides.

At the end, the user is asked to either input a new prompt and generation budget or exit the tool.

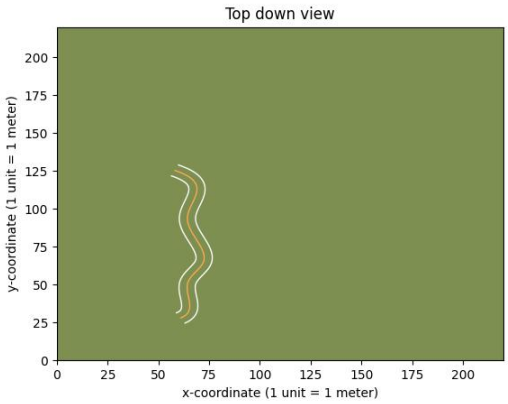

Figure 5.1: Bird's-Eye View of a road

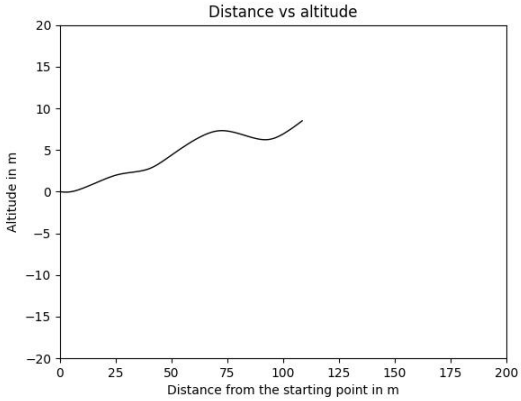

Figure 5.2: Distance against Height

Chapter 6
EVALUATION

For the evaluation of the tool, I used the CSV files to analyze the quantitative data produced by the tool. To assess whether the roads generated by ChatGPT align with the description and whether ChatGPT generates diverse roads with the same prompt, I conducted a user study. The results of the data analysis will be described below.

6.1 Analysis of the CSV Files

To analyze generation statistics, three prompts of varying specificity were used: "a road", "an uphill road with 3 turns", and "a road going uphill for 2 turns, followed by a straight downhill segment, ending in a left turn". A combined total of 300 roads were generated, with 100 roads produced for each prompt. Following the simulations, two CSV files were created for each prompt. One contained generation statistics (number of roads generated, valid/invalid counts), while the other recorded instances of cars driving out of their lanes. Of the 300 roads generated, 182 successfully passed validation, with 118 failing. Among the 182 roads simulated, the car steered out of its lane in 8 instances.

Among the 100 roads generated using the prompt "a road," 88 successfully passed validation, and 1 of those 88 roads resulted in the car driving out of the lane. For the prompt "an uphill road with 3 turns," out of the 100 roads generated, 57 met validation criteria, with 2 of those 57 ending with the car steering out of the lane. Lastly, from the 100 roads generated with the prompt "a road going uphill for 2 turns, followed by a straight downhill segment, ending in a left turn," only 37 successfully passed validation, with 5 out of those 37 roads ending with the car driving out of the lane.

This indicates that when prompts are more generic and the language model has more freedom, it tends to better adhere to the specified rules, such as the map size. However, roads generated from more general prompts seem to place less emphasis on testing the

lane-keeping functionality of the self-driving agent compared to roads generated from more specific prompts.

The chi-squared test was used to determine the statistical significance of these results. Regarding the number of valid roads generated per prompt, the hypotheses were as follows:

H0: the prompt does not influence the number of valid roads generated H1: the prompt does influence the number of valid roads generated

An alpha value of 0.05 was chosen.

First, a contingency table (6.1) was created to display the counts of valid and invalid roads generated for each prompt. This table was then used to derive the expected values table (6.2), calculated using the formula $\frac{row\ total * column\ total}{grand\ total}$ for each cell. With these two tables, the chi-square score was computed by applying the formula $\frac{(observed\ value - calculated\ value)^2}{calculated\ value}$ to each element and summing the results 6.3.

The degrees of freedom are determined by the formula $(number\ of\ rows - 1) * (number\ of\ columns - 1)$. The critical chi-square value can be found in the chi-square distribution table in Appendix A.1, using the degrees of freedom and the alpha value. Since the calculated chi-square value is 55.35, the critical chi-square value is 5.99, and 55.35 is greater than 5.99, the null hypothesis H0 is rejected. This indicates that the prompt does influence the number of valid roads generated.

In the following tables, the prompt "a road" is referred to as prompt 1, the prompt "an uphill road with 3 turns" is referred to as prompt 2, and the prompt "a road going uphill for 2 turns, followed by a straight downhill segment, ending in a left turn" is referred to as prompt 3.

	valid	invalid	total
prompt 1	88	12	100
prompt 2	57	43	100
prompt 3	37	63	100
total	182	118	300

Table 6.1: Road Validity Contingency Table

	valid	invalid	total
prompt 1	60.67	39.33	100
prompt 2	60.67	39.33	100
prompt 3	60.67	39.33	100
total	182	118	300

Table 6.2: Road Validity Expected Values Table

	observed	calculated	$\frac{(o-c)^2}{c}$
	88	60.67	12.32
	57	60.67	0.22
	37	60.67	9.32
	12	39.33	18.99
	43	39.33	0.34
	63	39.33	14.24
total			55.35

Table 6.3: Road Validity Chi-Square Calculation

6.2 Survey

For the user study, I distributed a survey to 11 individuals engaged in research or work related to autonomous vehicles. Of these, ten respondents provided answers.

6.2.1 Survey structure

The survey was split into three main sections. The first section aimed to gather demographic information from the participants. The second section asked participants to rate the alignment between a road and the prompt it was generated with. To assist in their evaluation, participants were provided with a bird's-eye view plot, an altitude plot, and two videos depicting a car driving on the road. The first video displayed the car from a third-person perspective, while the second one provided a view from the car's bonnet. In the final section, participants were asked to rate the similarity between two roads generated using the same prompt. For this task, the participants were provided with the altitude plot and the bird's eye view of each road stacked on top of each other, alongside two videos featuring a split-screen view of a car driving on both roads simultaneously. The first split-screen video showed the two cars from a third-person view, while the second one provided a view from the car's bonnets.

Given that all roads used in the user study fall within the coordinate range of 0 to 220, the axes of the bird's-eye view plot are set to span from point 0 to 220, with each unit on the plot representing 1 meter. The axis dimensions were selected to offer a neutral perspective of the roads, avoiding any emphasis on the turns of one road over another. Given that the maximum altitude and longest road length are restricted to 20 and 200 meters, respectively, these factors determine the scaling of the axes for visualizing distance and height in the plot.

6.2.2 Survey results

In this section, I will describe the results of the survey by section.

Demographic Data

The participants were asked about their age, occupation, possession of a driver's license, number of countries they have driven in, prior experience with generative AI, and familiarity with driving simulators.

Four participants fall within the age range of 20 to 29 years, five are aged between 30 to 39, and one participant is aged between 40 to 49 (see Figure 6.1). Nine participants hold a driver's license, while one does not (see Figure 6.2). Each participant has driven in at least one country (see Figure 6.3). Seven participants use generative AI regularly, while three reported using it occasionally (see Figure 6.4). Eight participants use driving simulators regularly, one uses them occasionally, and one participant has never used a driving simulator before (see Figure 6.5).

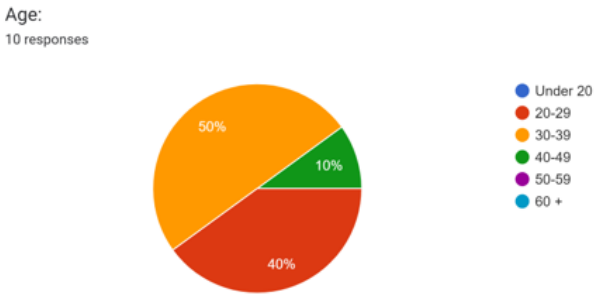

Figure 6.1: Age

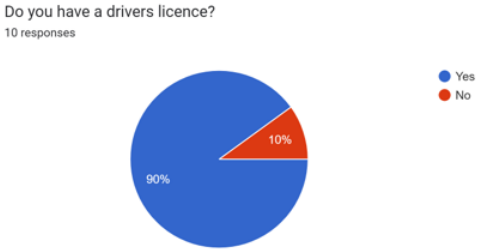

Figure 6.2: Driver's license

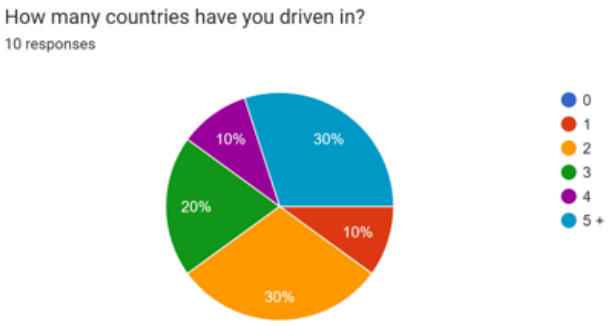

Figure 6.3: Countries driven in

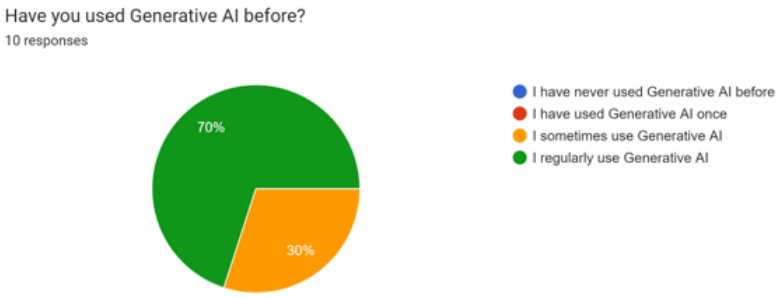

Figure 6.4: Generative AI

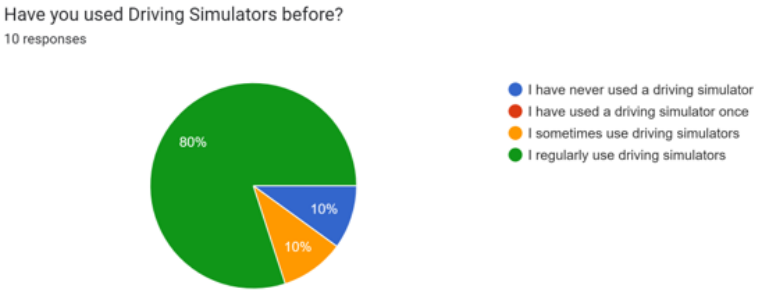

Figure 6.5: Driving Simulator

Road-Prompt Alignment

In the section concerning Road-Prompt Alignment, participants are tasked with assessing how well a given road corresponds to a provided prompt. This evaluation involves two main components: the overall suitability of the road and its adherence to specific criteria, such as the number of turns, incline, and sequence/order of turns. For visual reference, the participants were presented with an image showing the road plot from both a bird's-eye view and one illustrating altitude side by side as depicted in Figure 6.6. Additionally,

users had access to two videos: one featuring a car driving on the road from a third-person perspective (see Figure 6.7) and another showing the view from a camera mounted on the car's bonnet (see Figure 6.8).

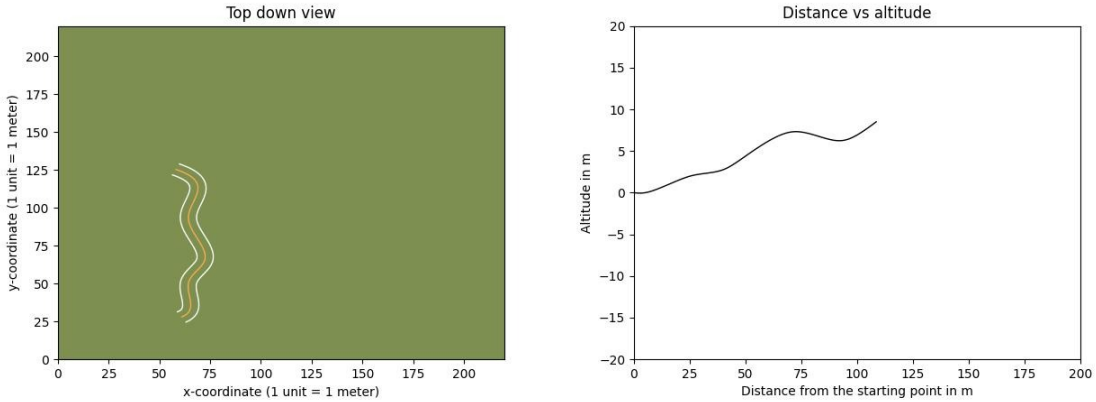

Figure 6.6: Visualization of a road generated with the prompt "a serpentine road"

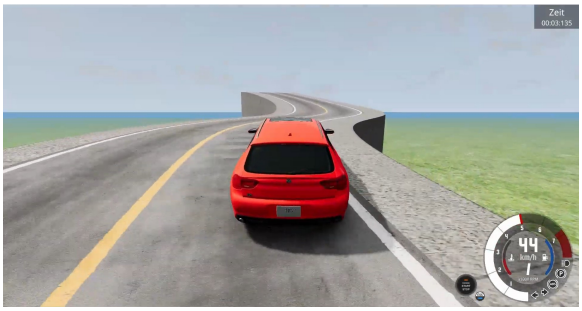

Figure 6.7: Screenshot from a Video with a Third Person Perspective of a car driving on the road

Figure 6.8: Screenshot from a Video with a Bonnet Perspective of a car driving on the road

Four prompts were used for road creation: "a serpentine road", "an uphill road with 3 turns", "a road going uphill for 3 turns and downhill for 2 turns", and "a road going uphill for 2 turns, followed by a straight downhill segment, ending with a left turn." These prompts were

chosen to transition from general to more specific. The participants evaluated how well the visualized road matched the respective prompt in both overall fit and specific criteria. Additionally, participants were given the option to provide extra feedback for each road. The participants were required to select from the options "does not match the description at all", "generally does not match the description", "somewhat matches the description", "mostly matches the description", and "perfectly matches the description" in order to cast their votes.

The prompt "a serpentine road" was the most generic prompt used for road generation (see figure 6.6). Therefore, participants were only required to rate their overall fit with the prompt. As depicted in figure 6.9, while one participant disagreed, the majority agreed that the road aligned well. Five out of ten participants rated it as a perfect match, one as mostly matching, and three as somewhat matching.

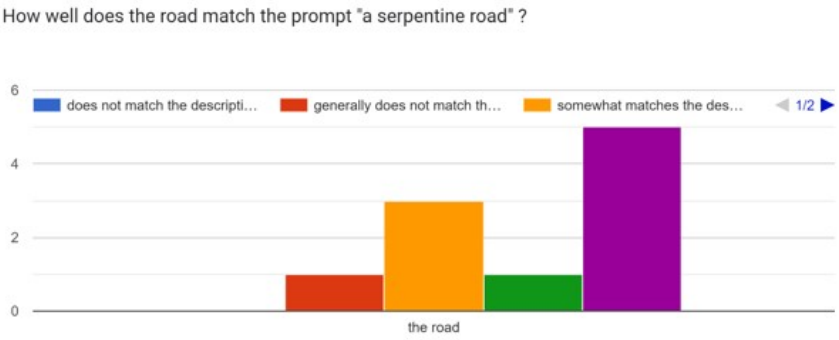

Figure 6.9: Results of the general road-prompt alignment for "a serpentine road"

The second prompt used was "an uphill road with 3 turns" (see figure 6.10), which was more detailed as it specified both the incline and the number of turns. Participants indicated that the road generally aligned with the prompt (see figure 6.11). Regarding the incline, the majority voted that it perfectly matched the prompt, while opinions were divided on the number of turns, with more participants indicating somewhat or mostly matching (see figure 6.12). Multiple respondents provided additional feedback, noting a turn starting at the road's end.

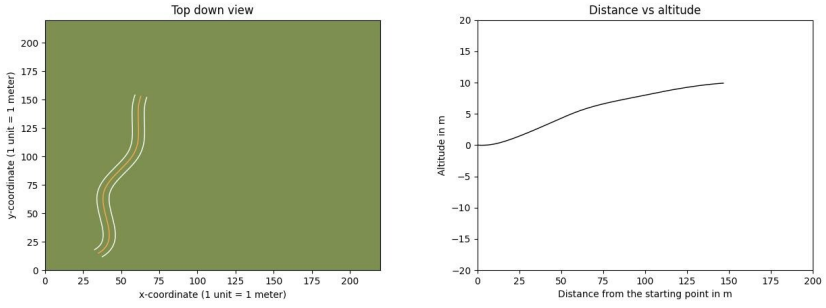

Figure 6.10: Visualization of a road generated with the prompt "an uphill road with 3 turns"

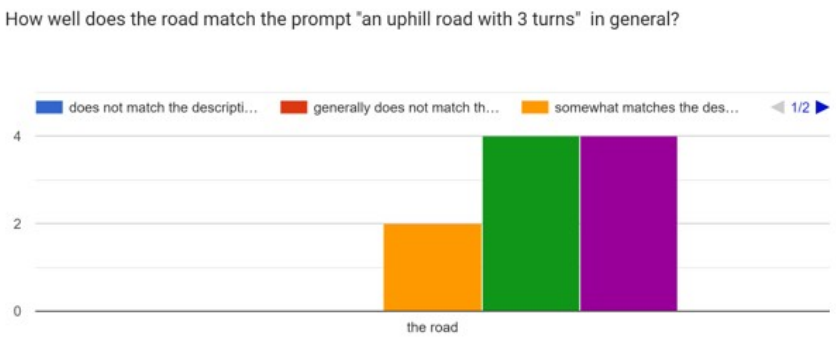

Figure 6.11: Results of the general road-prompt alignment for "an uphill road with 3 turns"

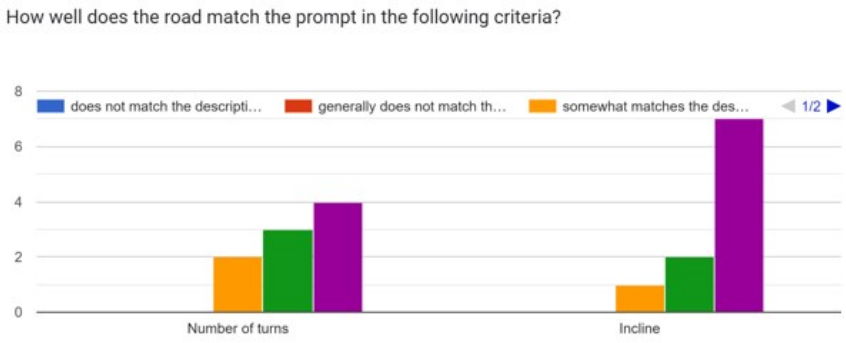

Figure 6.12: Results of the road-prompt alignment in specific criteria for "an uphill road with 3 turns"

The prompt "a road going uphill for 3 turns and downhill for 2 turns" (see figure 6.13) introduced an additional level of specificity by incorporating changes in incline. Although participants generally agreed that the road aligned with the prompt (see figure 6.14, closer inspection of specific aspects revealed mismatches. While the incline and the number of uphill turns closely matched the prompt, the count of downhill turns did not (figure 6.15). In the additional feedback, several participants highlighted the absence of a downhill turn. One participant noted that while the LLM interprets the incline correctly, "counting" the turns does not seem to be a task that it performs well in.

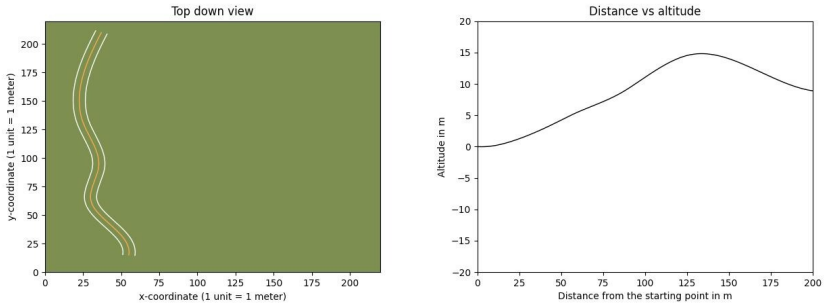

Figure 6.13: Visualization of a road generated with the prompt "a road going uphill for 3 turns and downhill for 2 turns"

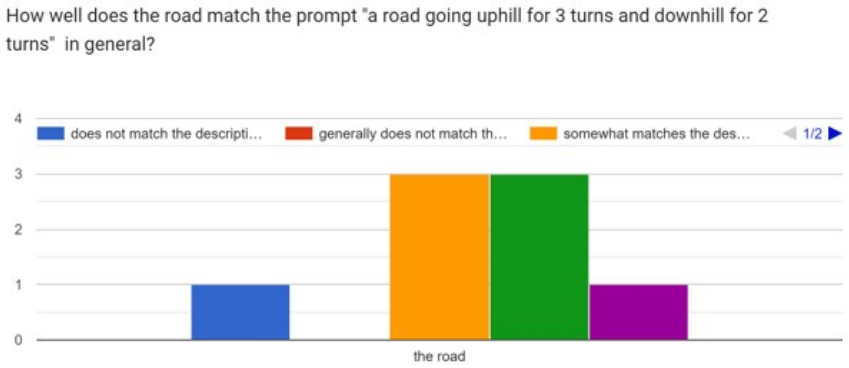

Figure 6.14: Results of the general road-prompt alignment for "a road going uphill for 3 turns and downhill for 2 turns"

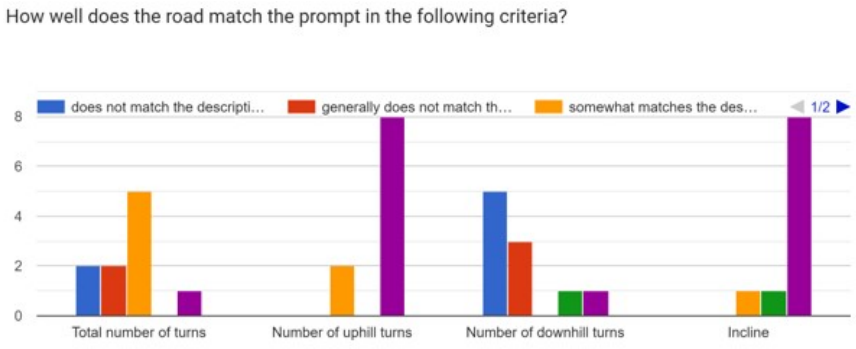

Figure 6.15: Results of the road-prompt alignment in specific criteria for "a road going uphill for 3 turns and downhill for 2 turns"

The most detailed prompt used is "a road going uphill for 2 turns, followed by a straight downhill segment, ending in a left turn" (figure 6.16). The prompt introduces a sequence for the road segments. Although most participants agreed that the road matched the prompt (figure 6.17), some were unsure about the total number of turns 6.18. In additional comments, two participants mentioned the difficulty in determining whether there was only one

turn or two in the uphill segments. Meanwhile, one participant expressed uncertainty about the presence of an additional turn in the uphill segment.

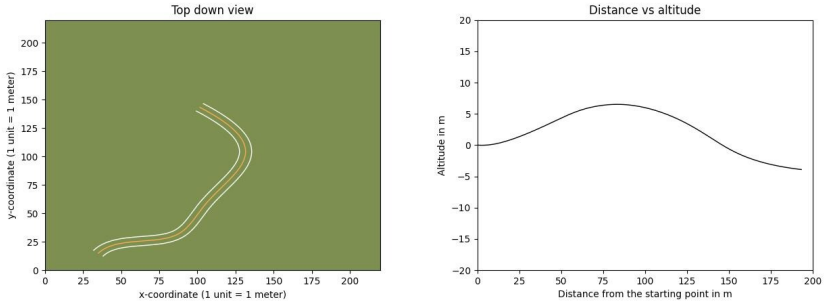

Figure 6.16: Visualization of a road generated with the prompt "a road going uphill for 2 turns, followed by a straight downhill segment, ending in a left turn"

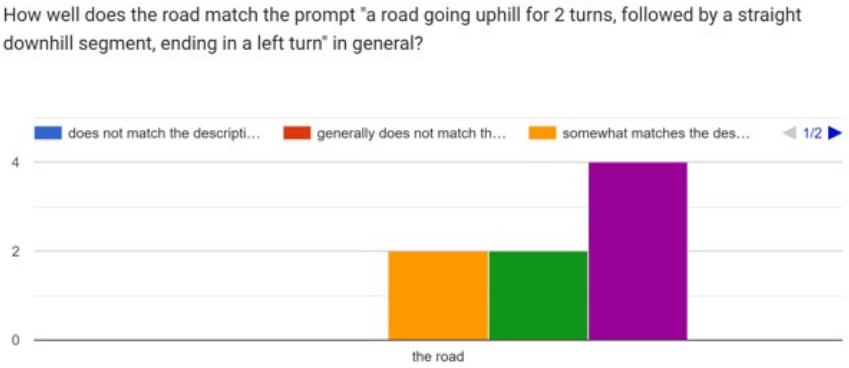

Figure 6.17: Results of the general road-prompt alignment for "a road going uphill for 2 turns, followed by a straight downhill segment, ending in a left turn"

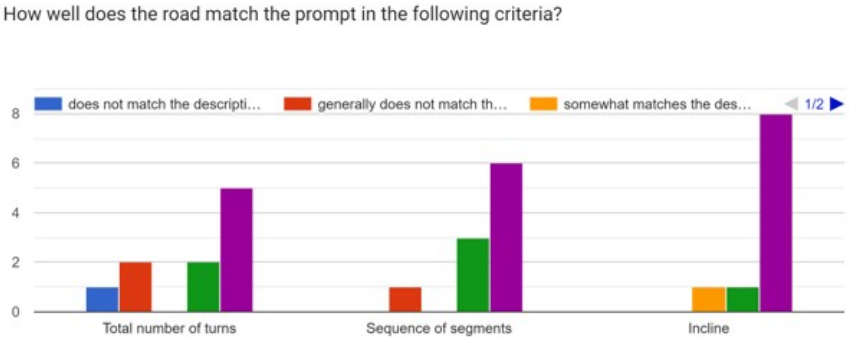

Figure 6.18: Results of the road-prompt alignment in specific criteria for "a road going uphill for 2 turns, followed by a straight downhill segment, ending in a left turn"

Reviewing the results, it seems that, as one user pointed out, the LLM performs well at interpreting the incline of the prompts but struggles with accurately identifying the specified number of turns.

Road Similarity

In the Road Similarity section, participants were presented with pairs of roads to evaluate their resemblance to each other. Visualizations presented both road plots, one above the other, as shown in figure 6.19. Additionally, participants were shown split-screen videos featuring cars driving along both roads simultaneously, with one providing a third-person perspective (figure 6.20) and the other showing the car's perspective from the bonnet (figure 6.21). Participants were then prompted to rate the general similarity between the roads and their similarity in specific criteria. Ratings ranged from "definitely different" to "basically the same road" for general similarity and from "definitely different" to "(almost) identical" for specific criteria.

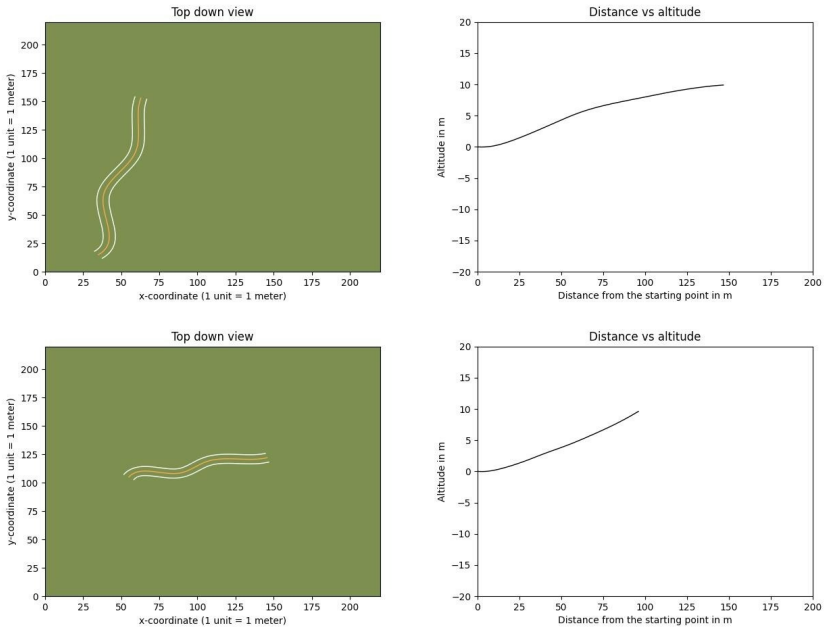

Figure 6.19: Comparison of 2 roads generated with the prompt "an uphill road with 3 turns"

Figure 6.20: Screenshot from a Split-Screen Video with a Third Person Perspective

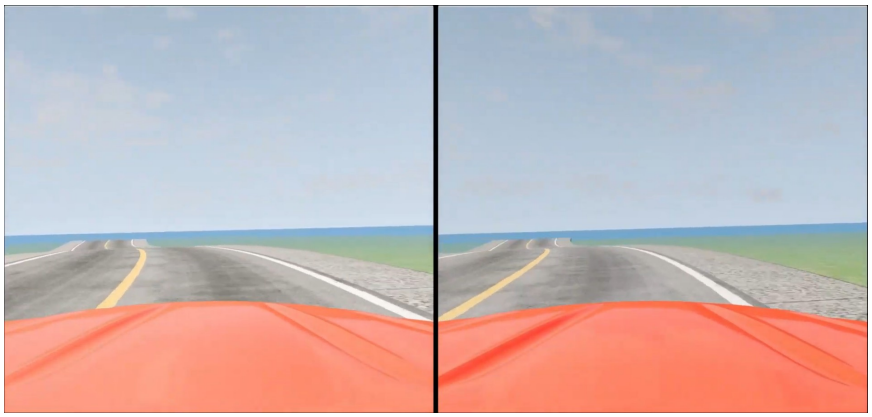

Figure 6.21: Screenshot from a Split-Screen Video with a Bonnet Perspective

The participants generally found the first two roads (figure 6.19), which were generated with the prompt "an uphill road with 3 turns", to be dissimilar, as illustrated in Figure 6.22. Although there was agreement on the similarity in the number of turns, differences were noted in the sequence of turns and the length of the road, which were distinctly perceived as dissimilar. Opinions on the road's incline varied, with three participants rating it as identical while another three rated it as mostly dissimilar (see figure 6.23).

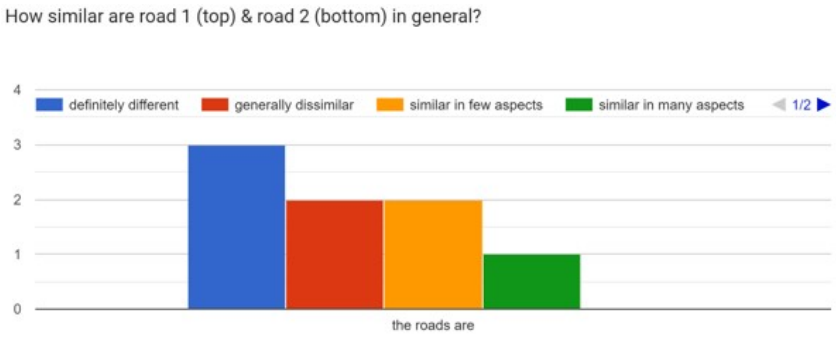

Figure 6.22: General similarity of the two roads generated with the prompt "an uphill road with 3 turns"

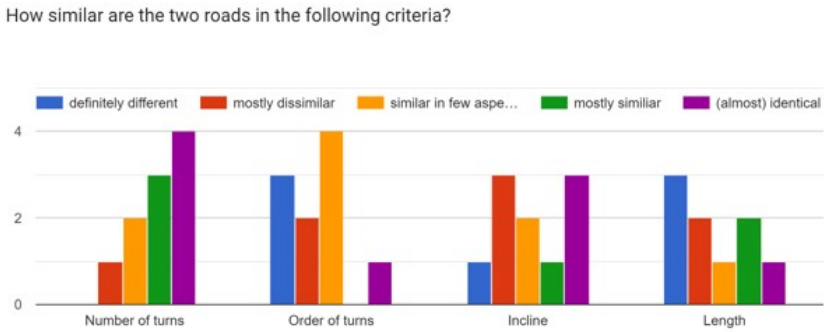

Figure 6.23: Similarity in certain criteria of the two roads generated with the prompt "an uphill road with 3 turns"

The second pair of roads, generated with the prompt "a road" (figure 6.24), were perceived as identical (figure 6.25). Participants rated the roads as identical across all criteria (figure 6.26). Several participants noted in the additional comments that the only difference was the orientation of the roads in the bird's eye view plot.

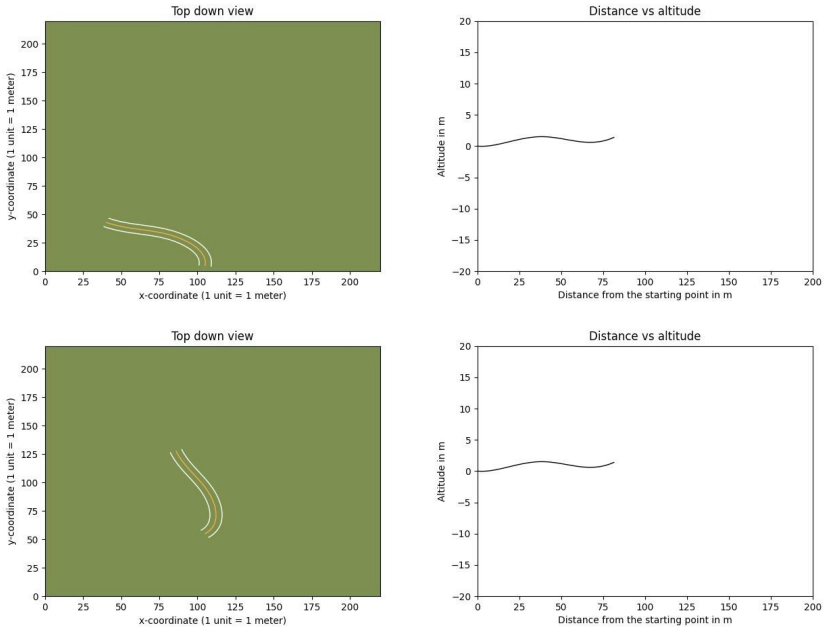

Figure 6.24: Comparison of 2 roads generated with the prompt "a road"

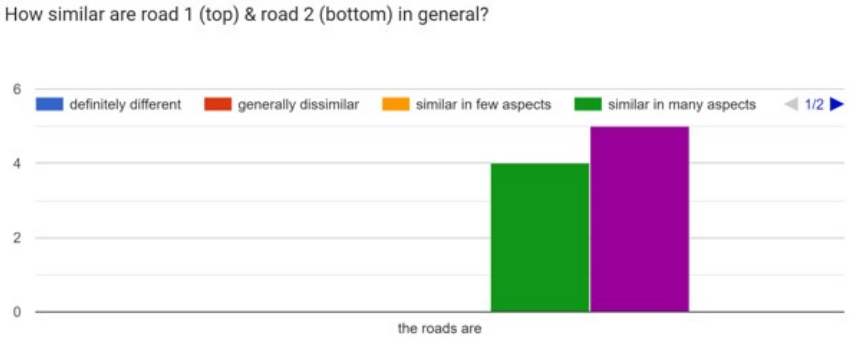

Figure 6.25: General similarity of the two roads generated with the prompt "a road"

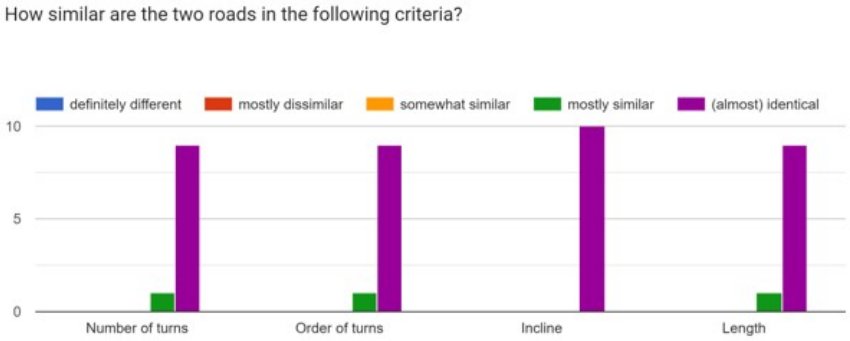

Figure 6.26: Similarity in certain criteria of the two roads generated with the prompt "a road"

In conclusion, the LLM is proficient in generating roads that meet specific criteria outlined in the prompts. While it accurately interprets inclines, determining the number of turns remains a challenge. When presented with identical prompts, ChatGPT sometimes produces roads that are almost identical, with the main difference being their orientation.

6.3 Conclusion

This section summarizes the results to address the research question **Can generative AI be used to automatically create three-dimensional roads for testing autonomous vehicles?**

The results imply that Generative AI has the potential to create virtual roads that are appropriate for testing autonomous vehicles. ChatGPT was used due to its accessibility and seamless integration via the OpenAI API. Despite the model not being trained for this particular use case, it produced roads that met most requirements. The user study indicated that while ChatGPT performed well in interpreting road inclines, it struggled with generating the correct number of turns and stressing the lane-keeping functionality of the self-driving agent. Since the model is not specifically trained or designed for this task, potential solutions to these issues may include the creation of a labeled dataset for fine-tuning or the training of a new model.

6.4 Open Issues and Possible Improvements

The evaluation of the tool revealed several issues. Firstly, it was found that the more specific the prompt, the fewer roads passed validation checks. Secondly, the generated roads lack scenarios that effectively challenge the lane-keeping functionality of autonomous vehicles. Additionally, ChatGPT struggled to generate the correct number of turns accurately. As ChatGPT is not explicitly trained for road generation, these issues might be resolved by compiling a labeled dataset with road descriptions as labels and the anticipated LLM output for fine-tuning an existing model, or by training a new model altogether.

A participant in the user study pointed out that *It would match better with the prompt if the generated road had straight segments at the start and end (which can be part of the prompt), so the number of turns was more obvious. Now, the road starts in the middle of the turn, and it has a small turn at the end of the road, too.* This feedback should be considered for future work stemming from this project.

Chapter 7
SUMMARY

The objective of this thesis was to create a tool utilizing generative AI to generate virtual roads for testing self-driving vehicles. The system was designed to streamline the process of testing autonomous vehicles by converting textual road descriptions into coordinate-based nodes. These nodes are used to construct the road within the simulation environment. This approach eliminates the need to write complicated algorithms.

The tool's implementation involved selecting appropriate technologies, such as the programming language, the chosen LLM, and the simulator. The evaluation comprised a quantitative analysis of the number of generated roads and instances where the car deviated from its lane, alongside a survey gathering feedback from target users regarding the generated roads.

Although the model didn't effectively stress the lane-keeping functionality, it's important to note that it wasn't trained for that purpose. Nevertheless, the model generally produced roads that aligned with the provided descriptions.

In conclusion, the tool developed in this thesis presents a promising solution for generating virtual roads using generative AI. The study results suggest that the tool holds the potential to serve as a foundation for future research in this field.

7.1 Future work

While this study established the groundwork for using generative AI in road generation, there are still many areas to explore to enhance the performance of AI models in this task.

Fine-Tuning a model: By compiling a dataset containing road descriptions paired with descriptive prompts as labels, it is possible to fine-tune a model specifically for this purpose and enhancing its performance. This dataset can be constructed either by labeling roads generated by an existing tool or by manually generating roads based on provided prompts.

Training a new model: Training a new model allows for fine-tuning it for specific purposes, ensuring adherence to all relevant rules. This can be achieved by training the model on either artificially generated roads or real roads. While training on generated roads can effectively stress test the lane-keeping functionality of the self-driving agent, training on real-world roads ensures that the model closely mimics real-world scenarios.

Terrain modification: At the time of writing, BeamNgpy introduced new functionality allowing users to import heightmaps and enable terrain modification. This introduces a challenge as road sections might be obscured by terrain. Moreover, when combined with traffic, it not only tests the car's lane-keeping capabilities but also its ability to avoid collisions.

Road Networks: Expanding the functionality of road generation tools to encompass entire road networks, including features such as junctions, offers numerous advantages. It provides a more comprehensive evaluation of autonomous vehicles in varied driving environments, testing their ability to navigate complex intersections and merge lanes. Including diverse lane markings enhances realism, resulting in more precise performance evaluations and better preparation for real-world deployment. In essence, shifting from individual roads to complete road networks enhances testing precision and helps in the development of safer autonomous driving systems.

Bibliography

[1] BeamNG GmbH. Beamng documentation. [Online]. Available: https://documentation.beamng.com/

[2] ——. Beamngpy documentation. [Online]. Available: https://beamngpy.readthedocs.io/en/latest/readme.html

[3] ——. Soft-body physics. [Online]. Available: https://www.beamng.com/game/about/physics/

[4] K. R. Chowdhary, *Natural Language Processing*. New Delhi: Springer India, 2020, pp. 603–649. [Online]. Available: https://doi.org/10.1007/978-81-322-3972-7_19

[5] D. Freedman, *The Chi-Square Test*. W.W. Norton, 2007, p. 523–544.

[6] A. Gambi, G. Jahangirova, V. Riccio, and F. Zampetti, "Sbst tool competition 2022," in *Proceedings of the 15th Workshop on Search-Based Software Testing*, ser. SBST '22. New York, NY, USA: Association for Computing Machinery, 2023, p. 25–32. [Online]. Available: https://doi.org/10.1145/3526072.3527538

[7] A. Gambi, M. Mueller, and G. Fraser, "Asfault: Testing self-driving car software using search-based procedural content generation," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2019, pp. 27–30.

[8] ——, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 318–328. [Online]. Available: https://doi.org/10.1145/3293882.3330566

[9] W. Huang, K. Wang, Y. Lv, and F. Zhu, "Autonomous vehicles testing methods review," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 163–168.

[10] IBM. What is artificial intelligence (ai)? [Online]. Available: https://www.ibm.com/topics/artificial-intelligence

[11] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Comput. Surv.*, vol. 54, no. 10s, sep 2022. [Online]. Available: https://doi.org/10.1145/3505244

[12] W. M. Lim, A. Gunasekara, J. L. Pallant, J. I. Pallant, and E. Pechenkina, "Generative ai and the future of education: Ragnarök or reformation? a paradoxical perspective from management educators," *The International Journal of Management Education*, vol. 21, no. 2, p. 100790, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1472811723000289

[13] Microsoft. Introduction to prompt engineering. [Online]. Available: https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/prompt-engineering#basics

[14] NumPy. [Online]. Available: https://numpy.org/doc/stable/user/absolute_beginners.html

[15] OpenAI. [Online]. Available: https://openai.com/blog/chatgpt

[16] ——. [Online]. Available: https://openai.com/dall-e-2

[17] OpenAI. How chatgpt and our language models are developed. [Online]. Available: https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-language-models-are-developed

[18] ——. Openai api documentation. [Online]. Available: https://platform.openai.com/docs/introduction/key-concepts

[19] ——. Pricing. [Online]. Available: https://openai.com/pricing#language-models

[20] S. Pathrudkar, S. Mukherjee, V. Sarathi, and M. Chowdhary, "Scevar (scenario variations) database: Real world statistics driven scenario variations for av testing in simulation: Abstraction of static and dynamic entities from road network-traffic ecosystem and their interactions or relationships in semantic data models for realistic simulation-based testing of avs," in *Companion Publication of the 13th ACM Web Science Conference 2021*, ser. WebSci '21 Companion. New York, NY, USA: Association for Computing Machinery, 2021, p. 126–129. [Online]. Available: https://doi.org/10.1145/3462741.3466655

[21] C. U. Press. The Cambridge Dictionary of Philosophy. Simulation. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/simulation

[22] Python. [Online]. Available: https://www.python.org/doc/essays/blurb/

[23] SAE International. (2021, Apr) Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. [Online]. Available: https://saemobilus.sae.org/content/j3016_202104

[24] SBFT. Sbft'23 homepage. [Online]. Available: https://sbft23.github.io/

[25] H.-P. Schöner, "Simulation in development and testing of autonomous vehicles," in *18. Internationales Stuttgarter Symposium*, M. Bargende, H.-C. Reuss, and J. Wiedemann, Eds. Wiesbaden: Springer Fachmedien Wiesbaden, 2018, pp. 1083–1095.

[26] Scipy. [Online]. Available: https://docs.scipy.org/doc/scipy/tutorial/index.html#user-guide

[27] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, "Procedural content generation via machine learning (pcgml)," *IEEE Transactions on Games*, vol. 10, no. 3, pp. 257–270, 2018.

[28] Synopsys Inc. The 6 levels of vehicle autonomy explained. [Online]. Available: https://www.synopsys.com/automotive/autonomous-driving-levels.html

[29] Y. Tang, Y. Zhou, K. Yang, Z. Zhong, B. Ray, Y. Liu, P. Zhang, and J. Chen, "Automatic map generation for autonomous driving system testing," 2022.

[30] TWI. What is an autonomous vehicle. [Online]. Available: https://www.twi-global.com/technical-knowledge/faqs/what-is-an-autonomous-vehicle

[31] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," 2023.

[32] E. Winsberg, "Computer Simulations in Science," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta and U. Nodelman, Eds. Metaphysics Research Lab, Stanford University, 2022.

Appendix A

Example Appendix 1

d.f.	.995	.99	.975	.95	.9	.1	.05	.025	.01
1	0.00	0.00	0.00	0.00	0.02	2.71	3.84	5.02	6.63
2	0.01	0.02	0.05	0.10	0.21	4.61	5.99	7.38	9.21
3	0.07	0.11	0.22	0.35	0.58	6.25	7.81	9.35	11.34
4	0.21	0.30	0.48	0.71	1.06	7.78	9.49	11.14	13.28
5	0.41	0.55	0.83	1.15	1.61	9.24	11.07	12.83	15.09
6	0.68	0.87	1.24	1.64	2.20	10.64	12.59	14.45	16.81
7	0.99	1.24	1.69	2.17	2.83	12.02	14.07	16.01	18.48
8	1.34	1.65	2.18	2.73	3.49	13.36	15.51	17.53	20.09
9	1.73	2.09	2.70	3.33	4.17	14.68	16.92	19.02	21.67
10	2.16	2.56	3.25	3.94	4.87	15.99	18.31	20.48	23.21
11	2.60	3.05	3.82	4.57	5.58	17.28	19.68	21.92	24.72
12	3.07	3.57	4.40	5.23	6.30	18.55	21.03	23.34	26.22
13	3.57	4.11	5.01	5.89	7.04	19.81	22.36	24.74	27.69
14	4.07	4.66	5.63	6.57	7.79	21.06	23.68	26.12	29.14
15	4.60	5.23	6.26	7.26	8.55	22.31	25.00	27.49	30.58
16	5.14	5.81	6.91	7.96	9.31	23.54	26.30	28.85	32.00
17	5.70	6.41	7.56	8.67	10.09	24.77	27.59	30.19	33.41
18	6.26	7.01	8.23	9.39	10.86	25.99	28.87	31.53	34.81
19	6.84	7.63	8.91	10.12	11.65	27.20	30.14	32.85	36.19
20	7.43	8.26	9.59	10.85	12.44	28.41	31.41	34.17	37.57
22	8.64	9.54	10.98	12.34	14.04	30.81	33.92	36.78	40.29
24	9.89	10.86	12.40	13.85	15.66	33.20	36.42	39.36	42.98
26	11.16	12.20	13.84	15.38	17.29	35.56	38.89	41.92	45.64
28	12.46	13.56	15.31	16.93	18.94	37.92	41.34	44.46	48.28
30	13.79	14.95	16.79	18.49	20.60	40.26	43.77	46.98	50.89
32	15.13	16.36	18.29	20.07	22.27	42.58	46.19	49.48	53.49
34	16.50	17.79	19.81	21.66	23.95	44.90	48.60	51.97	56.06
38	19.29	20.69	22.88	24.88	27.34	49.51	53.38	56.90	61.16
42	22.14	23.65	26.00	28.14	30.77	54.09	58.12	61.78	66.21
46	25.04	26.66	29.16	31.44	34.22	58.64	62.83	66.62	71.20
50	27.99	29.71	32.36	34.76	37.69	63.17	67.50	71.42	76.15
55	31.73	33.57	36.40	38.96	42.06	68.80	73.31	77.38	82.29
60	35.53	37.48	40.48	43.19	46.46	74.40	79.08	83.30	88.38
65	39.38	41.44	44.60	47.45	50.88	79.97	84.82	89.18	94.42
70	43.28	45.44	48.76	51.74	55.33	85.53	90.53	95.02	100.43
75	47.21	49.48	52.94	56.05	59.79	91.06	96.22	100.84	106.39
80	51.17	53.54	57.15	60.39	64.28	96.58	101.88	106.63	112.33
85	55.17	57.63	61.39	64.75	68.78	102.08	107.52	112.39	118.24
90	59.20	61.75	65.65	69.13	73.29	107.57	113.15	118.14	124.12
95	63.25	65.90	69.92	73.52	77.82	113.04	118.75	123.86	129.97
100	67.33	70.06	74.22	77.93	82.36	118.50	124.34	129.56	135.81

Table A.1: Chi-Square Distribution Table. Adapted from [5]