

4-10-2021

Automated Scenario Generation Using Halton Sequences for the Verification of Autonomous Vehicle Behavior in Simulation

Andrew Ferree
ferreea1@my.erau.edu

Follow this and additional works at: <https://commons.erau.edu/edt>



Part of the [Automotive Engineering Commons](#), and the [Mechanical Engineering Commons](#)

Scholarly Commons Citation

Ferree, Andrew, "Automated Scenario Generation Using Halton Sequences for the Verification of Autonomous Vehicle Behavior in Simulation" (2021). *PhD Dissertations and Master's Theses*. 591.
<https://commons.erau.edu/edt/591>

This Thesis - Open Access is brought to you for free and open access by Scholarly Commons. It has been accepted for inclusion in PhD Dissertations and Master's Theses by an authorized administrator of Scholarly Commons. For more information, please contact commons@erau.edu.

AUTOMATED SCENARIO GENERATION USING HALTON SEQUENCES FOR
THE VERIFICATION OF AUTONOMOUS VEHICLE BEHAVIOR IN SIMULATION

by

Andrew James Ferree

A Thesis Submitted to the College of Engineering Department of Mechanical
Engineering in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Mechanical Engineering

Embry-Riddle Aeronautical University
Daytona Beach, Florida
April 2021

AUTOMATED SCENARIO GENERATION USING HALTON SEQUENCES FOR
THE VERIFICATION OF AUTONOMOUS VEHICLE BEHAVIOR IN SIMULATION

by

Andrew James Ferree

This thesis was prepared under the direction of the candidate's Thesis Committee Chair, Dr. Patrick Currier, Professor, Daytona Beach Campus, and Thesis Committee Members Dr. M. Ilhan Akbas, Professor, Daytona Beach Campus, and Dr. Eric Coyle, Professor, Daytona Beach Campus, and has been approved by the Thesis Committee. It was submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

Thesis Review Committee:



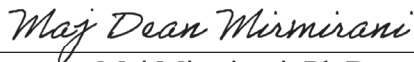
Patrick Currier, Ph.D.
Committee Chair



M. Ilhan Akbas, Ph.D.
Committee Member



Jean-Michel Dhainaut, Ph.D.
Graduate Program Chair,
Mechanical Engineering



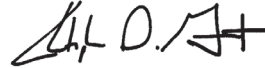
Maj Mirmirani, Ph.D.
Dean, College of Engineering



Eric Coyle, Ph.D.
Committee Member



Eduardo Divo, Ph.D.
Department Chair,
Mechanical Engineering



Christopher Grant, Ph.D.
Associate Vice President of Academics

4/30/2021

Date

Acknowledgements

I would like to thank all the members of my committee: Dr. Patrick Currier, Dr. Ilhan Akbas, and Dr. Eric Coyle for their help and support throughout this entire process. This would have been impossible without them. As the phrase goes: “We stand on the shoulders of giants,” and I am definitely standing on their shoulders.

I could not have done this without my family. There are too many of you to list, but every single one of you helped me get here. Mom and Dad, there are not enough words in the dictionary to describe my thanks. Our phone calls (that might not had been as frequent as we all would have liked), your confidence in me, and most importantly your love helped me through this. The work you put into raising me has gotten me to where I am today. This should be every bit as much of your accomplishment as it is mine.

To my girlfriend, Mary. Thank you for putting up with my crazy work schedule and supporting me throughout the years. Your kindness and compassion have helped keep me sane (mostly) and I could not have done this without you.

To all of my friends, both from North Carolina and the ones I’ve made at Embry-Riddle, you helped keep me going through it all. I will never forget the support and the many needed laughs you have given me.

Lastly, to the cigarette man at the end of my street, whose many friendly waves over the years when leaving for campus always gave me that extra smile for the day. I may have never even known your name, but you will never be forgotten.

Abstract

Researcher: Andrew James Ferree
Title: Automated Scenario Generation Using Halton Sequences for the
Verification of Autonomous Vehicle Behavior in Simulation
Institution: Embry-Riddle Aeronautical University
Degree: Master of Science in Mechanical Engineering
Year: 2021

As autonomous vehicles continue to develop, verifying their safety remains a large hurdle to mass adoption. One component of this is testing, however it has been shown that it is impractical to statistically prove an autonomous vehicle's safety using real-world testing alone. Therefore, simulation tools and other virtual testing methods are being employed to assist with the verification process. Testing in simulation still faces some of the challenges of the real world, such as the difficulty in exhaustively testing the system in all scenarios it will encounter. Manual scenario creation is time consuming and does not guarantee scenario coverage. Pseudo-random scenario generation is a faster option, but still does not ensure coverage of the state space. Therefore, this study proposes the use of Halton sequences to automatically generate scenarios for autonomous vehicle testing in simulation. It compares these scenarios against a set of pseudo-randomly generated scenarios and assesses the performance of each method to cover the simulation state space and provide an accurate depiction of the capabilities of the system-under-test. These tests are carried out in the CARLA simulation environment on an open source, published driving model called "Learning by Cheating" which takes place as the system-under-test. This study concludes that the scenario set generated by the Halton sequence is better at providing an accurate representation of the capabilities of the system-under-test than the pseudo-random scenario generation method.

Table of Contents

Acknowledgements.....	ii
Abstract.....	iii
Table of Contents.....	iv
List of Figures.....	vi
List of Tables.....	x
Nomenclature.....	xi
1 Introduction.....	1
2 Background.....	8
3 Methods.....	15
3.1 Variables and Metrics.....	15
3.2 Simulation Environment.....	16
3.3 SUT Performance Metrics.....	19
3.4 SUT Vehicle Model.....	21
3.5 Halton Sequences.....	22
3.6 Application of Halton Sequences.....	26
4 Results.....	29
4.1 Input Space Coverage.....	29
4.2 SUT Performance.....	35
4.3 Performance Variance.....	45

4.4	Testing Variability.....	46
4.5	Testing Time	48
5	Conclusions	49
6	Future Work.....	50
7	References	52
	Appendix A.....	56
	Appendix B.....	61

List of Figures

Figure 1: The Process of Model-Based Testing [21]	5
Figure 2: Levels of autonomy defined by SAE showing the jump to AD [11], [24].....	6
Figure 3: Overview of a general automated testing approach [31].....	9
Figure 4: Representation of scenes using a layer model [26], [34]	10
Figure 5: Expansion of an RRT [35].....	11
Figure 6: Example of an unavoidable collision [38].....	12
Figure 7: RRT generated test resulting in a collision [38].....	12
Figure 8: A 2-dimensional state space with 50 samples generated by a pseudo-random method (left) and by a Halton sequence (right)	14
Figure 9: Different environmental conditions in CARLA [45]	17
Figure 10: Diversity of assets in CARLA [45]	18
Figure 11: Privileged agent (left) and Sensorimotor agent (right) used in the Learning by Cheating driving model [48].....	22
Figure 12: High correlation between dimensions 19 and 20 of a 20-dimensional state space with 200 samples from the Halton sequence	24
Figure 13: Dimensions 19 and 20 (left) and dimensions 16 and 17 (right) of a 20-dimensional state space with 200 samples from a Halton sequence with a skip value of 3 and a leap value of 78.	26
Figure 14: Histogram of 1000 normalized samples in a 10-dimensional state space from pseudo-random (left) and Halton sequence (right)	30
Figure 15: Histogram of 1000 normalized samples from a Halton sequence with a poorly chosen leap value (1000) in a 10-dimensional state space	31

Figure 16: Cloudiness vs. Fog Density and Fog Density vs. Sun Altitude Angle for 800 scenarios generated by a Halton sequence and 800 scenarios which were pseudo-randomly generated.....	32
Figure 17: Histogram of sun altitude angle from 800 scenarios pseudo-randomly generated (left) and generated from the Halton sequence (right)	33
Figure 18: Mean driving score for 800 scenarios generated pseudo-randomly and by a Halton sequence	36
Figure 19: Percent difference between running mean and final mean driving score.....	37
Figure 20: Histogram of driving scores for the pseudo-random scenario set (left) and the Halton scenario set (right).....	38
Figure 21: Sun altitude angle vs driving score for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	39
Figure 22: Fog density vs driving score for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	39
Figure 23: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 50 scenarios	41
Figure 24: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 100 scenarios	41
Figure 25: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 400 scenarios	42
Figure 26: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	42

Figure 27: Cloudiness vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	43
Figure 28: Wind intensity vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	43
Figure 29: Variance of Halton and pseudo-random driving score.....	45
Figure 30: Mean driving score across 4 repeated runs of the same set of 50 scenarios ...	47
Figure 31: Cloudiness vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	56
Figure 32: Fog density vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	56
Figure 33: Fog distance vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	57
Figure 34: Fog Falloff vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	57
Figure 35: Precipitation deposits vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios.....	58
Figure 36: Precipitation vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios	58
Figure 37: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios.....	59
Figure 38: Sun azimuth angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios.....	59

Figure 39: Wetness vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios 60

Figure 40: Wind Intensity vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios 60

List of Tables

Table 1.1	<i>Examples of miles and years needed to demonstrate autonomous vehicle reliability.....</i>	2
Table 3.1	<i>Computer hardware and software versions used in study.....</i>	16
Table 3.2	<i>Failures for which the SUT will be assessed</i>	20
Table 3.3	<i>Infraction penalty coefficients</i>	20
Table 3.4	<i>Simulation parameter state space to be varied by Halton sequence</i>	27
Table 4.1	<i>Mean of 800 Halton and pseudo-randomly generated scenarios and their percent difference from the ideal mean</i>	34
Table 4.2	<i>A subset of 10 scenarios of the results of 50 scenarios run 4 times</i>	47
Table 7.1	<i>Results of the same set of 50 scenarios run 4 separate times</i>	61

Nomenclature

ACC – Adaptive Cruise Control

AD – Automated Driving

ADAS – Advanced Driver Assistance System

AV – Autonomous Vehicle

LKAS – Lane Keeping Assistance System

NTSB - National Transportation Safety Board

ODD – Operational Design Domain

PRM – Probabilistic Roadmap

RRT – Rapidly-Exploring Random Trees

SUT – System Under Test

V&V – Verification and Validation

1 Introduction

In 2018 alone, 36,560 people died and 2,710,000 were injured in car accidents in the United States [1], [2]. The economic impact of car accidents in 2010 cost the U.S. \$242 billion, which represented roughly 1.6 percent of U.S. Gross Domestic Product [3]. Factoring in the total societal harm, when including quality-of-life valuations, that cost increases to \$836 billion [3]. It is estimated that about 94% of accidents are caused by human error, while the other 6% is attributed to the vehicle, environment, or unknown reasons [4]. For years, autonomous vehicles (AVs) have promised to drastically reduce the number of these accidents by removing the segment that is caused by human error, as AVs offer clear advantages, such as never getting distracted, fatigued, or intoxicated, however, no system will ever be perfect [5].

The developers of AVs must ensure that they do not just trade accidents caused by human error for accidents caused by computer error. It has been found that public trust in AVs is one of the driving factors in their acceptance [6]. Already, 56% of Americans would not want to ride in AVs if given the opportunity, citing a general lack of trust that the AV would perform in a safe manner [7]. If AVs are to help prevent accidents due to human error, then the humans must be willing to trust the technology and cede control of driving to the computer. More accidents involving AVs will not help increase this consumer confidence in the technology, therefore demonstrating the capability and safety of an AV should be a primary objective before mass adoption.

One way to demonstrate that AVs are safe enough is through real-world test driving [8]–[11]. However, as shown by Kalra and Paddock, to statistically prove AVs

are safe enough would require millions if not billions of miles driven, which would take tens to hundreds of years to drive with a fleet of 100 AVs (Table 1.1) [12].

Table 1.1

Examples of miles and years needed to demonstrate autonomous vehicle reliability

How many miles (years) would have to be driven... X ...Y.				
		Y		
		(A) 1.09 fatalities per 100 million miles?	(B) 77 reported injuries per 100 million miles?	(C) 190 reported crashes per 100 million miles?
	(1) without failure to demonstrate with 95% confidence that their failure rate is at most...	275 million (12.5 years)	3.9 million (2 months)	1.6 million (1 month)
X	(2) to demonstrate with 95% confidence their failure rate to within 20% of the true rate of...	8.8 billion (400 years)	125 million (5.7 years)	51 million (2.3 years)
	(3) to demonstrate with 95% confidence and 80% power that their failure rate is 20% better than the human driver failure rate of...	11 billion (500 years)	161 million (7.3 years)	65 million (3 years)

From “Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?” by N. Kalra & S. M. Paddock, *Transportation Research Part A: Policy and Practice*, 94, p. 182–193.

The time it would take in years to drive the necessary number of miles is calculated using the following assumptions: a fleet of 100 autonomous vehicles (larger than any known existing fleet) driving 24 hours a day, 365 days a year, at an average speed of 25 miles per hour [12]. This shows that proving the safety of an AV solely through real-world on-the-road testing is not feasible. Other methods for proving AV safety, such as simulations, should be used[13].

By improving the detection of failures of AV driving functions in a simulated environment, systems engineering teams can identify key software modules in need of improvement and additional development before testing in the real world. Not only does this save valuable testing time on physical vehicles, but it will also help to prevent accidents. For example, in the March 2018 accident involving an Uber Technologies, Inc. developmental automated driving system and a pedestrian in Tempe, Arizona, the National Transportation Safety Board (NTSB) noted in its findings that the Uber Advanced Technologies Group failed to manage the risk of the limitations of its automated driving system [14]. If this scenario had been tested in simulation prior to the vehicle being tested on the road, then Uber Technologies, Inc. might have been able to better identify the limitations of the system and ultimately saved a life.

Using simulation in autonomous vehicle verification and validation (V&V) has been suggested as playing an important role in verifying the safety of an autonomous system [10], [15], [16]. The usage of simulation is even discussed in one of the first proposed safety standards for building the safety case for the development of AVs [17]. There are, however, several limitations of simulation. The variance of the simulated dynamic model and the real-world dynamics the system-under-test (SUT) experiences are just some examples. Other limitations include imprecise sensor models and the lack of injection of real-world sensor noise [10]. When testing the AV, these limitations and variances from the real-world can result in behaviors that are shown to be safe in simulation but result in unsafe behavior in the real-world. In the proposed standard, UL 4600 requires the identification and documentation of these limitations to build an appropriate safety case for the system [17].

AVs have extensive and complex requirements due to the complexity of the environments they operate in [16]. Consequently, the resulting software developed to meet these requirements is very complex, made up of a multitude of modules ranging from perception to path planning and decision making [10]. The more complex the software, the more difficult it becomes to verify that it meets its requirements. For example, it is an immense challenge to verify a planner since they are designed to discover solutions to problems with very large state spaces [18].

Another large challenge for testing AVs is the popularity of non-deterministic algorithms such as machine learning or algorithms utilizing random number generators [10], [19], [20]. When non-deterministic algorithms are executed, different outputs might be observed on different executions when given the same exact inputs. This means passing a unit test once does not necessarily mean the system will always pass that exact same unit test [10].

According to Utting et al., “The goal of testing is failure detection” which means finding differences between the implemented and intended behavior of the SUT as defined by its requirements [21]. An example of one type of testing process can be seen in Figure 1. In this model-based testing process, requirements drive both the model and the test selection criteria. From the selection criteria, test case specifications are created. Then, using these test case specifications and the model, full test cases can be developed and subsequently executed using a test script. Often, some adaptor script is needed to feed the SUT the information from the test cases. Finally, the verdicts are returned after the test cases are complete.

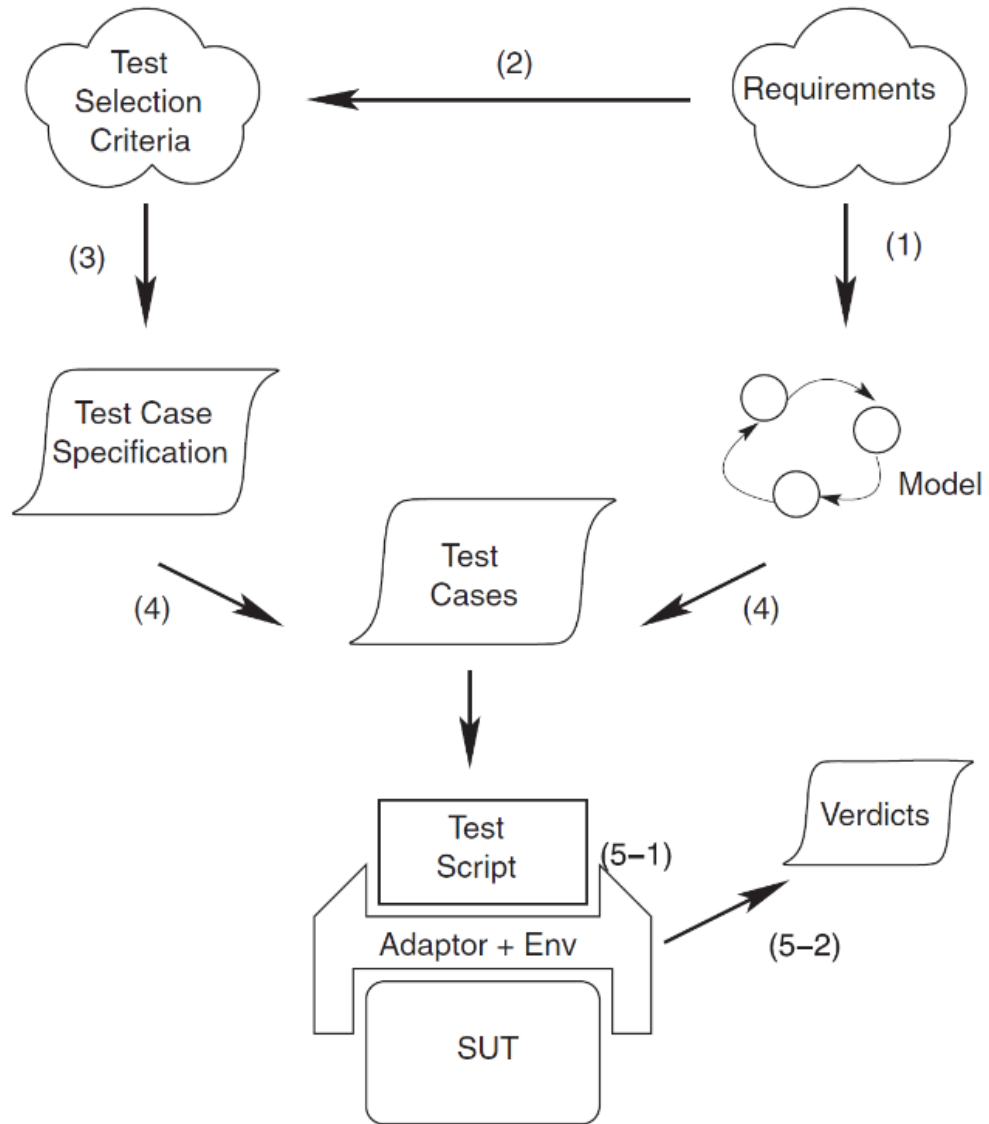


Figure 1: The Process of Model-Based Testing [21]

Developing these tests to determine situations where an AV fails is critical to their development and future safety [8]. Work has been done to formally represent different driving scenarios the AV might encounter and then verify that the vehicle has responded in an appropriate and safe manner [9], [22], [23]. In previous Advanced Driver Assistance Systems (ADAS), such as Lane Keeping Assistance (LKAS) and Adaptive Cruise Control (ACC), the number of scenarios that the system would be expected to

perform in was limited and keeps the driver in the loop. However, when moving to the more unbounded operating conditions of Automated Driving (AD) such as in the SAE autonomy levels three through five, the number of scenarios the system can encounter becomes immeasurable [11]. These levels of autonomy can be found below in Figure 2.

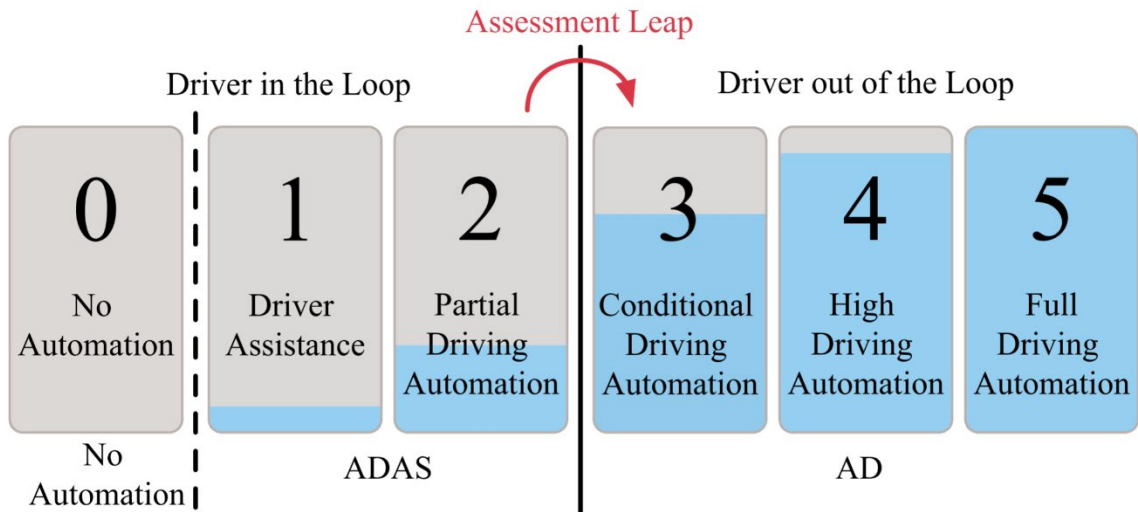


Figure 2: Levels of autonomy defined by SAE showing the jump to AD [11], [24]

As the autonomy level increases, so does the scope and complexity of the operational design domain (ODD) and defining tests that accurately describe all of the scenarios in which the system will operate and exhaustively test all AD functions becomes significantly more difficult [8], [10], [11]. Creation of a catalog of scenarios by experts has been used for testing critical situations in the past, such as is required in the ISO 26262 standard hazard analysis and risk assessment where hazardous events must be determined using adequate techniques [25]. However, this manual method is infeasible to construct an exhaustive list of scenarios for testing high level AD functions due to the complexity of the ODD and the immeasurable number of potential scenarios which the

AV might encounter [11], [26]. Another method, potentially an automated approach, will need to be implemented.

Since simulation will play an important role in the V&V of AV behavior, there needs to be methods available to design scenarios that exhaustively test AD functions [9], [12]. While complete coverage of scenarios in the real-world ODD is infeasible, an algorithm that can generate scenarios by more uniformly covering the simulation state space is a more tractable goal [8], [10], [11]. Therefore, it is the purpose of this study to evaluate using Halton sequences to improve coverage of the simulation state space while limiting the number of redundant scenarios which can then be used for testing AD performance. It will explore the following questions:

- How could a larger number of challenging edge-cases be discovered by applying an automated scenario variation technique to increase the risk dimension of the scenarios?
- How can Halton sequences be used to create variations of existing scenarios that improve the exploration of the simulation state space while decreasing the time and number of scenarios?
- How does the mean performance and performance variance of the system-under-test (SUT) in scenarios generated by a Halton sequence compared to those which are randomly generated?

2 Background

This study has discussed scenarios and test cases and sometimes used the two terms interchangeably. The reason for this requires further discussion. Formally defined, a scenario “is a description that contains (1) actors, (2) background information on the actors and assumptions about their environment, (3) goals or objectives, and (4) sequences of actions and events” [27]. A scenario can also be described as a sequence of scenes, where scenes are individual snapshots in time which contains all of the instantaneous parameters of the scenario at that point [11], [28]. Scenes are to scenarios as frames are to videos. The parameters in each scene and the changes from one scene to another help create the entire scenario. Some of these changes from one scene to another might be, for example, the velocity of another actor or the luminosity of the environment. While there are parameters that might change from scene to scene, there are also parameters that can be defined at the beginning of the scenario, such as precipitation, global luminosity, road friction, etc. [23], [29]. All of these parameters help bound and define distinct scenarios to test the AV.

One method to describe these scenarios is by using ontologies [26], [30]. Ontologies are a set of concepts in a domain that formally portrays different entities, their properties, and their relationships between one another. Ontologies have been proposed for use in different applications ranging from test generation to AV decision making [30]–[33]. The variables then defined in these ontologies can be combined to generate test cases. These test cases then can be used to evaluate what situations cause failures of the autonomous driving functions [30]. Figure 3 shows a general automated testing approach that utilizes ontologies.

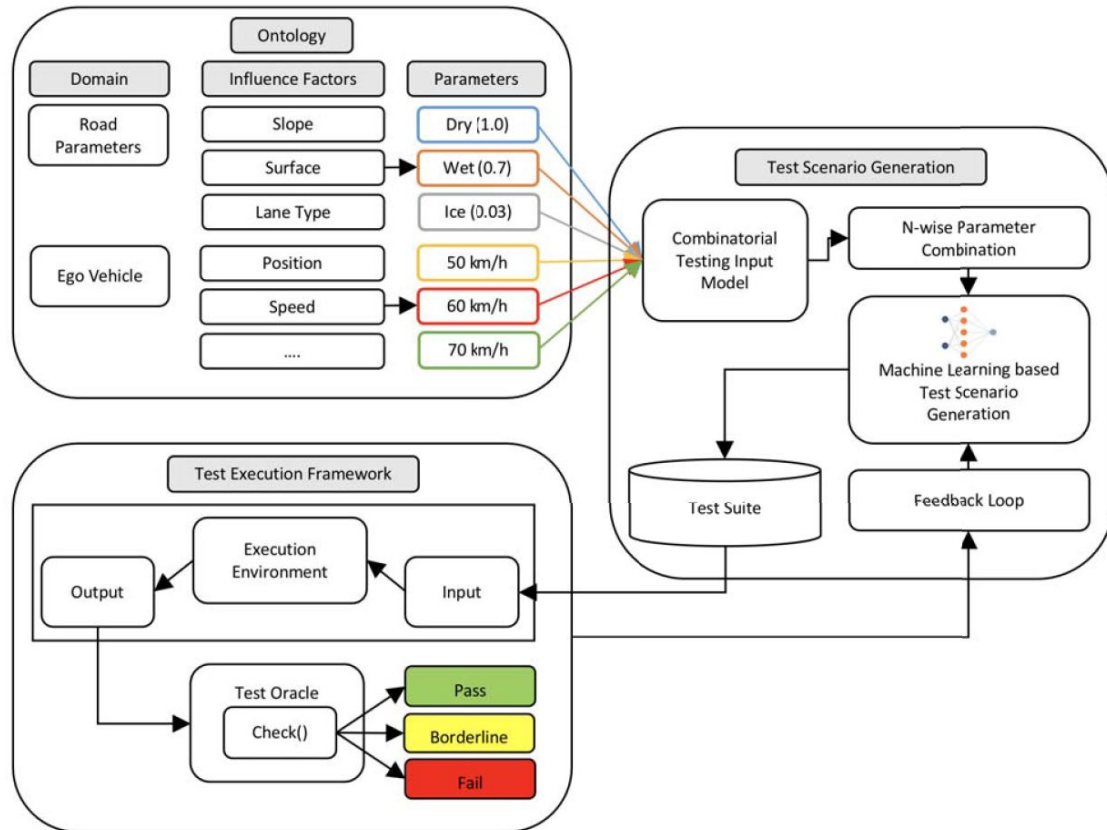


Figure 3: Overview of a general automated testing approach [31]

It has been proposed that using ontologies to represent scenes offer a natural language approach to generating scenarios in an efficient manner [26]. In doing this, Bagschik et al. show that the knowledge base could be more easily represented and leveraged to produce a diverse set of scenarios. The information is organized into a layered model representing the knowledge base, as shown in Figure 4.

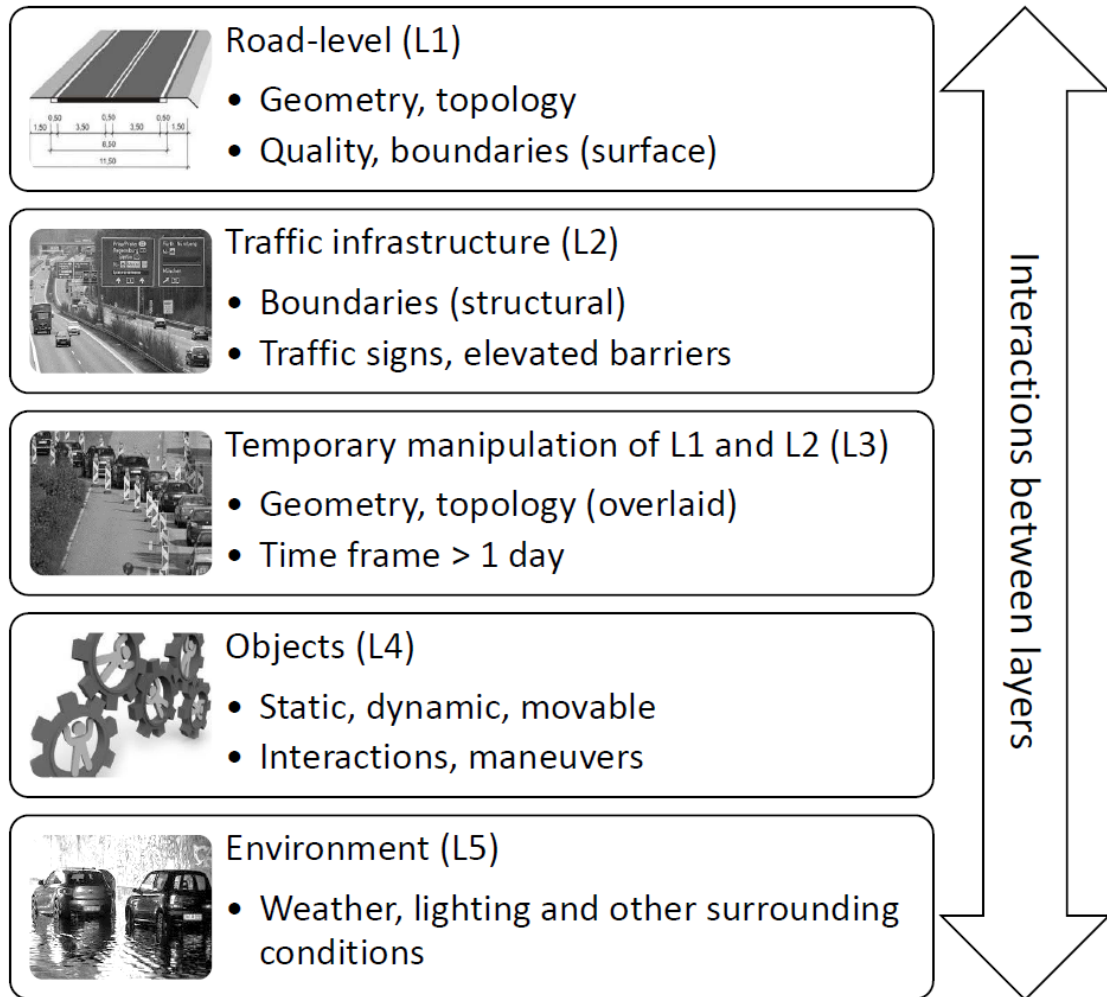


Figure 4: Representation of scenes using a layer model [26], [34]

This method produced a scene catalog that did not need to be reviewed by analysts since as long as the elements in the ontology are accurately modeled, then the elements would be correctly combined with other layers. While this method provides an excellent way for combinatorial testing, it does not guarantee coverage of, for example, all the different environmental conditions in L5 since it is still just representing expert knowledge. Therefore, a more automated approach needs to be applied to guarantee coverage of this state space.

One way to automatically cover the state space to generate scenarios is by using Rapidly-Exploring Random Trees (RRTs). RRTs are typically used as path planning algorithms [35], [36]. They have been useful in this domain for multiple reasons such as their ability to inexpensively compute paths from a start to a goal and their bias towards exploration of the state space [35]. It does this by sampling the state space and extending toward the sampled point. An example of 2D RRT growth is shown in Figure 5. Note how the RRT grows towards the unexplored areas in the space. The more iterations the algorithm runs, the more explored the state space will become.

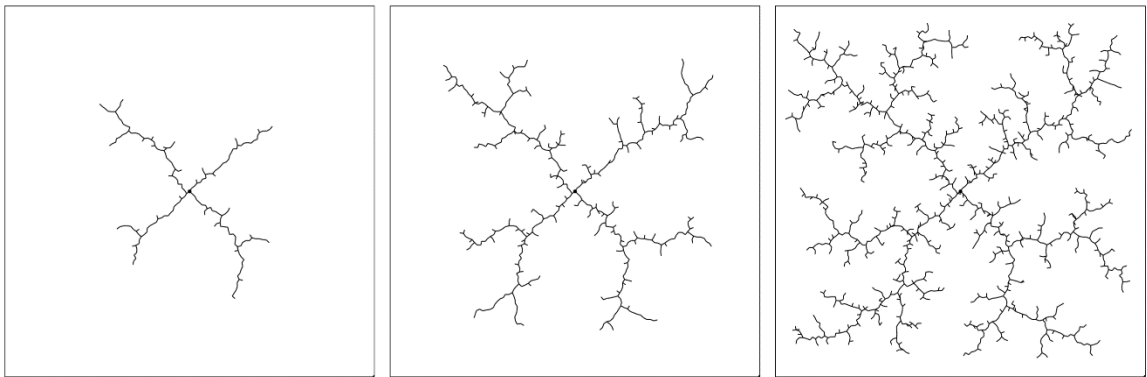


Figure 5: Expansion of an RRT [35]

This bias towards exploration of the state space could be applied to generating scenarios in a simulation environment since another valuable feature of RRTs are their ability to be applied to higher-dimensional state spaces [37]. Already, there is some evidence that RRT's could be used for test generation for AVs [38].

One approach for applying RRT's to scenario generation is to determine the boundaries in which situations an AV can and can't avoid collisions [38]. For example, Figure 6 shows the red vehicle entering the lane of the yellow vehicle at a distance too

close for the yellow vehicle to stop. By applying an RRT, multiple trajectories can be generated to find these types of situations.

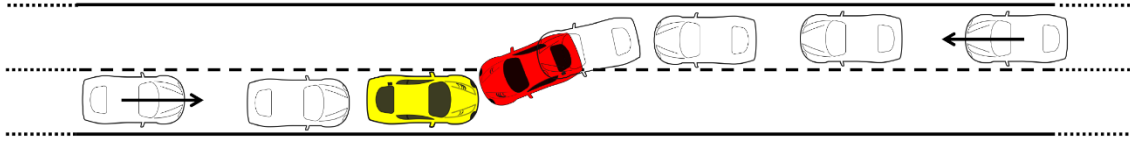


Figure 6: Example of an unavoidable collision [38]

This is done by sampling a target path segment, where the target path segment is just a set of waypoints that are defined by an x and y coordinate, target heading and speed. Once this is sampled, a segment is extended some distance from the waypoint. The results of this method are random paths that can produce numerous situations that might produce collisions between the AV and other actors on the road and it minimizes the need for manually designing scenarios for the SUT [38]. An example of one case where a collision was found using the automated RRT method is shown in Figure 7. Identifying what situations AVs cannot avoid collisions can inform system designers on which areas to focus improvement efforts on, as well as advanced knowledge of situations the AV might fail to react properly during real-world testing.

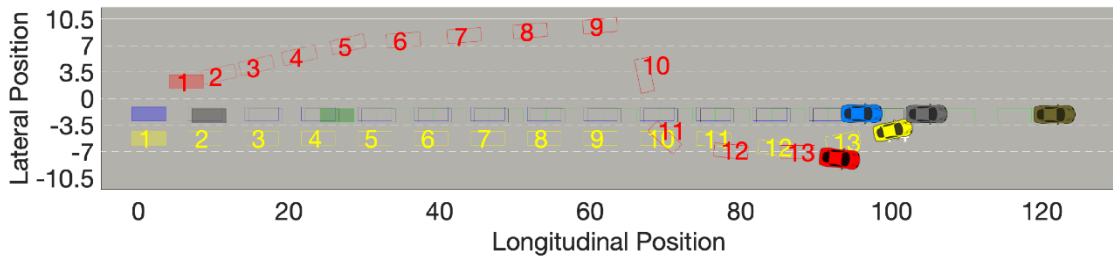


Figure 7: RRT generated test resulting in a collision [38]

RRT's might provide a good way to cover the state space due to its bias towards exploration, however this method might prove to be inefficient due to the need to start with some root scenario and then expand out away from that scenario. The result would be starting with scenarios closer to the root scenario and working away. While the distance the algorithm can extend away from that scenario can be tuned, the system still must start with some root scenario and then move through the state space from there. Though the RRT method might be better suited for making variations to one scenario, for a more generic testing approach it might be more desired to have a uniform distribution of samples in the state space to evenly distribute scenarios.

To distribute samples of simulation parameters more uniformly in the state space, quasi-random, low-discrepancy sequences could be a promising avenue. Sampling techniques that use quasi-random sequences such as Hammersley, Sobol, Faure, and Halton sequences have been used in multiple instances in the field of path planners for robots [39]–[41]. Distributing samples more uniformly, also referred to as having a lower discrepancy, is one of the advantages of quasi-random over pseudo-random sampling [42]. For example, Figure 8 shows 50 samples in a two-dimensional space that were generated by a pseudo-random method and by a quasi-random method, specifically a Halton sequence. Note that the points generated by the pseudo-random method suffer from clumping of points and larger spaces without samples, whereas the samples generated by the Halton sequence are more uniformly distributed in the space. For a more in depth explanation of Halton sequences, refer to Section 3.5.

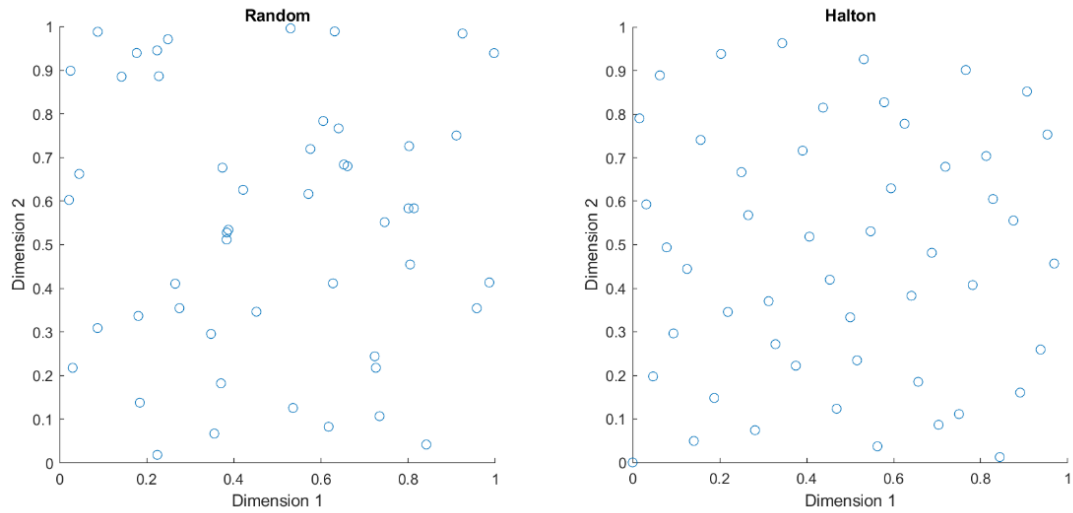


Figure 8: A 2-dimensional state space with 50 samples generated by a pseudo-random method (left) and by a Halton sequence (right)

Since the goal of testing is failure detection, the need to cover the large number of scenarios an AV may encounter is a key element of finding areas where an AD system fails [8], [10], [11], [21]. Using a quasi-random sampling method, such as a Halton sequence, to generate diverse scenario catalogs that have a more uniform coverage of the state space over pseudo-random sampling could be a useful solution. Halton sequences may provide better coverage of the simulation state space and ensure that there are fewer redundant scenarios in the resulting scenario catalog [43].

3 Methods

This study will use an open-source, photorealistic simulation environment to evaluate using Halton sequences over pseudo-random sampling to generate scenarios for AV testing. This methods section will outline the controlled and extraneous variables in Section 3.1 and summarize the metrics for evaluating the system model's performance in Section 3.3. The simulation platform that will be used is discussed in Section 3.2 and the vehicle model that will be evaluated in tests is shown in Section 3.4. Halton sequences and their implementation for generating scenarios will also be covered in Sections 3.5 and 3.6, respectively.

3.1 Variables and Metrics

When evaluating the performance of techniques in creating scenarios, the following metrics will be used. Outlined by Chance et al. (2020), these are attributes of 'good' test cases.

1. Effectiveness – How well do the generated scenarios find failures?
2. Efficiency – Are the number of test cases reduced?
3. Economy – How long does it take to generate the scenarios?
4. Robustness – How well does it handle changes in the state space?

Variables that will be fixed during this study will be the computation hardware, simulation environment, and the vehicle model being used as the system-under-test (SUT). Extraneous variables include outputs from non-deterministic algorithms in the SUT and will be discussed at the end of this section.

Hardware is one of the controls in this study, as one of the metrics that will be evaluated, as discussed above, is “economy,” or how long it takes to generate the scenarios. The hardware used for this study is shown in Table 3.1.

Table 3.1

Computer hardware and software versions used in study

CPU	Intel Core i9-9900K CPU @ 3.70GHz
RAM	32GB DDR4
GPU	Nvidia GeForce RTX 2080
OS	Ubuntu 20.04 LTS
CARLA Version	0.9.10.1

Extraneous variables that could have an impact on the results of the tests include the outputs from non-deterministic algorithms. As discussed before, non-deterministic algorithms, which are popular in AVs, can result in different outputs when given the same set of input parameters [10]. Because of this, a scenario that was generated and used for vehicle testing might reveal a system failure on the first run but might not reveal a system failure on a subsequent run. Multiple simulation test runs with the same set of generated scenarios will be conducted to observe the impact some of these non-deterministic systems have on the results. The variance will be calculated for the scores across all runs of the same scenario set to determine the variability of repeated executions of the same test.

3.2 Simulation Environment

The simulation environment used in this study is CARLA (Car Learning to Act), an open-source 3D AV simulator with a high degree of controllability of the simulation

environment [45]. Environmental conditions (such as precipitation, fog, and ambient occlusion), sensor characteristics, roadway placement, pedestrian locations, and vehicle dynamics are just some of the many parameters that can be controlled. Such high controllability of the simulation state space can enable scenario generation algorithms to produce unique scenarios that can be used to evaluate the SUT. While a vast number of parameters can be controlled, this study will focus primarily on the parameters related to environmental conditions.



Figure 9: Different environmental conditions in CARLA [45]

Another benefit of CARLA is the diverse number of cars, pedestrians, and maps available for use. There are 50 different animated pedestrian models, 16 unique vehicle models (with options of varying textures), 40 different buildings, as well as multiple pre-made towns with varying driving environments (city, highway, etc.) [45]. Samples of some of these assets can be seen in Figure 10.



Figure 10: Diversity of assets in CARLA [45]

A diverse range of assets will allow for multiple combinations of assets to enable diverse scenarios to be generated. CARLA also has methods to create vehicles, sensors, and maps for a more customized environment. Road networks can be imported from OpenDRIVE, which is an open file format used to easily exchange road network logic between different simulators [46]. CARLA also supports scenario specification using the OpenSCENARIO standard, which defines an XML format for describing the complex, dynamic environments and maneuvers in simulation environments [29].

CARLA also has an Autonomous Driving Leaderboard, which evaluates various autonomous driving agents that are submitted to the leaderboard using an assortment of predefined traffic scenarios and weather conditions. The driving model evaluated in this study was found on the Autonomous Driving Leaderboard and was chosen because it is well documented and open-sourced. Additionally, the metrics this study uses to evaluate the AD system's performance are the same ones that would be evaluated by the Autonomous Driving Leaderboard. These performance metrics and the driving model used as the SUT will be described in the following sections.

3.3 SUT Performance Metrics

Some expected failures that the SUT will be tested for include collisions, blocked actor, lane departure, and moving violations. Collisions that will be tested for include collisions with other vehicles, pedestrians, and infrastructure (signs, light poles, buildings, etc.). A blocked actor is the result of vehicle immobility for a longer-than-expected period. Examples of a blocked actor failure would be the vehicle remaining stationary at a green light or never proceeding after a stop sign. Lane departure/incursion includes leaving the roadway, entering lanes of oncoming traffic, and departing the lane the vehicle is supposed to be in (i.e., drifting into another lane). Finally, moving violations include some of the standard rules-of-the-road, such as running a red light or a stop sign. These failures are tabulated below and are the only ones that the SUT will be evaluated for.

Table 3.2

Failures for which the SUT will be assessed

Category	Failure
Collisions	Collided with another vehicle
	Collided with a pedestrian
	Collided with static elements
Blocked Actor	Actor immobile for more than 180 seconds
Lane Departure	Actor is outside of route lane lines
Moving Violations	Failure to stop at a stop sign
	Failure to stop at a stop light

Each of these failures can be characterized by parameters, called infraction penalties, that can then be used to calculate an overall score of the test. These penalties and their corresponding values are shown in Table 3.3. The coefficients assigned to each infraction penalty are defined by the CARLA Autonomous Driving Leaderboard and do not necessarily reflect a defined real-world severity ranking. Lane departure, which has no infraction penalty, is accommodated for by deducting the percentage of the route for which the SUT is outside the route lane lines from the total route completion percentage.

Table 3.3

Infraction penalty coefficients

Collisions with pedestrians	0.50
Collisions with vehicles	0.60
Collisions with static elements	0.65
Failure to stop at a red light	0.70
Failure to stop at a stop sign	0.80

The overall score is calculated by calculating the percentage of the route completed, as well as then including penalties for the infractions the SUT committed. Using these penalty coefficients, p , and the percentage of the route completed, R , the following equations can be used to calculate an overall driving score. Equation 1 aggregates the infraction penalties and Equation 2 combines both to get an overall average driving score for the i -th route across N routes [47]. A perfect score (meaning 100% route completion, 0% of the route outside of lane lines, and 0 infractions) would result in a value of 100, whereas the worst score would be 0.

$$P_i = \prod_j^{ped, \dots, stop} (p_i^j)^{\# \text{ of infractions}_j} \quad (1)$$

$$Driving \ Score = \frac{1}{N} \sum_i^N R_i P_i \quad (2)$$

3.4 SUT Vehicle Model

This study uses an existing, open-source driving model and evaluates that driving model using the afore mentioned metrics in a variety of conditions that will be automatically generated. The model used is the Learning by Cheating driving model, which uses a trained agent with access to privileged information to train a vision-based agent which uses only a forward facing RGB camera [48]. The privileged agent uses ground-truth information such as the locations of other actors and the layout of the environment and is then trained from a set of expert trajectories. The second, a sensorimotor agent which does not have access to this ground-truth information, then learns to imitate the privileged agent. Figure 11 shows an overview of both of these agents.

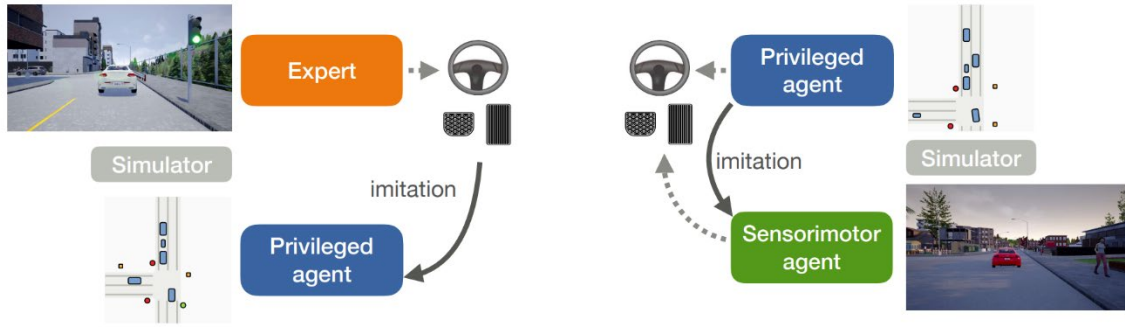


Figure 11: Privileged agent (left) and Sensorimotor agent (right) used in the Learning by Cheating driving model [48]

This agent is trained using a dataset with various situations and weather conditions, for example, changing the weather every few seconds to add variety to the images collected for agent to be trained with. This should result in at least a moderate resilience to changing environmental conditions such as luminosity, fog, and rain.

The Learning by Cheating driving model will be used as the SUT and evaluated based on the performance metrics that have been previously discussed. This study uses a pre-trained model from the Learning by Cheating’s Wandb which was trained on April 18th, 2020 and is named “command_coefficient=0.01_sample_by=even_stage2”. The scenarios that will be used to test this driving model will be generated using Halton sequences, which will be discussed next.

3.5 Halton Sequences

Halton sequences are quasi-random, multi-dimensional sequences which generate samples in a space and have been used in applications such as Monte Carlo simulations and probabilistic roadmaps for path planning [41], [49], [50]. They are also low-discrepancy, meaning that they attempt to more uniformly distribute samples inside of a state space. Halton sequences are generated by using coprime numbers as its bases

$b_1, b_2 \dots b_s$ for each dimension s . The following equations describe the Halton sequence [51], [52]. Any positive integer n can be written in base b with the integer string d_i ($d_0, d_1, \dots d_i$) such that:

$$n = d_0 + d_1 b^1 + d_2 b^2 + \dots + d_j b^j, \quad 0 \leq d_i < b \quad (3)$$

Then, using the radical inverse of n for base b :

$$\varphi_b(n) = d_0 b^{-1} + d_1 b^{-2} + \dots + d_j b^{-j-1} \quad (4)$$

Finally, the Halton sequence for each prime base b in each dimension s is the set:

$$x_n = \left(\varphi_{b_1}(n), \dots, \varphi_{b_s}(n) \right), \quad n = 0, 1, 2, \dots \quad (5)$$

Therefore, to demonstrate, the first 4 points of a 2-dimensional Halton sequence using the primes 2 and 3 as b_1 and b_2 , respectively, would result in $\left[(0,0), \left(\frac{1}{2}, \frac{1}{3}\right), \left(\frac{1}{4}, \frac{2}{3}\right), \left(\frac{3}{4}, \frac{1}{9}\right) \right]$.

One of the problems associated with Halton sequences is the high correlation between dimensions of higher dimensional state spaces [50], [52]. For example, Figure 12 shows dimensions 19 and 20 of a 20-dimensional state space plotted against each other.

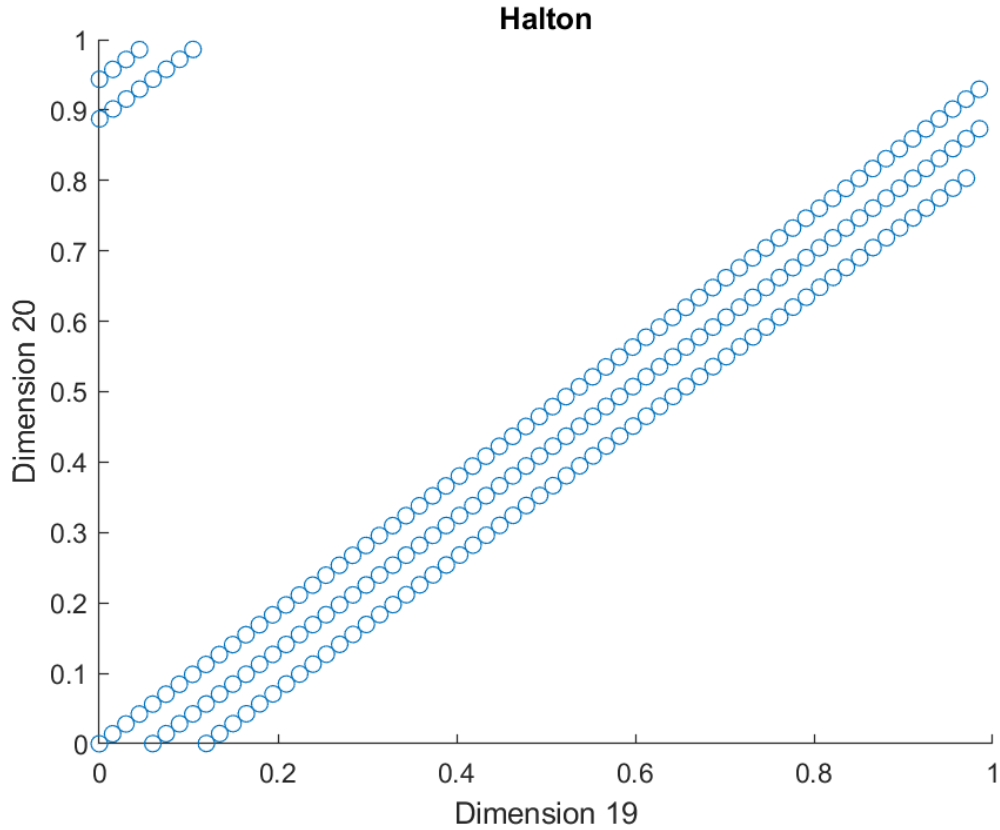


Figure 12: High correlation between dimensions 19 and 20 of a 20-dimensional state space with 200 samples from the Halton sequence

This correlation between higher dimensions in the state space would be sub-optimal for applying Halton sequences to uniformly sample the simulation state space since there could be a multitude of parameters that are desired to be varied. Fortunately, there are a few remedies to this problem.

The three ways this study explored to correct for the correlation between higher dimensions is the application of skipping, leaping, and scrambling to the Halton sequence. The first method, skipping, simply omits a specified number of initial points from the sequence. For example, when using primes 23 and 29 as the bases, the first five point-pairs of the sequence are $\left[(0,0), \left(\frac{1}{23}, \frac{1}{29}\right), \left(\frac{2}{23}, \frac{2}{29}\right), \left(\frac{3}{23}, \frac{3}{29}\right), \left(\frac{4}{23}, \frac{4}{29}\right) \right]$. If a skip of 3

was applied, then the resulting sequence would start with $\left(\frac{3}{23}, \frac{3}{29}\right)$ and continue onward, like so $\left[\left(\frac{3}{23}, \frac{3}{29}\right), \left(\frac{4}{23}, \frac{4}{29}\right), \left(\frac{5}{23}, \frac{5}{29}\right), \dots\right]$. However, the perfectly linear correlation between the two dimensions still remains, therefore leaping is also applied.

Leaping is a method to attempt to eliminate the cycles and linear correlation between higher dimensions, improving overall point set quality to be more uniform [50]. It accomplishes this by specifying a number of points in the sequence to omit after each sample, hence “leaping” through the sequence. To demonstrate, consider again the first five point-pairs of a Halton sequence with base primes as 23 and 29, resulting in $\left[(0,0), \left(\frac{1}{23}, \frac{1}{29}\right), \left(\frac{2}{23}, \frac{2}{29}\right), \left(\frac{3}{23}, \frac{3}{29}\right), \left(\frac{4}{23}, \frac{4}{29}\right)\right]$. If a leap of 3 was applied, the resulting sequence would become $\left[(0,0), \left(\frac{4}{23}, \frac{4}{29}\right), \left(\frac{8}{23}, \frac{8}{29}\right), \dots\right]$. Selecting a correct leap value for this is important, as the wrong leap value can lead to new, undesirable cycles [50]. One way to selecting a decent leap value is by selecting a prime that is greater than the largest prime in the sequence and subtracting 1. For example, in a 20-dimensional state space the largest prime is 71, with the next two primes that are unused being 73 and 79. If following the suggestion, the prime number 79 could be used and then have 1 subtracted from it to get 78, which would be the value to use for the leap. In Figure 13 is the same number of samples and dimensions as in Figure 12, however with a skip of 3 and a leap of 78. This still does not solve the problem for all dimensions, though. When plotting dimensions 16 and 17 against each other, the secondary cycling induced by the chosen leap value is observed.

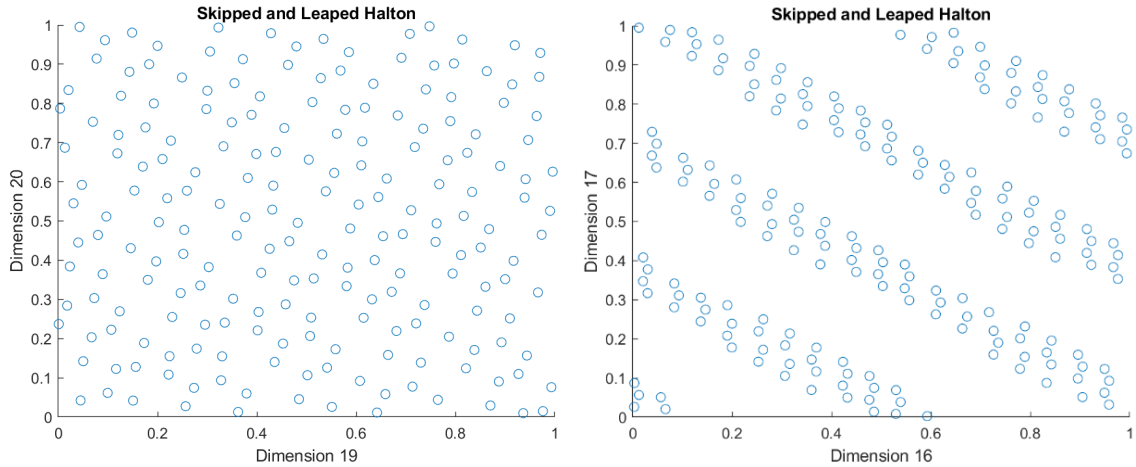


Figure 13: Dimensions 19 and 20 (left) and dimensions 16 and 17 (right) of a 20-dimensional state space with 200 samples from a Halton sequence with a skip value of 3 and a leap value of 78.

To fix the issues of these cycles occurring in higher dimensions, a scramble is applied to the sequence. The specific scrambling algorithm used is the Reverse-Radix Algorithm, otherwise known as HaltonRR2. According to Kocis and Whiten (1997), who developed this scrambling method, it works by changing the values of d mentioned above and “reversing the binary digits of integers, expressed using a fixed number of base-2 digits, and removing any values that are too large.” This results a lower discrepancy, or more uniform distribution, of samples in higher dimensional state spaces [50]. The built-in functions in MATLAB were used to both generate the Halton sequence and to apply the HaltonRR2 scramble to the sequence. Once the sequence was generated in the proper number of dimensions, each dimension could be scaled to the proper range for the simulation parameters.

3.6 Application of Halton Sequences

As mentioned previously in Section 3.2, CARLA can simulate many different weather conditions. This paper proposes applying Halton sequences to automatically

cover these different conditions more uniformly so that the capabilities of the SUT may be more accurately described. The variables that will be varied by using a Halton sequence are outlined and described in Table 3.4.

Table 3.4

Simulation parameter state space to be varied by Halton sequence

	Minimum	Maximum	Description
Cloudiness	0	100	0 is completely clear, 100 is overcast
Fog Density	0	100	0 is no fog, 100 is extremely thick fog
Fog Distance	0	5	Distance the fog starts, in meters
Fog Falloff	0	5	Specific mass of the fog; the larger the value, the closer the fog will be to the ground. At 1, it is approximately as dense as the air.
Precipitation	0	100	0 is no rain, 100 is heavy rain
Precipitation Deposits	0	100	Amount of water on the road. 0, there is no water, 100 completely covered
Wetness	0	100	Humidity percentage of the road
Wind Intensity	0	100	0 is no wind, 100 is strong wind
Sun Azimuth Angle	0	180	Arbitrary north is 0, with south being 180
Sun Altitude Angle	-90	90	90 corresponds with noon, whereas -90 corresponds to midnight

Using these parameters results in a 10-dimensional state space, of which sample sets will be generated using both the Halton sequence and a pseudo-random generator. While parameters take on varying ranges of values, the values will initially be generated on the unit hypercube and will then be scaled to their appropriate range. For example, a sample point of 0.5 in the Fog Distance dimension would be scaled to 2.5. Once these are scaled, the resulting parameters are written to an XML file that can be read by the simulator.

In addition to comparing the Halton generation method to the pseudo-random method, this study will also examine how the number of scenarios generated impact each methods ability to accurately describe the capabilities of the SUT. Analyzing this should then, if the hypothesis of the study is correct, show that Halton sequences are able to accurately describe the capabilities of the SUT in a fewer number of samples than the pseudo-random method since the Halton sequences are more uniformly covering the state space.

4 Results

The results of this study will cover a few different perspectives. The first will be the coverage of the input space by both pseudo-random and Halton sequence generated point sets. Next, the observed performance of the SUT for both the pseudo-random and Halton generated point sets will be shown. Lastly, the variance of the resulting scores for varying number of point sets and the variability of results when running the same scenario multiple times will be examined.

4.1 Input Space Coverage

While it has been proven that Halton sequences have a lower discrepancy than pseudo-random sequences in lower dimensions, and that this lower discrepancy can be extended to higher dimensions by applying a scramble to the sequence such as the HaltonRR2 method, a sanity check for uniformity should still be performed [50], [52]. After generating scenario files for the simulator to run, these weather parameters are read back into MATLAB and then histograms are created. Since the parameters had been scaled, as described in Section 3.6, the data is then normalized to the range of 0 to 1 and all of the points generated for all of the dimensions are fit onto one histogram. If the points are indeed uniformly distributed, then there should be a uniform distribution on the histogram. As seen in Figure 14, the Halton sequence, as expected, has a much more uniform distribution of samples than the pseudo-random generator for 100 scenarios for a 10-dimensional state space (1000 total samples). Quantitatively, the variance of the quantity of samples in each bin on the histogram for the pseudo-random scenario set was 174.7, whereas the variance for the Halton scenario set was 4.22.

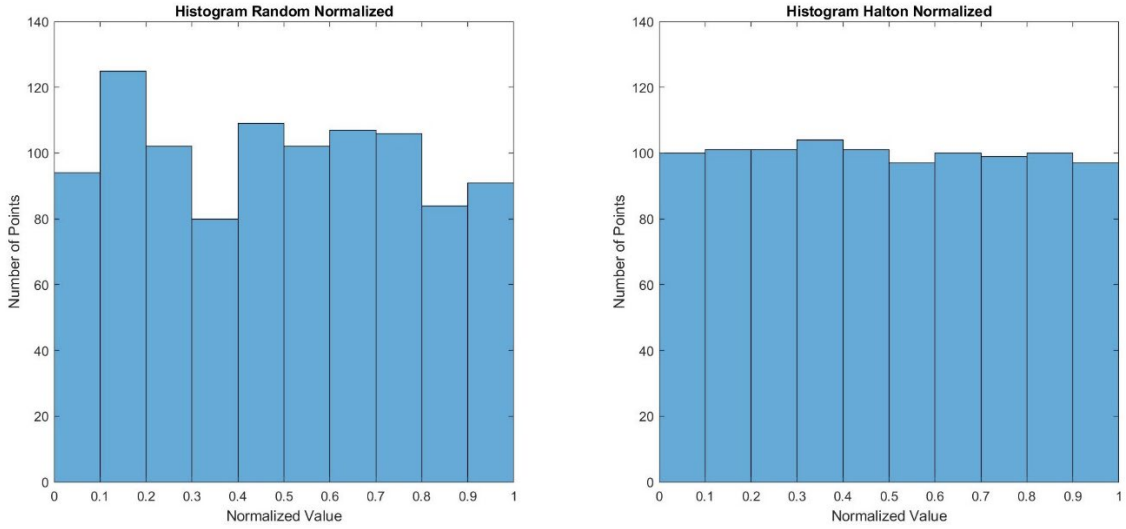


Figure 14: Histogram of 1000 normalized samples in a 10-dimensional state space from pseudo-random (left) and Halton sequence (right)

If the samples are not uniformly distributed in the state space, then a result as in Figure 15 will be observed. In this instance, a Halton sequence was generated using a poorly chosen leap value of 1000 which, when combined with a skip value of 1000 and the HaltonRR2 scramble, induced some secondary cycling which greatly impacted dimensions 4 (Fog Falloff), 5 (Precipitation), and 6 (Precipitation Deposits), resulting in those dimensions not being uniformly covered throughout their entire range. The dimension representing precipitation, for example, had a minimum value of 63.65 and a maximum value of 72.71, meaning that every scenario which was generated had a similar amount of precipitation present, which would not have properly evaluated the performance of the SUT in all precipitation conditions. This also would have biased the overall final score of the vehicle, as the SUT’s performance in precipitation would most likely be poorer than in clear conditions due to it being a solely vision-based system.

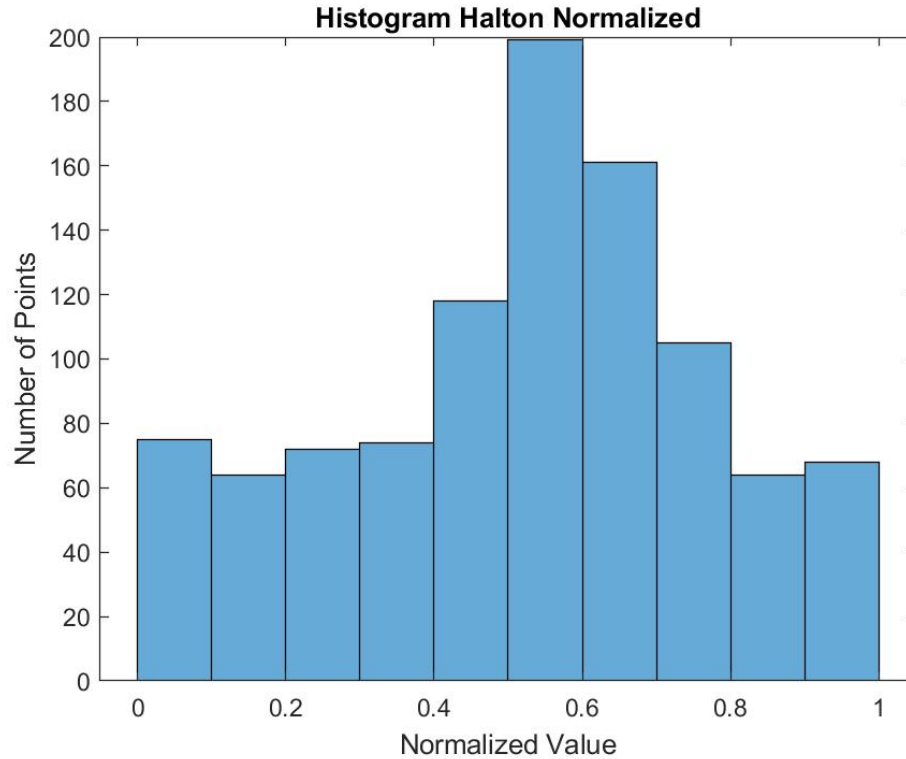


Figure 15: Histogram of 1000 normalized samples from a Halton sequence with a poorly chosen leap value (1000) in a 10-dimensional state space

This shows that great care must be taken when applying any sampling method for generating scenarios, as the resulting bias could potentially skew the perceived performance of the SUT. In Section 4.2 the effects of biased scenario generation will be discussed in greater detail.

The final values chosen to generate the Halton sequence were a skip value of 20 and a leap value of 0. After the sequence was generated, the HaltonRR2 scramble was applied. This resulted in uniform coverage with limited correlation between dimensions. A total of 800 scenarios were generated using the Halton sequence and 800 were generated using the pseudo-random generator in MATLAB. While it is impractical to inspect every dimension, a few dimensions can be selected to check for correlation

between dimensions. This will at least give additional insight into how well the Halton sequence generation method performed against the pseudo-random method. Below in Figure 16, cloudiness and fog density are plotted against each other, as well as fog density versus sun altitude angle for two sets of 800 scenarios, one set generated by a Halton sequence, with the other pseudo-randomly generated.

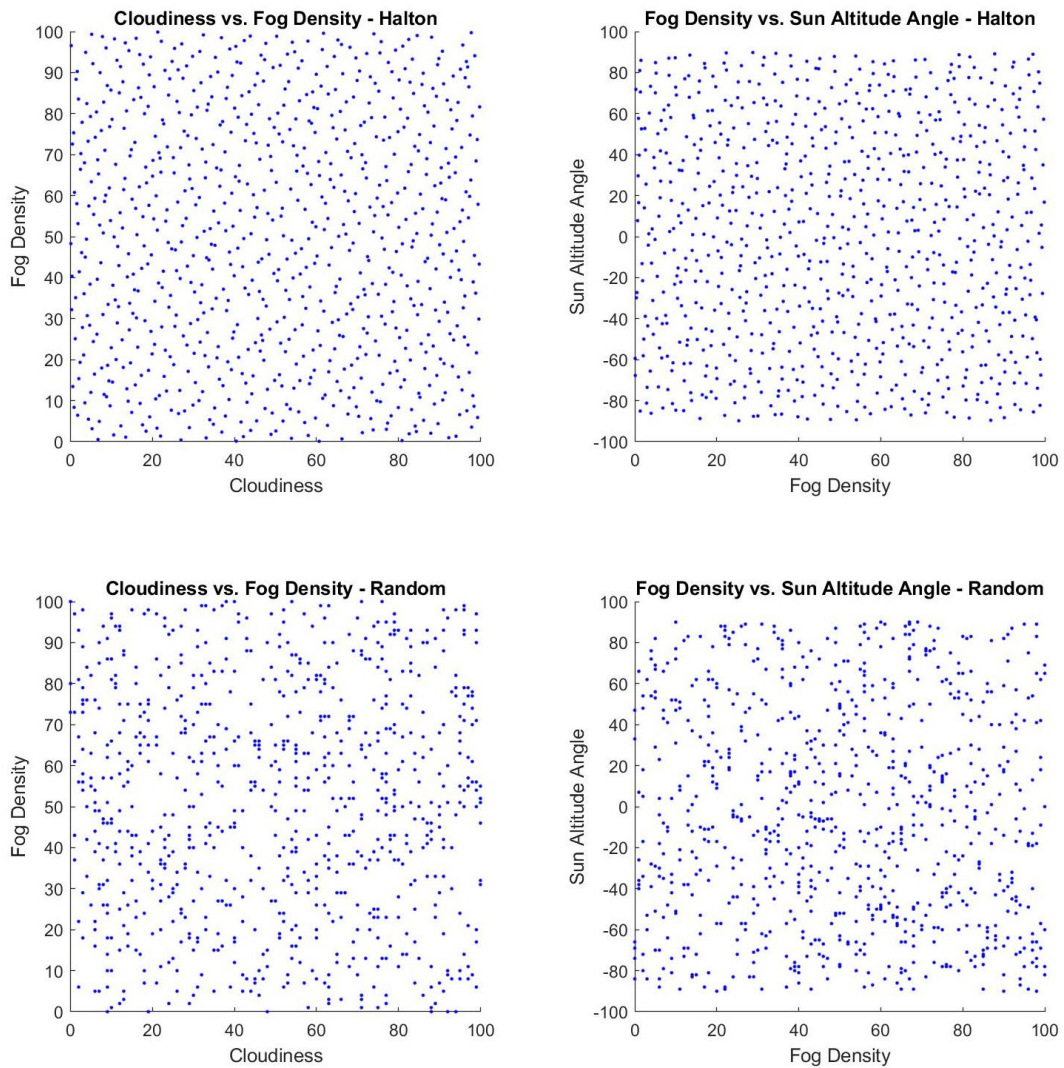


Figure 16: Cloudiness vs. Fog Density and Fog Density vs. Sun Altitude Angle for 800 scenarios generated by a Halton sequence and 800 scenarios which were pseudo-randomly generated

This shows the much more uniform distribution of samples generated by the Halton sequence over these dimensions. This means that in the scenario set generated by the Halton sequence, the SUT experienced a similar number of scenarios with high density fog around noon (sun altitude angle equal to 90) as it experienced around midnight (sun altitude angle equal to -90). The pseudo-random generated set of scenarios, though, could have gaps or bias towards each side. When looking at the histogram for sun altitude, as in Figure 17, it is demonstrated that there were more scenarios that occurred at night rather than during the day in the pseudo-randomly generated set. This can be further shown by taking the mean of each dataset. The mean for the sun altitude angle for the set of 800 pseudo-randomly generated scenarios was -3.82 while the mean for the set of 800 scenarios generated by the Halton sequence was 0.06. Also, the variance of the number of samples in each bin for the Halton and random sets were 0.667 and 100.2, respectively. This shows that incidentally, in this scenario set, the random method was biased towards scenarios at night whereas the Halton sequence was evenly distributed.

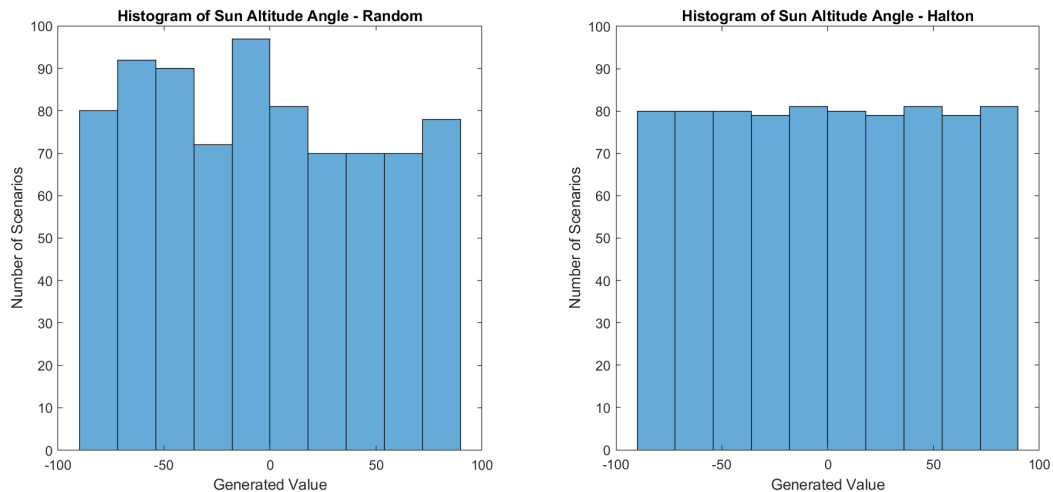


Figure 17: Histogram of sun altitude angle from 800 scenarios pseudo-randomly generated (left) and generated from the Halton sequence (right)

When examining all the dimensions, it can be shown that the mean of the pseudo-random scenario set is always further from the ideal mean than the Halton generated scenario set in these two scenario sets. The maximum percent difference from the ideal mean for the randomly generated scenario set was -4.24% while the Halton generated scenario set had a maximum percent difference of -0.29%. This means, for example, there was a 4.24% bias towards nighttime scenarios in the randomly generated scenario set, whereas the Halton sequence had only a 0.07% bias towards scenarios occurring during the daytime.

Table 4.1

Mean of 800 Halton and pseudo-randomly generated scenarios and their percent difference from the ideal mean

	Ideal Mean	Random Mean	Random % Difference	Halton Mean	Halton % Difference
Cloudiness	50.00	50.17	0.33%	49.90	-0.20%
Fog Density	50.00	50.81	1.62%	49.85	-0.29%
Fog Distance	2.50	2.46	-1.76%	2.50	-0.04%
Fog Falloff	2.50	2.58	3.34%	2.49	-0.24%
Precipitation	50.00	51.05	2.10%	49.97	-0.07%
Precipitation Deposits	50.00	50.67	1.33%	49.89	-0.23%
Wetness	50.00	49.93	-0.14%	49.95	-0.10%
Wind Intensity	50.00	51.24	2.48%	50.04	0.07%
Sun Azimuth Angle	90.00	87.11	-3.21%	89.89	-0.13%
Sun Altitude Angle	0.00	-3.82	-4.24%	0.06	0.07%

4.2 SUT Performance

The performance of the SUT can be examined in a few different ways: the overall mean driving score for the entire scenario set, the relationship between driving score and a specific simulation parameter, and the average number of infractions for each infraction type. This should give a decent overview of the SUT performance along the tested route and how different simulation parameters impact the driving score.

The overall mean driving score for the SUT in the Halton generated scenario set was 9.36 and was 8.79 for the pseudo-randomly generated scenario set after 800 scenarios. As defined in Section 3.3, these scores are on a range of 0-100, with 100 being a perfect score and 0 being the worst score. In Figure 18, the mean driving score after each of the 800 generated scenarios for both sets can be seen. The two responses show that eventually, both scenario sets begin to come to a similar agreement on the mean driving score. The percent difference between the two resulting mean driving scores after 400 scenarios was 21.1% while the percent difference after all 800 scenarios was 6.27%. It took 381 scenarios for the Halton generated scenario set to converge to within 5% of the final mean score, whereas the pseudo-randomly generated scenario set converged to within 5% of the final mean score after 673 scenarios. The plot of percent difference of the current mean driving score from the final mean driving score can be seen in Figure 19. Additionally, a histogram of the driving scores for both the pseudo-random and Halton scenario sets is shown in Figure 20.

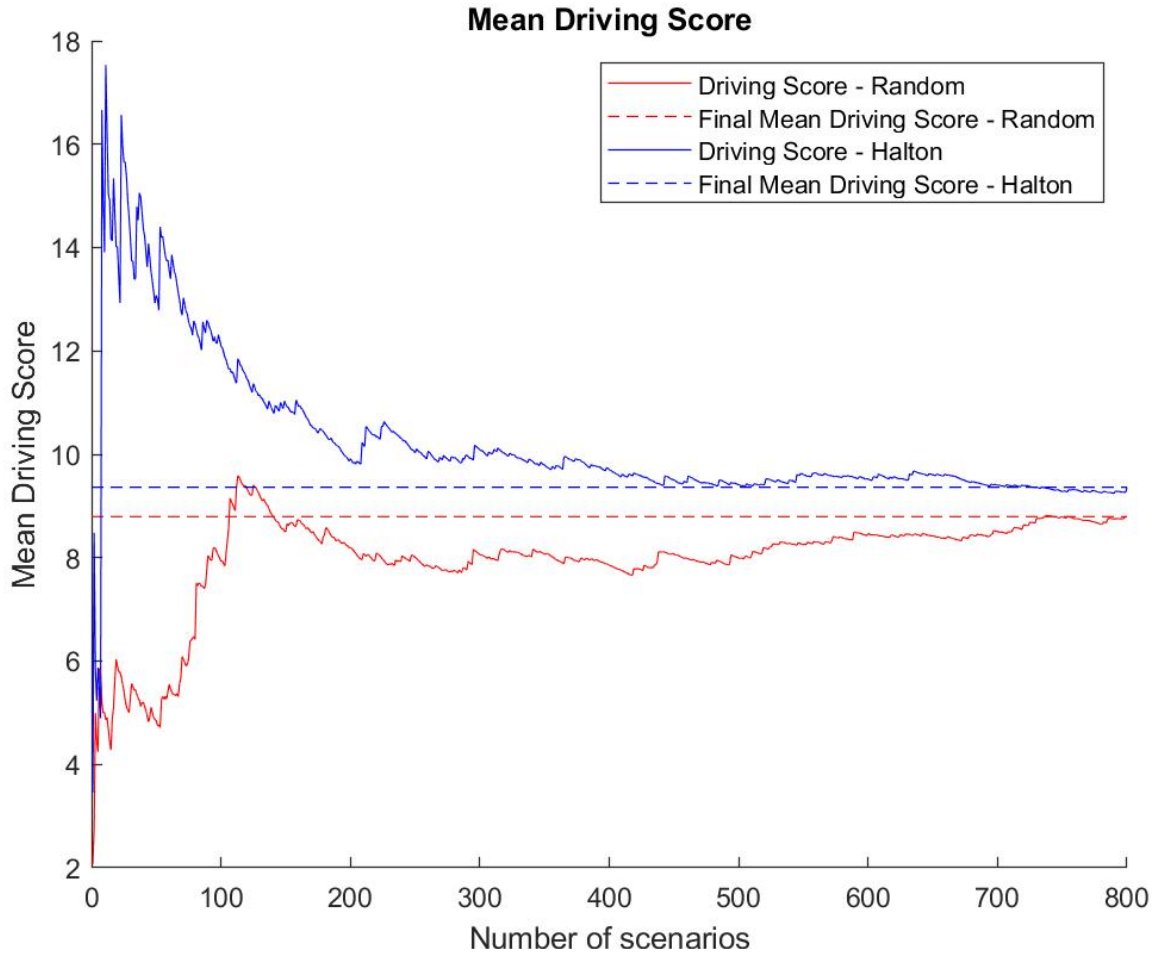


Figure 18: Mean driving score for 800 scenarios generated pseudo-randomly and by a Halton sequence

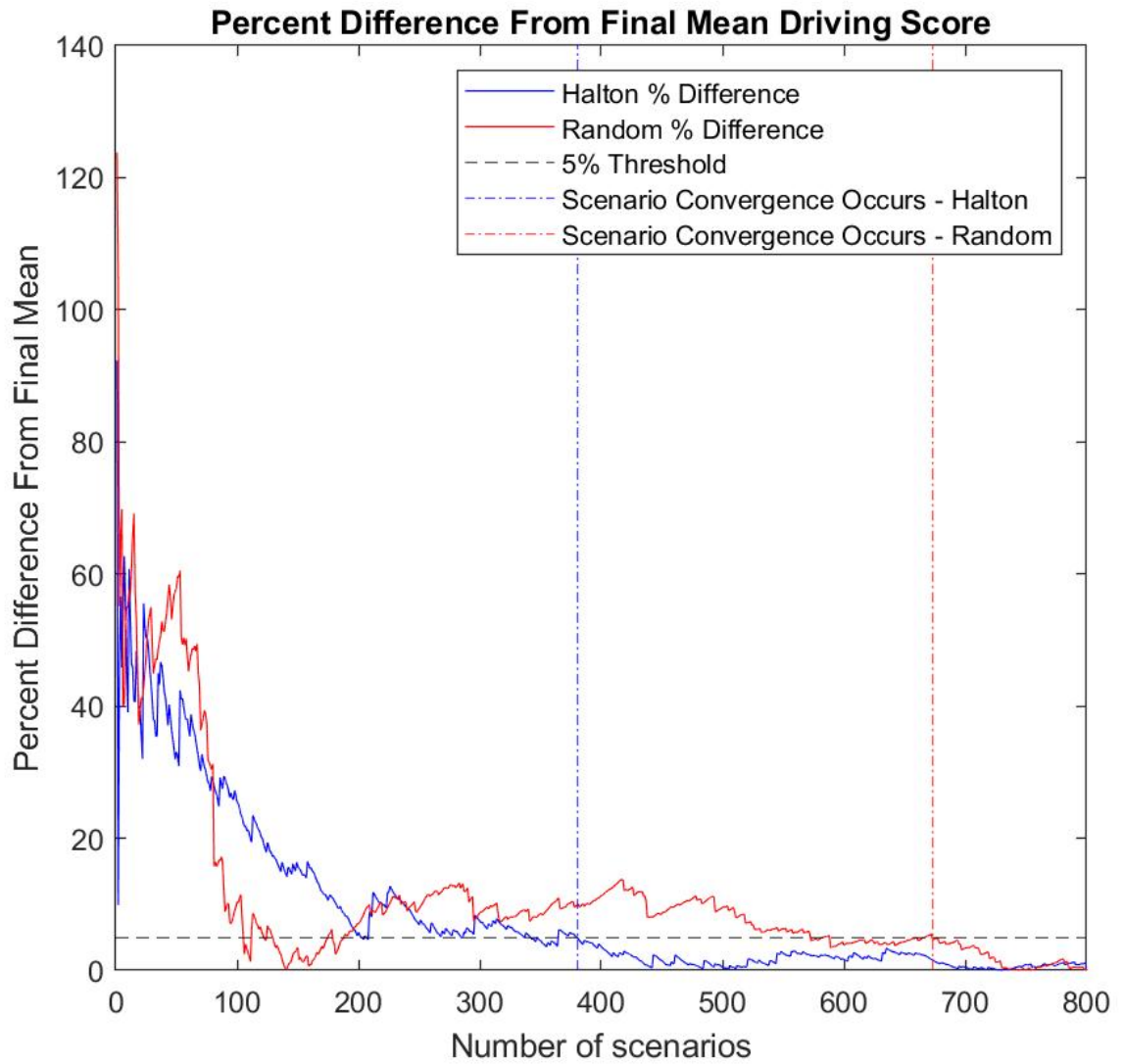


Figure 19: Percent difference between running mean and final mean driving score

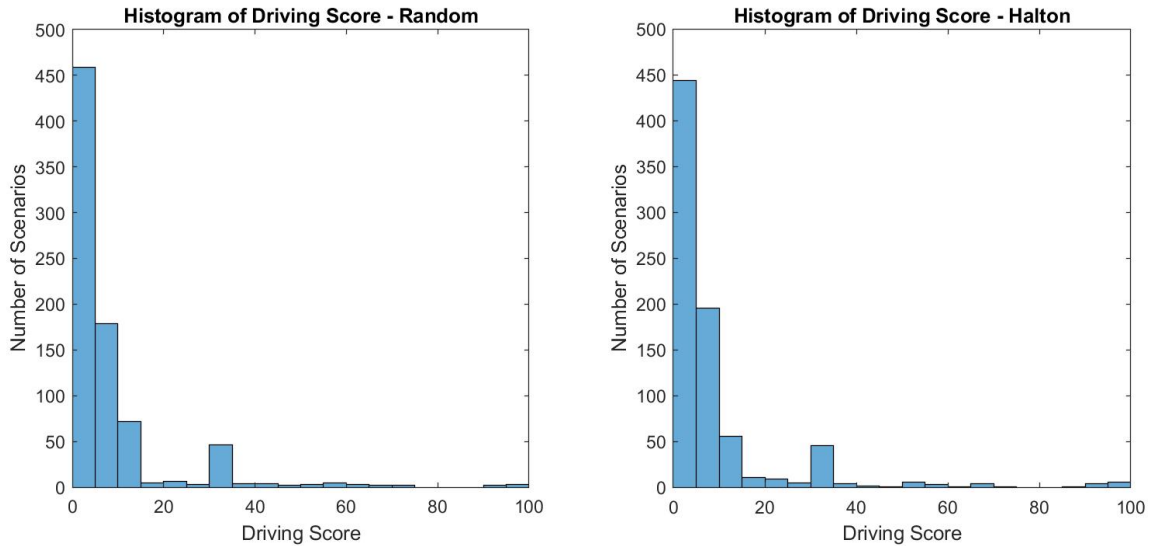


Figure 20: Histogram of driving scores for the pseudo-random scenario set (left) and the Halton scenario set (right)

While these plots give an overview of the overall performance of the SUT, they do not provide as much feedback for specific improvement areas. To get this, the driving score in different dimensions can be examined to determine if there is any correlation. For parameters sun altitude angle and fog density, in Figure 21 and Figure 22, respectively, there was some correlation with driving score. With sun altitude angle, there was a sharp drop off in vehicle performance below a sun altitude angle of 0. In the Halton generated set, no scenarios with a sun altitude angle of less than or equal to -7 achieved a driving score greater than 14.45. In the pseudo-randomly generated set, there was one outlier scenario which was able to achieve a driving score of 33.89 with a sun altitude angle of -57. However, the same general trend of poor performance in nighttime scenarios as observed in the Halton generated set was also prevalent in the pseudo-randomly generated set.

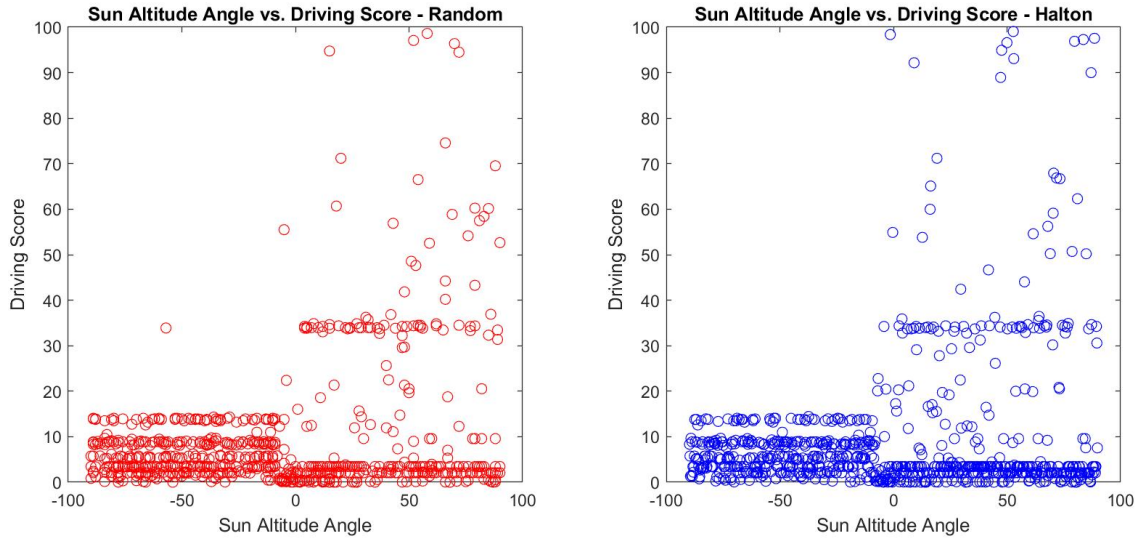


Figure 21: Sun altitude angle vs driving score for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

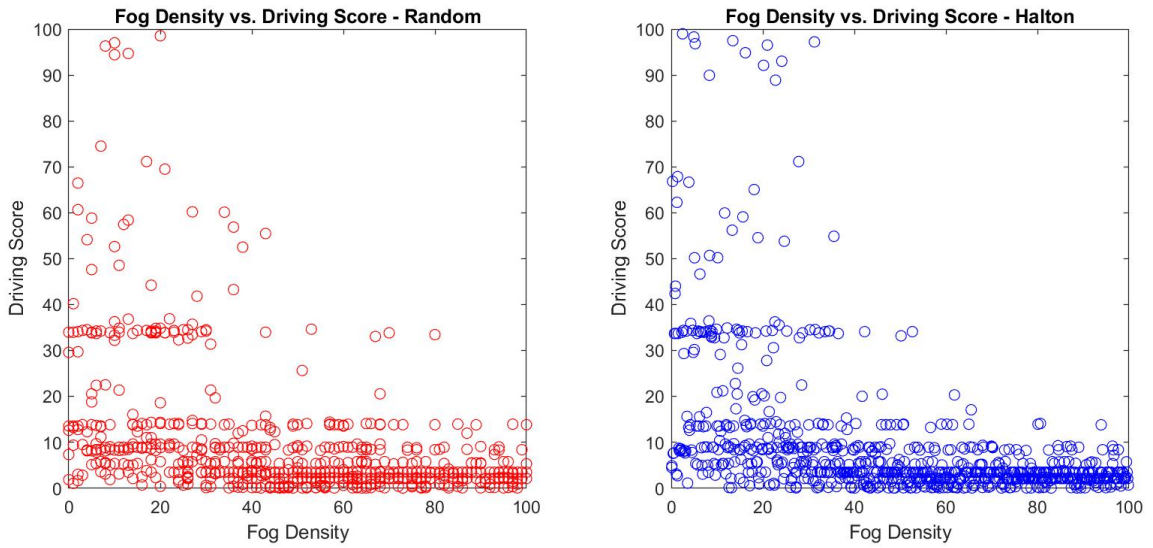


Figure 22: Fog density vs driving score for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

A more gradual decline in performance of the SUT was observed for the fog density parameter. For a fog density of greater than 80, both scenario sets did not encounter a driving score above 15. Examining each scenario set after 800 scenarios gives a similar amount of insight into the capabilities and limitations of the system. However, a better way to explore the performance of both scenario generation methods might be to examine how well each method portrays the capabilities of the system as the number of scenarios run increases. Below in Figure 23 through Figure 26, a quadratic polynomial trendline is fit to the driving score and sun altitude angle after 50, 100, 400, and 800 scenarios for each scenario generation method. Through visual inspection, the trend of the SUT's performance after 50 scenarios is generally the same as after the 800 scenarios for the Halton generation method, whereas the pseudo-random method has a trend after 50 and 100 scenarios which would indicate decreasing performance of the SUT as the sun altitude angle increases. In Figure 25, which is after 400 scenarios, the pseudo-random scenario set indicates a similar trend to the Halton generated set. After 800 scenarios, both methods are in general agreement of the trend of the SUT's performance over a varying sun altitude angle. This is most likely due to the Halton sequences more uniform coverage of the state space, as discussed in Section 4.1.

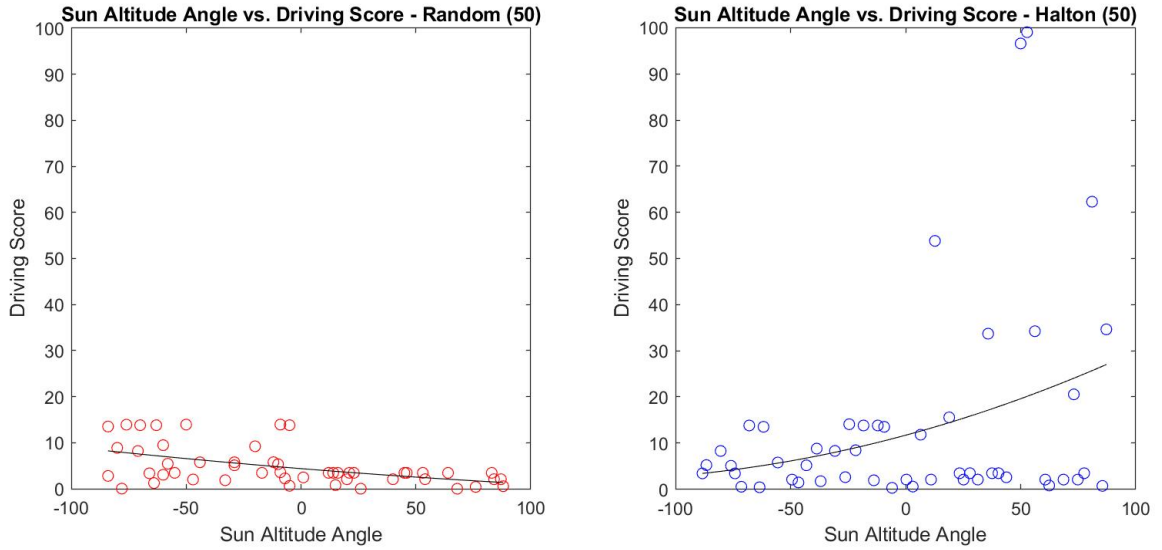


Figure 23: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 50 scenarios

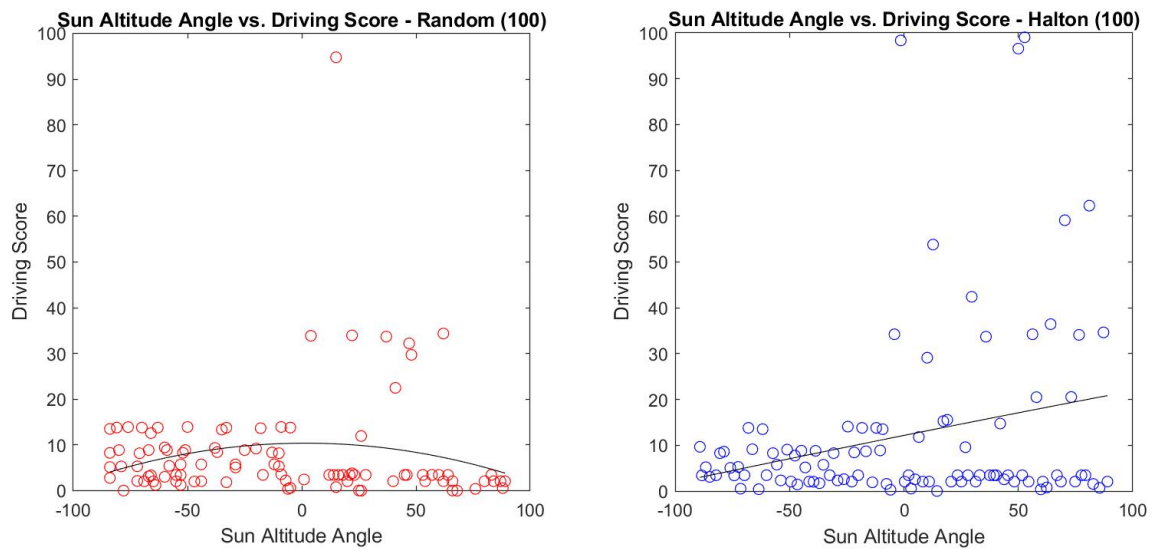


Figure 24: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 100 scenarios

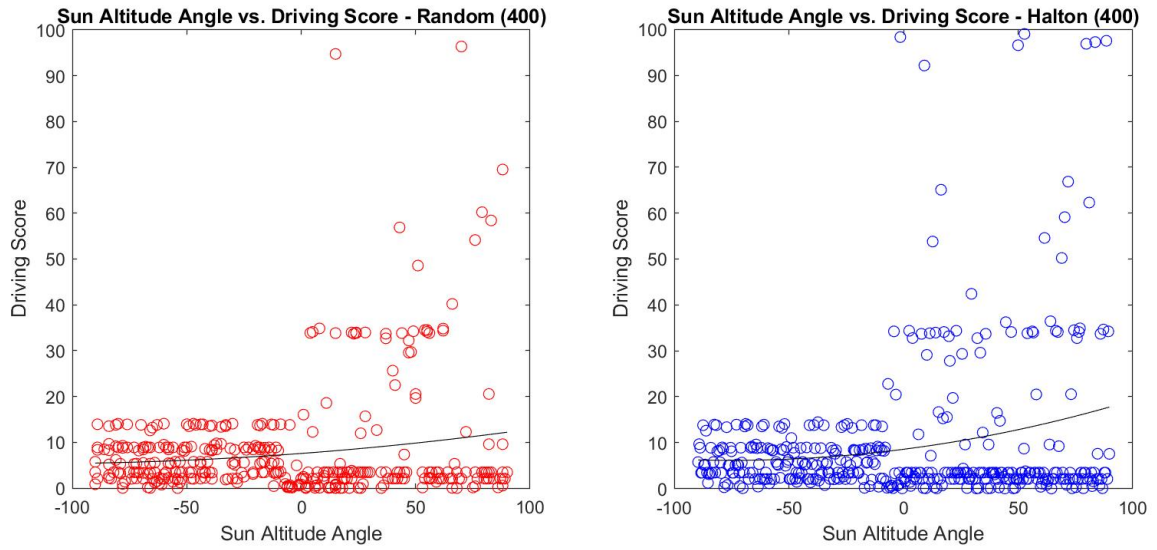


Figure 25: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 400 scenarios

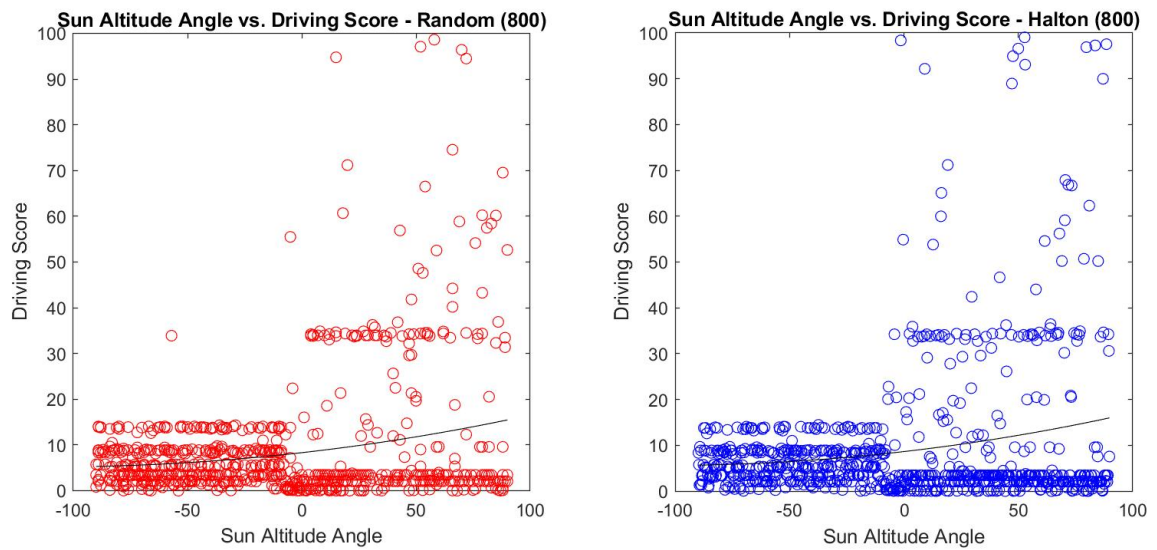


Figure 26: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

While some dimensions, specifically sun altitude angle and fog density, had a fairly large impact on the overall driving score, other dimensions seemed to have little to no effect on the final driving score. For example, neither cloudiness nor wind intensity appeared to have even a marginal, if any, impact on driving score. Appendix A contains a plot of each of these dimensions after 800 scenarios.

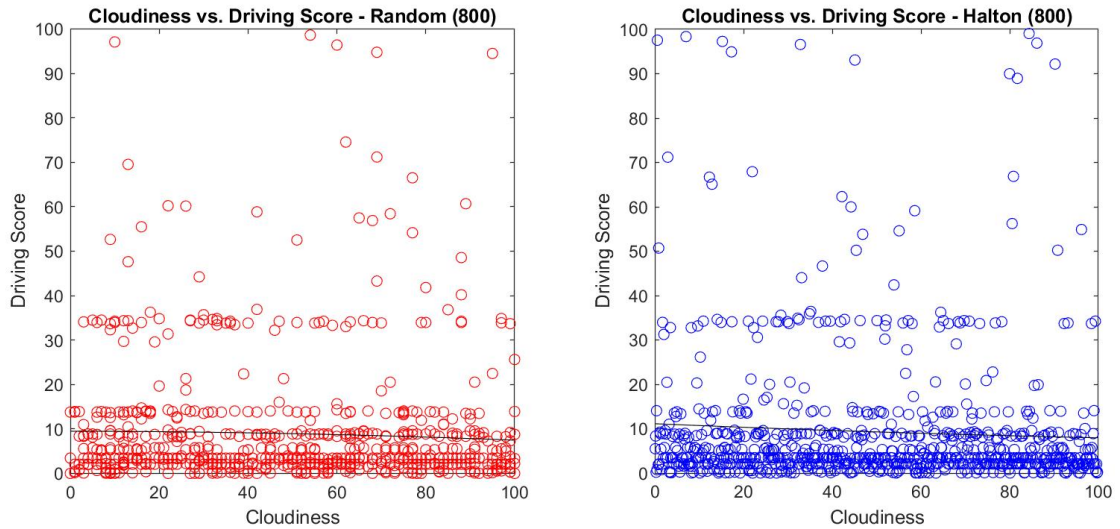


Figure 27: Cloudiness vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

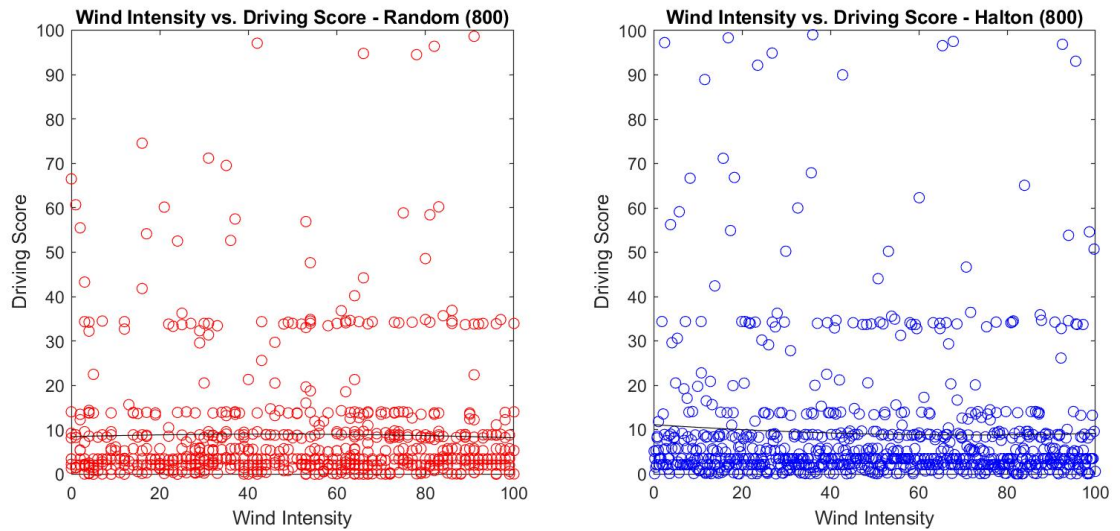


Figure 28: Wind intensity vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

Since it has now been shown that certain parameters have a larger impact on the overall driving score, the difference of 6.27% between the final overall driving scores (Figure 18) from the Halton scenario set and the pseudo-randomly generated scenario set can be rationalized. As discussed in Section 4.1, the pseudo-random scenario set had more bias in each dimension than the Halton scenario set (shown in Table 4.1). Take, for example, sun altitude angle, with which the pseudo-random scenario set had a bias of 4.24% towards scenarios occurring at night (meaning a sun altitude angle of less than 0) and the Halton scenario set had a bias of 0.07% towards daytime scenarios. It has been shown that the SUT had a poorer performance in scenarios occurring at night. Therefore, it makes sense that the pseudo-random set had a lower overall mean driving score than the Halton set, as it was biased towards scenarios with which the SUT performs poorly, whereas the Halton set had a smaller, nearly negligible bias. More samples towards either end of the spectrum can cause the testing to result in either an over estimation of the SUT's abilities if more scenarios with more favorable conditions are generated, or an under-estimation if there are more scenarios that are difficult for the SUT to perform well in. While this bias could have caused the overall mean driving score of the pseudo-random set to have been lower, other factors, which will be discussed in Section 4.3, could have impacted these results.

4.3 Performance Variance

If the Halton sequence generated scenario set more uniformly distributed samples in the state space, then it should also produce a higher variance in the output space. This higher variance would be indicative of a more diverse scenario set that properly tested the system both in scenarios where it would perform well and where it would perform poorly. Therefore, in Figure 29, the running variance of the scenarios run thus far is calculated and then plotted every 10 scenarios. For instance, the first sample point on the plot is the variance from scenarios 1 through 10, next is the from scenarios 1 through 20, then scenarios 1 through 30, etc.

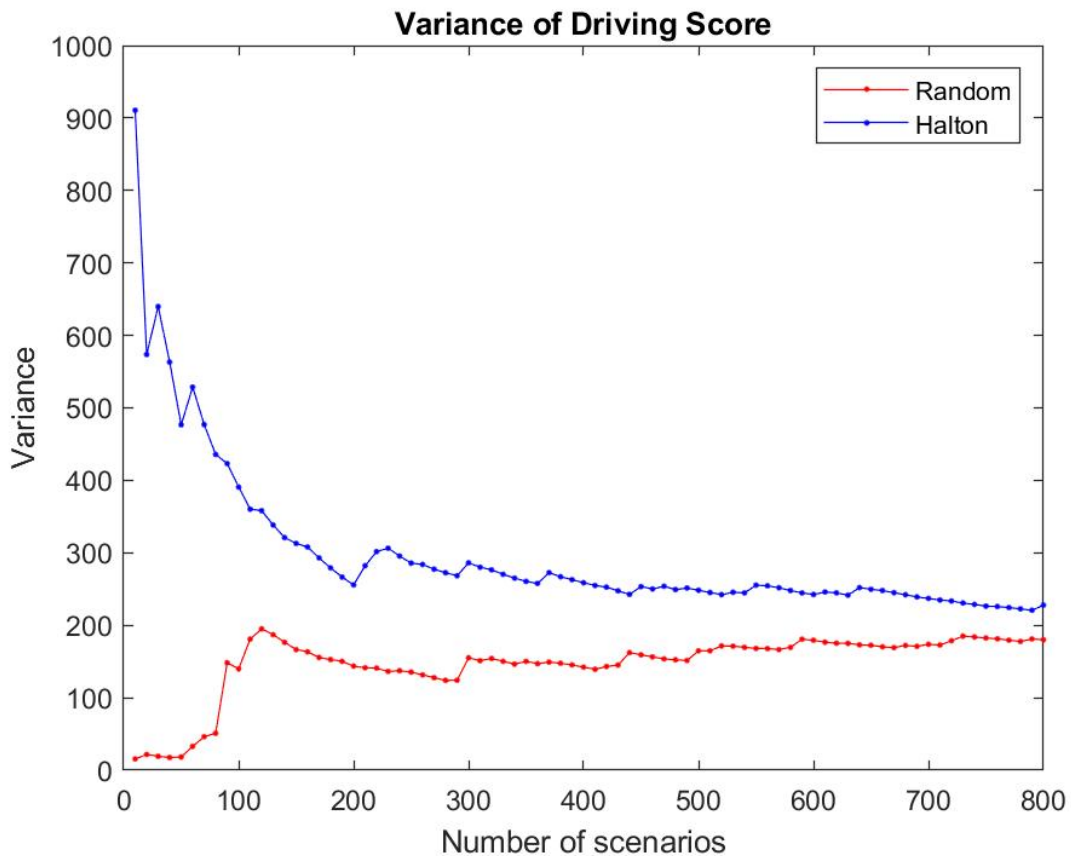


Figure 29: Variance of Halton and pseudo-random driving score

The variance of the driving score from the Halton scenario set after the first 10 scenarios was 910.8, whereas the variance from the pseudo-random set was 15.7. As the number of scenarios increase, the variance of the Halton scenario set decreases, with occasional spikes up. The variance of the pseudo-random set, however, starts from a fairly low variance and increases. This further reinforces the claim that the Halton sequence is more uniformly testing the SUT, whereas the pseudo-random set might be testing redundant scenarios. The Halton scenario set ends at 800 scenarios with a variance of 227.8 while the pseudo-random set ends with a variance of 180.0.

4.4 Testing Variability

As mentioned in Section 3.1, one of the extraneous variables is the impact of non-determinism on the testing results. First, the variability of the results will be shown and then the potential causes will be discussed. To characterize the variability of the results, the same set of 50 scenarios were run 4 different times. In Figure 30, the rolling mean driving score for these same 50 scenarios is exhibited. The final mean driving scores for runs 1 through 4, in order, were: 9.25, 7.02, 7.91, 7.20. This gives a final aggregate mean score between all 4 runs of 7.85. The resulting variance of the final mean driving score across all runs was 1.02. The variability in these results show that there is in fact variability when running the same scenario to evaluate the SUT. While there are still many scenarios that, across all four runs, result in the same driving score, there are some which can experience wildly different outcomes. A selection of the scores for individual scenarios and their variance can be seen in Table 4.2, whereas the full dataset can be found in 0.

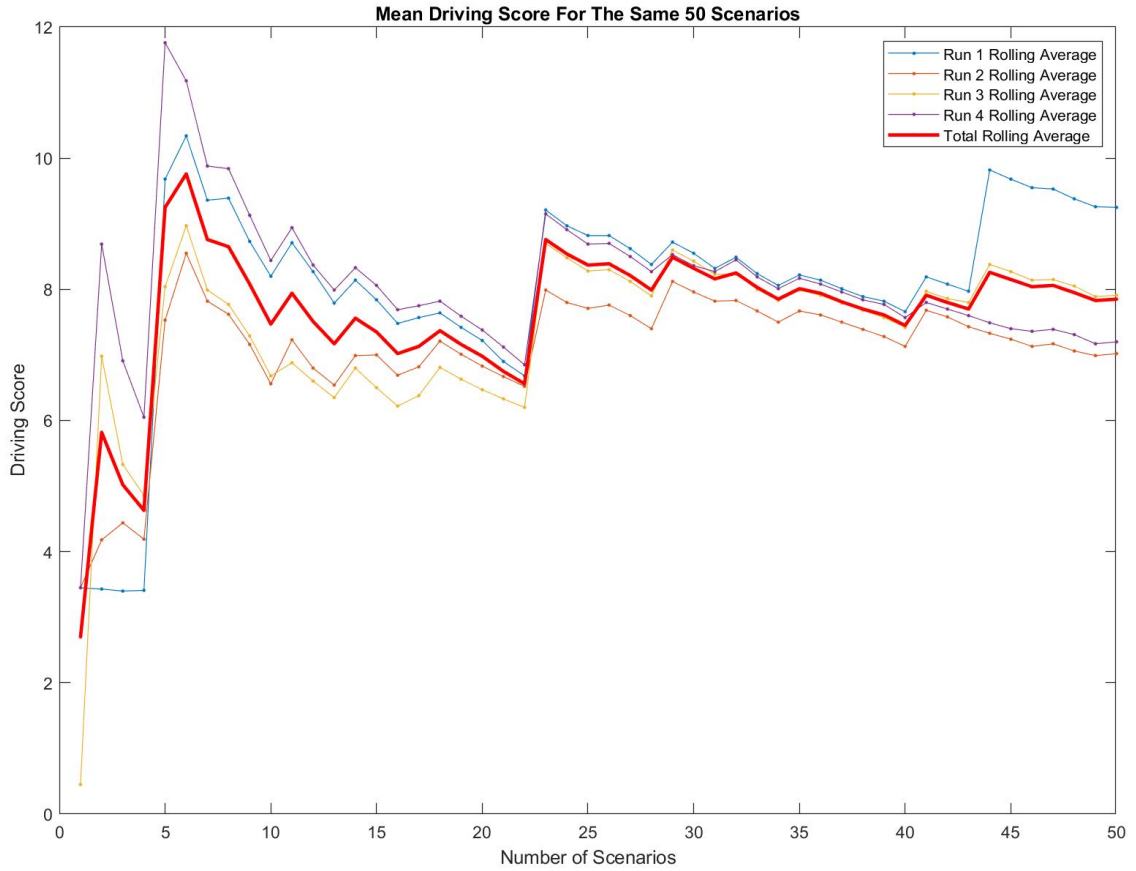


Figure 30: Mean driving score across 4 repeated runs of the same set of 50 scenarios

Table 4.2

A subset of 10 scenarios of the results of 50 scenarios run 4 times

Scenario ID	Run 1	Run 2	Run 3	Run 4	Variance
1	3.45	3.45	0.45	3.45	2.25
2	3.40	4.91	13.51	13.93	30.90
3	3.35	4.96	2.03	3.35	1.44
4	3.45	3.45	3.45	3.45	0.00
5	34.77	20.86	20.76	34.63	64.31
6	13.65	13.65	13.65	8.27	7.23
7	3.45	3.45	2.07	2.07	0.63
8	9.58	6.23	6.23	9.58	3.75
9	3.45	3.45	3.45	3.45	0.00
10	3.45	1.23	1.22	2.22	1.11

Since the SUT uses machine learning, the core element of the driving model is inherently non-deterministic, meaning that given the same set of inputs this driving model could react in a completely different way. Another non-deterministic aspect of the testing process is the simulator itself, specifically the traffic manager. Meaning multiple different runs on the simulator may result in actors not behaving the same exact way, in turn changing the SUT's experience on the route and causing different inputs that should have been consistent. While this is more impactful on the outcomes of individual or small test scenarios, the variance of 1.02 across the final means scores of all 4 runs show that the effects of this most likely even out across the aggregate scenario set. However, the same scenario set should probably be re-run multiple times for more accurate testing results. Re-running the same set of 800 scenarios was infeasible for this study due to the long testing time, which will be mentioned in the next section.

4.5 Testing Time

Another datapoint that should be discussed is the amount of time it took to run these scenarios using the hardware and software listed in Table 3.1. The time it took to run 800 scenarios was approximately 72 hours, resulting in an average time of 5.4 minutes per scenario. The time to generate scenarios using the Halton sequence and the pseudo-random method was effectively negligible compared to the testing time. Both took essentially the same amount of time to generate 800 scenarios, around 3.6 seconds on average. This long testing time is the reason why the number of 800 scenarios for each method was chosen. This long testing time for each scenario also underscores the benefit of using the Halton sequence generation method, as the vehicles performance was characterized in fewer scenarios than the pseudo-random method.

5 Conclusions

The future of AV's will depend on their ability to be fully and effectively tested, and simulation will play a large role in this testing regime [6], [9], [12], [16]. This study has proposed and evaluated using Halton sequences to generate scenarios for testing AD systems in simulation. The method was compared to traditional pseudo-random scenario generation and evaluated for uniformity of the coverage of input space and its ability to accurately evaluate the SUT's performance.

Halton sequences were chosen because of their low-discrepancy in smaller state spaces, and their ability to expand their low-discrepancy characteristics to higher dimensional state spaces by using a scrambling method, namely the HaltonRR2 method [49], [50], [52]. Halton sequences were shown to have a more uniform distribution of the state space and converge to within 5% of the final driving score in 381 scenarios, whereas the pseudo-random method took 673. It additionally showed that the general trend of the SUT's performance in a certain dimension was correctly characterized by the Halton method before the pseudo-random generation method. The Halton scenario set also had a higher variance than the pseudo-random method, especially during the initial scenarios. This indicates that the Halton scenario set generated a more diverse set of scenarios to evaluate the SUT, finding both scenarios where the SUT performed poorly, and where it performed well. While it was found that there is variability in the test results and that running the same scenario can yield different outputs each time, the variance of the aggregate driving score over many scenarios was relatively low.

As mentioned in Section 3.1, attributes of good test cases are effectiveness, efficiency, economy, and robustness [44]. The Halton sequence generation method was

effective as it uniformly covered the state space, testing a diverse set of scenarios. It was also more efficient than the pseudo-random generation as it took nearly 300 fewer scenarios to converge to a final score, and it took fewer scenarios to characterize the SUT's performance in relation to each parameter. It was just as economical as the pseudo-random generation method, as each took essentially the same amount of time to generate the set of scenarios. Finally, a Halton sequence should be resilient to changes in the state space, so long as appropriate methods are used to prevent correlation between higher dimensions and avoid secondary cycling. Overall, this study finds that it is advantageous for AV testing in simulation to use Halton sequences for scenario generation over pseudo-random methods because of its uniform state space coverage and testing efficiency.

6 Future Work

While the use of Halton sequences to generate a uniform, diverse scenario set has been shown to be promising in this paper, their performance and application should be further studied. First, more analysis of the performance of Halton sequences should be completed across many other vehicle models, and the same scenario sets should be re-run multiple times to provide statistical significance. A larger set of scenarios to evaluate performance beyond the 800 scenarios generated, as well as scenarios with larger state spaces should also be generated and evaluated.

Additionally, the scenarios generated here were focused on weather variation along a route, however this could easily be expanded well beyond weather parameters. For example, consider a "Cut-In" type scenario where the SUT is travelling along a road and

another actor cuts in front of it, something most drivers on the road have probably experienced at some point. Halton sequences could be applied to vary parameters such as velocity of the SUT, velocity of the other actor, distance at cut-in, velocity after cut-in, in addition to the weather parameters. The batches of scenarios which would be shorter than an entire route, but cover a larger state space, would be an interesting application of Halton sequences.

The performance of other low-discrepancy, quasi-random sequences should be evaluated and compared to Halton and pseudo-random scenario generation. Sequences such as Sobol, Hammersley, Faure or some of the many others should be explored.

Finally, a uniform distribution, such as the Halton sequence provides, might not always be the preferred way to test. For example, a vehicle most likely does not experience the same number of scenarios with heavy rain as it experiences a clear, sunny day. Characterizing the distribution of scenarios in the ODD, while not a trivial task, should also be investigated. If at least a rudimentary representation of the distribution of scenarios in the ODD could be characterized, then an overall driving score would be more representative of the vehicle's ability in the real world. This would not replace the benefit of evaluating the vehicles performance using a uniform distribution but could supplement the estimated performance of the vehicle in the real world.

7 References

- [1] National Center for Statistics and Analysis, “2018 Fatal Motor Vehicle Crashes: Overview,” National Highway Traffic Safety Administration, Washington, DC, Traffic Safety Facts Research Note DOT HS 812 826, Oct. 2019.
- [2] National Center for Statistics and Analysis, “Police-Reported Motor Vehicle Traffic Crashes in 2018,” National Highway Traffic Safety Administration, Washington, DC, Traffic Safety Facts Research Note DOT HS 812 860, Nov. 2019. [Online]. Available: <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812860>.
- [3] L. Blincoe, T. R. Miller, E. Zaloshnja, and B. A. Lawrence, “The economic and societal impact of motor vehicle crashes, 2010 (Revised),” National Highway Traffic Safety Administration, Washington, DC, DOT HS 812 013, May 2015.
- [4] S. Singh, “Critical reasons for crashes investigated in the National Motor Vehicle Crash Causation Survey,” National Highway Traffic Safety Administration, Washington, DC, Traffic Safety Facts Crash•Stats DOT HS 812 115, Feb. 2015.
- [5] U.S. Department of Transportation, *Preparing for the Future of Transportation: Automated Vehicles 3.0*. U.S. Department of Transportation, 2018.
- [6] T. Zhang, D. Tao, X. Qu, X. Zhang, R. Lin, and W. Zhang, “The roles of initial trust and perceived risk in public’s acceptance of automated vehicles,” *Transp. Res. Part C Emerg. Technol.*, vol. 98, pp. 207–220, Jan. 2019, doi: 10.1016/j.trc.2018.11.018.
- [7] A. Smith and M. Anderson, “Automation in everyday life,” *Wash. Pew Res. Cent.*, 2017.
- [8] P. Du and K. Driggs-Campbell, “Finding Diverse Failure Scenarios in Autonomous Systems Using Adaptive Stress Testing,” *SAE Int. J. Connect. Autom. Veh.*, vol. 2, no. 4, pp. 12-02-04–0018, Dec. 2019, doi: 10.4271/12-02-04-0018.
- [9] D. J. Fremont *et al.*, “Formal Scenario-Based Testing of Autonomous Vehicles: From Simulation to the Real World,” *ArXiv200307739 Cs Eess*, Mar. 2020, Accessed: Apr. 12, 2020. [Online]. Available: <http://arxiv.org/abs/2003.07739>.
- [10] P. Helle, W. Schamai, and C. Strobel, “Testing of Autonomous Systems - Challenges and Current State-of-the-Art,” *INCOSE Int. Symp.*, vol. 26, no. 1, pp. 571–584, Jul. 2016, doi: 10.1002/j.2334-5837.2016.00179.x.
- [11] S. Wagner, A. Knoll, K. Groh, T. Kühbeck, D. Watzenig, and L. Eckstein, “Virtual Assessment of Automated Driving: Methodology, Challenges, and Lessons Learned,” *SAE Int. J. Connect. Autom. Veh.*, vol. 2, no. 4, pp. 12-02-04–0020, Dec. 2019, doi: 10.4271/12-02-04-0020.
- [12] N. Kalra and S. M. Paddock, “Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?,” *Transp. Res. Part Policy Pract.*, vol. 94, pp. 182–193, Dec. 2016, doi: 10.1016/j.tra.2016.09.010.
- [13] Florida Polytechnic Univ and R. Razdan, “Unsettled Technology Areas in Autonomous Vehicle Test and Validation,” SAE International, Jun. 2019. doi: 10.4271/epr2019001.
- [14] “Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian, Tempe, Arizona, March 18, 2018,” National Transportation Safety Board, Washington, DC, Highway Accident Report NTSB/HAR-19/03, Nov. 2019. [Online]. Available: <https://www.nts.gov/investigations/AccidentReports/Reports/HAR1903.pdf>.

- [15] T. F. Koné, E. Bonjour, E. Levrat, F. Mayer, and S. Géronimi, “Challenges for Autonomous Vehicles (AVs) Engineering: Safety Validation of Functional Performance Limitations,” *INSIGHT*, vol. 22, no. 4, pp. 23–25, Dec. 2019, doi: 10.1002/inst.12270.
- [16] P. Koopman and M. Wagner, “Challenges in Autonomous Vehicle Testing and Validation,” *SAE Int. J. Transp. Saf.*, vol. 4, no. 1, pp. 15–24, Apr. 2016, doi: 10.4271/2016-01-0128.
- [17] “UL 4600.” Underwriters Laboratories, Dec. 13, 2019, Accessed: Jan. 10, 2020. [Online]. Available: https://edge-case-research.com/wp-content/uploads/2019/12/191213_UL4600_VotingVersion.pdf.
- [18] G. Brat and A. Jonsson, “Challenges in verification and validation of autonomous systems for space exploration,” in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, MONTreal, QC, Canada, 2005, vol. 5, pp. 2909–2914, doi: 10.1109/IJCNN.2005.1556387.
- [19] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *ArXiv180402767 Cs*, Apr. 2018, Accessed: Mar. 23, 2020. [Online]. Available: <http://arxiv.org/abs/1804.02767>.
- [20] T. Siméon, J.-P. Laumond, and C. Nissoux, “Visibility-based probabilistic roadmaps for motion planning,” *Adv. Robot.*, vol. 14, no. 6, pp. 477–493, Jan. 2000, doi: 10.1163/156855300741960.
- [21] M. Utting, A. Pretschner, and B. Legeard, “A taxonomy of model-based testing approaches,” *Softw. Test. Verification Reliab.*, vol. 22, no. 5, pp. 297–312, Aug. 2012, doi: 10.1002/stvr.456.
- [22] T. Dreossi *et al.*, “VerifAI: A Toolkit for the Formal Design and Analysis of Artificial Intelligence-Based Systems,” in *Computer Aided Verification*, vol. 11561, I. Dillig and S. Tasiran, Eds. Cham: Springer International Publishing, 2019, pp. 432–442.
- [23] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, “Scenic: A Language for Scenario Specification and Scene Generation,” *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Des. Implement. - PLDI 2019*, pp. 63–78, 2019, doi: 10.1145/3314221.3314633.
- [24] S. A. E. international, “Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,” *SAE Int.*, 2016.
- [25] ISO, *Road vehicles – Functional safety*. ISO, Geneva, Switzerland, 2018.
- [26] G. Bagschik, T. Menzel, and M. Maurer, “Ontology based Scene Creation for the Development of Automated Vehicles,” *ArXiv170401006 Cs*, Apr. 2018, Accessed: Apr. 26, 2020. [Online]. Available: <http://arxiv.org/abs/1704.01006>.
- [27] K. Go and J. M. Carroll, “The blind men and the elephant: views of scenario-based system design,” *interactions*, vol. 11, no. 6, pp. 44–53, Nov. 2004, doi: 10.1145/1029036.1029037.
- [28] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, “Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Gran Canaria, Spain, Sep. 2015, pp. 982–988, doi: 10.1109/ITSC.2015.164.
- [29] “ASAM OpenSCENARIO V1.0.0.” Association for Standardization of Automation and Measuring Systems, 2020.

- [30] Y. Li, J. Tao, and F. Wotawa, “Ontology-based test generation for automated and autonomous driving functions,” *Inf. Softw. Technol.*, vol. 117, p. 106200, Jan. 2020, doi: 10.1016/j.infsof.2019.106200.
- [31] F. Klueck, Y. Li, M. Nica, J. Tao, and F. Wotawa, “Using Ontologies for Test Suites Generation for Automated and Autonomous Driving Functions,” in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Memphis, TN, Oct. 2018, pp. 118–123, doi: 10.1109/ISSREW.2018.00-20.
- [32] C. D. Nguyen, A. Perini, and P. Tonella, “Ontology-based test generation for multiagent systems,” in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, 2008, pp. 1315–1320.
- [33] R. Regele, “Using Ontology-Based Traffic Models for More Efficient Decision Making of Autonomous Vehicles,” in *Fourth International Conference on Autonomic and Autonomous Systems (ICAS’08)*, Gosier, Guadeloupe, Mar. 2008, pp. 94–99, doi: 10.1109/ICAS.2008.10.
- [34] F. Schuldt, “Ein Beitrag für den methodischen Test von automatisierten Fahrfunktionen mit Hilfe von virtuellen Umgebungen,” PhD Thesis, 2017.
- [35] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.
- [36] S. M. LaValle, J. J. Kuffner, B. Donald, and others, “Rapidly-exploring random trees: Progress and prospects,” *Algorithmic Comput. Robot. New Dir.*, no. 5, pp. 293–308, 2001.
- [37] S. M. LaValle and J. J. Kuffner, “Randomized Kinodynamic Planning,” *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001, doi: 10.1177/02783640122067453.
- [38] C. E. Tuncali and G. Fainekos, “Rapidly-exploring Random Trees-based Test Generation for Autonomous Vehicles,” *ArXiv190310629 Cs*, Mar. 2019, Accessed: Apr. 13, 2020. [Online]. Available: <http://arxiv.org/abs/1903.10629>.
- [39] M. S. Branicky, S. M. LaValle, K. Olson, and Libo Yang, “Quasi-randomized path planning,” in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, Seoul, South Korea, 2001, vol. 2, pp. 1481–1487, doi: 10.1109/ROBOT.2001.932820.
- [40] B. Park and W. K. Chung, “Efficient environment representation for mobile robot path planning using CVT-PRM with Halton sampling,” *Electron. Lett.*, vol. 48, no. 22, p. 1397, 2012, doi: 10.1049/el.2012.2894.
- [41] J. Velagic, D. Delimustafic, and D. Osmanovic, “Mobile robot navigation system based on Probabilistic Road Map (PRM) with Halton sampling of configuration space,” in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, Istanbul, Turkey, Jun. 2014, pp. 1227–1232, doi: 10.1109/ISIE.2014.6864789.
- [42] W. J. Morokoff and R. E. Caflisch, “Quasi-Random Sequences and Their Discrepancies,” *SIAM J. Sci. Comput.*, vol. 15, no. 6, pp. 1251–1279, Nov. 1994, doi: 10.1137/0915077.
- [43] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, “Paracosm: A Language and Tool for Testing Autonomous Driving Systems,” *ArXiv190201084*

- Cs, Jan. 2021, Accessed: Mar. 16, 2021. [Online]. Available: <http://arxiv.org/abs/1902.01084>.
- [44] G. Chance, A. Ghobrial, S. Lemaignan, T. Pipe, and K. Eder, “An Agency-Directed Approach to Test Generation for Simulation-based Autonomous Vehicle Verification,” *ArXiv191205434 Cs*, Feb. 2020, Accessed: Feb. 21, 2020. [Online]. Available: <http://arxiv.org/abs/1912.05434>.
- [45] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
- [46] M. Dupuis, M. Strobl, and H. Grezlikowski, “OpenDRIVE 2010 and Beyond—Status and Future of the de facto Standard for the Description of Road Networks,” in *Proc. of the Driving Simulation Conference Europe*, 2010, pp. 231–242.
- [47] “Carla Autonomous Driving Leaderboard,” Feb. 25, 2020. <https://leaderboard.carla.org/> (accessed Mar. 28, 2021).
- [48] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, “Learning by Cheating,” *ArXiv191212294 Cs*, Dec. 2019, Accessed: Mar. 17, 2021. [Online]. Available: <http://arxiv.org/abs/1912.12294>.
- [49] J. H. Halton, “On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals,” *Numer. Math.*, vol. 2, no. 1, pp. 84–90, Dec. 1960, doi: 10.1007/BF01386213.
- [50] L. Kocis and W. J. Whiten, “Computational investigations of low-discrepancy sequences,” *ACM Trans. Math. Softw.*, vol. 23, no. 2, pp. 266–294, Jun. 1997, doi: 10.1145/264029.264064.
- [51] E. Braaten and G. Weller, “An improved low-discrepancy sequence for multidimensional quasi-Monte Carlo integration,” *J. Comput. Phys.*, vol. 33, no. 2, pp. 249–258, Nov. 1979, doi: 10.1016/0021-9991(79)90019-6.
- [52] B. Vandewoestyne and R. Cools, “Good permutations for deterministic scrambled Halton sequences in terms of L2-discrepancy,” *J. Comput. Appl. Math.*, vol. 189, no. 1–2, pp. 341–361, May 2006, doi: 10.1016/j.cam.2005.05.022.

Appendix A

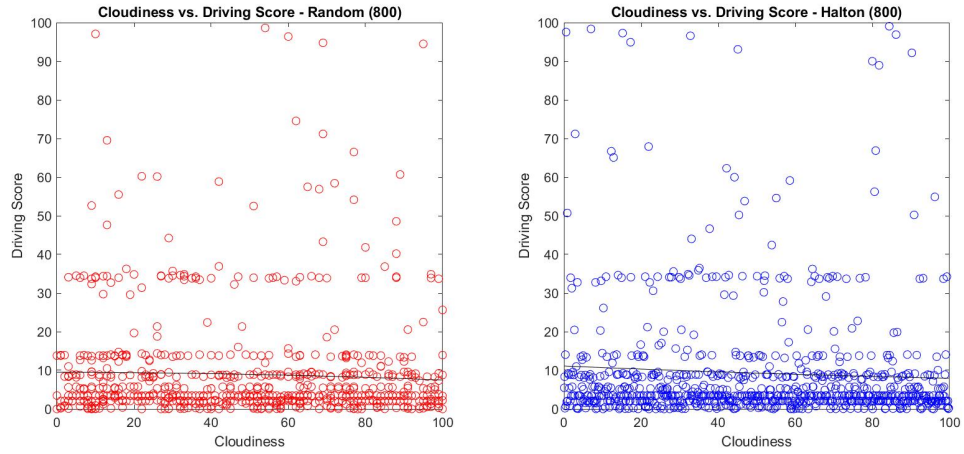


Figure 31: Cloudiness vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

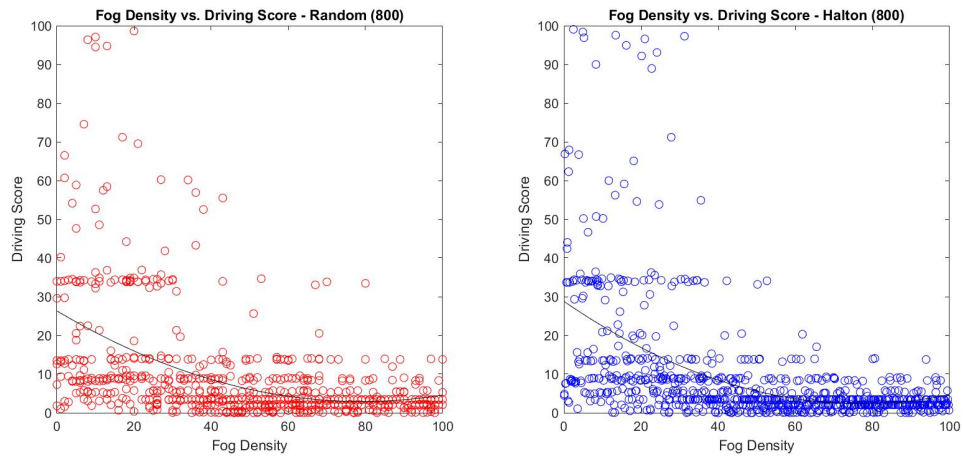


Figure 32: Fog density vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

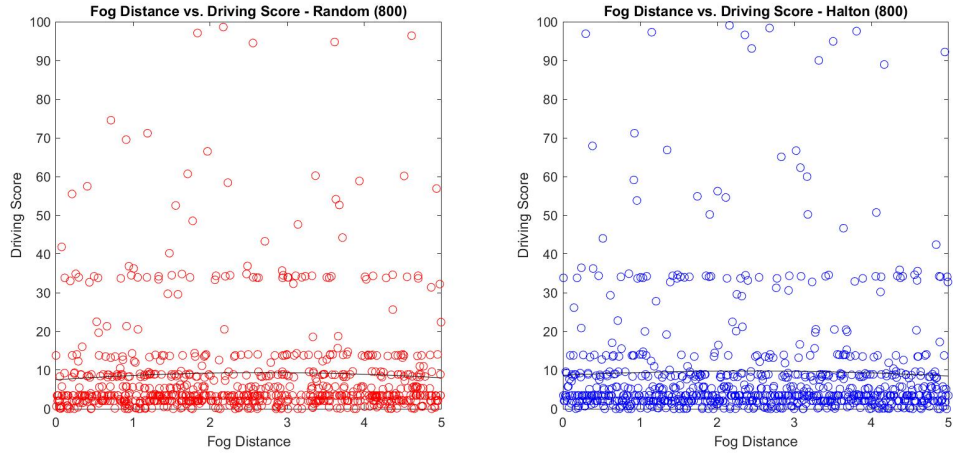


Figure 33: Fog distance vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

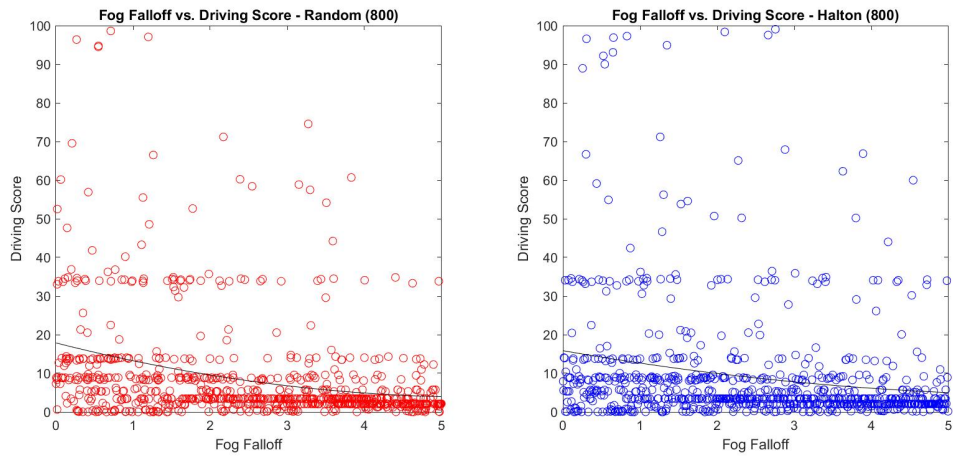


Figure 34: Fog Falloff vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

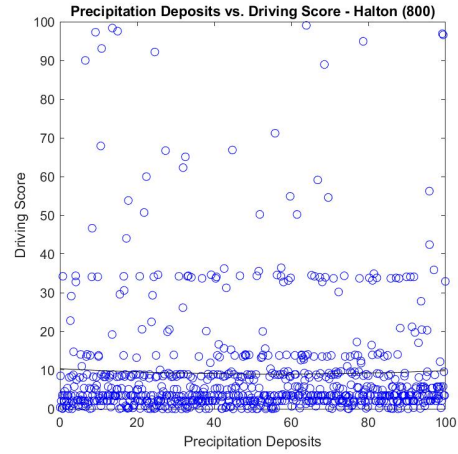
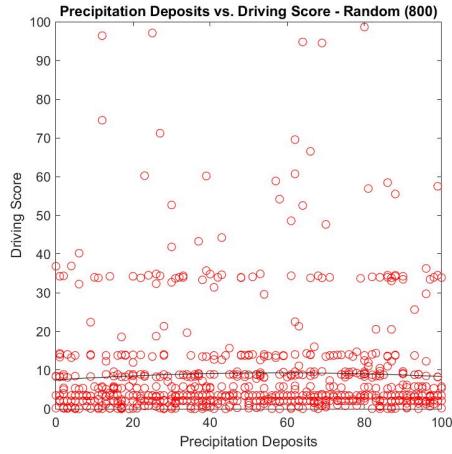


Figure 35: Precipitation deposits vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

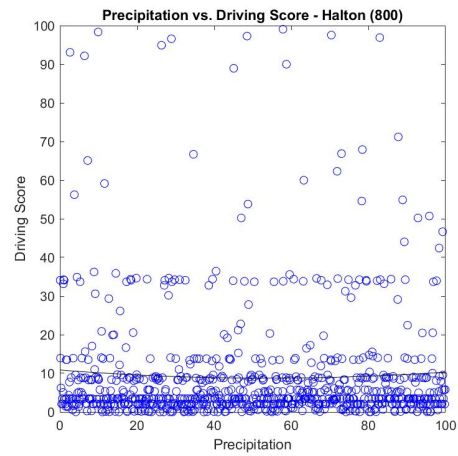
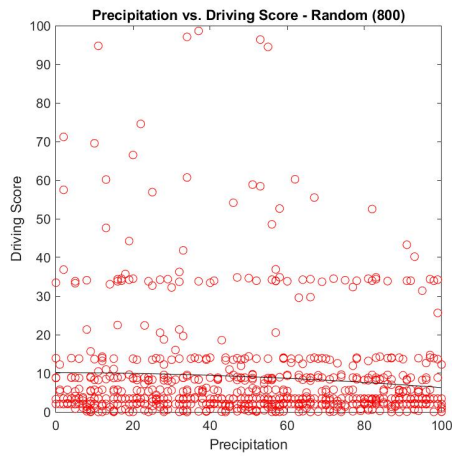


Figure 36: Precipitation vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

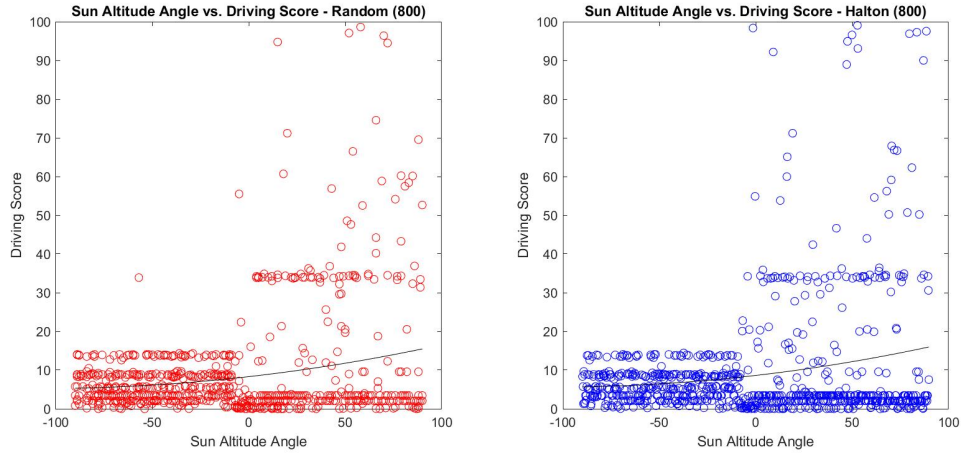


Figure 37: Sun altitude angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

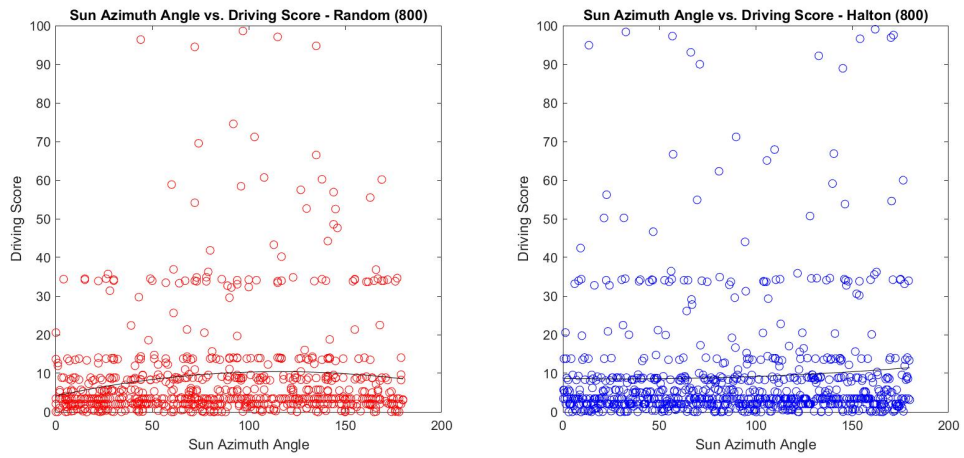


Figure 38: Sun azimuth angle vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

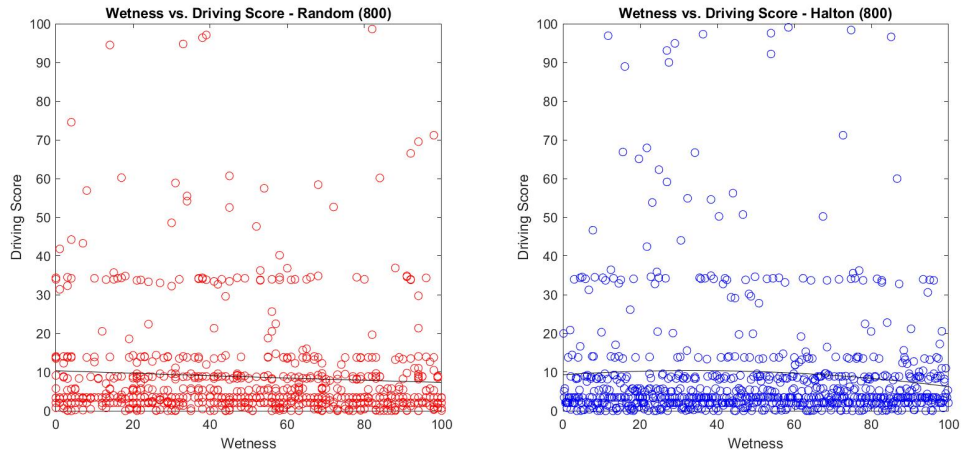


Figure 39: Wetness vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

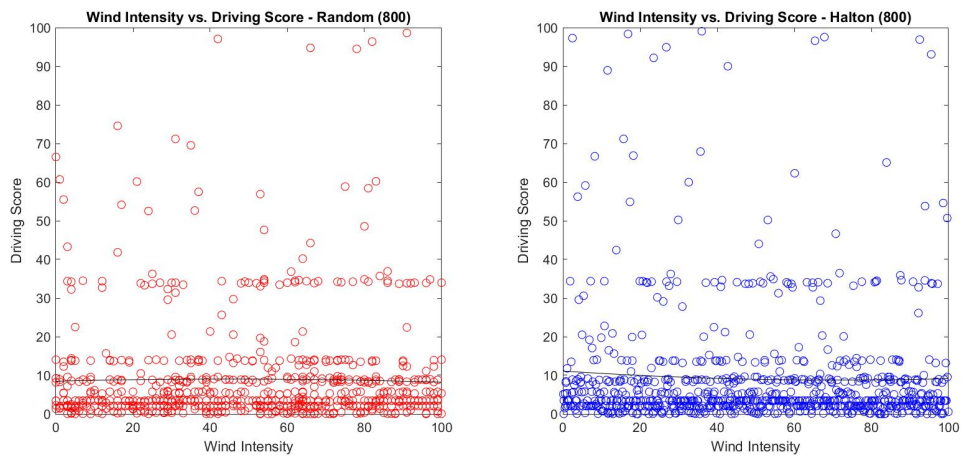


Figure 40: Wind Intensity vs driving score with trendline for pseudo-random scenario set (left) and Halton scenario set (right) for 800 scenarios

Appendix B

Table 7.1

Results of the same set of 50 scenarios run 4 separate times

Scenario ID	Run 1	Run 2	Run 3	Run 4	Variance
1	3.45	3.45	0.45	3.45	2.25
2	3.40	4.91	13.51	13.93	30.90
3	3.35	4.96	2.03	3.35	1.44
4	3.45	3.45	3.45	3.45	0.00
5	34.77	20.86	20.76	34.63	64.31
6	13.65	13.65	13.65	8.27	7.23
7	3.45	3.45	2.07	2.07	0.63
8	9.58	6.23	6.23	9.58	3.75
9	3.45	3.45	3.45	3.45	0.00
10	3.45	1.23	1.22	2.22	1.11
11	13.80	13.93	8.88	13.93	6.27
12	3.45	2.07	3.45	2.07	0.63
13	2.07	3.45	3.45	3.45	0.48
14	12.59	12.74	12.59	12.74	0.01
15	3.74	7.22	2.32	4.33	4.23
16	2.07	2.07	2.07	2.07	0.00
17	8.95	8.82	8.95	8.82	0.01
18	8.88	13.78	14.07	8.88	8.49
19	3.45	3.45	3.45	3.45	0.00
20	3.45	3.45	3.45	3.45	0.00
21	0.41	3.39	3.39	1.89	2.03
22	2.07	3.42	3.45	1.24	1.17
23	64.99	40.30	63.75	59.65	131.70
24	3.45	3.45	3.45	3.45	0.00
25	5.06	5.70	3.42	3.42	1.35
26	8.78	8.88	8.88	8.88	0.00
27	3.45	3.45	3.45	3.45	0.00
28	2.07	2.07	2.07	2.07	0.00
29	18.31	28.16	28.16	15.60	43.10
30	3.45	3.45	3.45	3.45	0.00
31	1.34	3.42	2.07	5.75	3.75
32	13.91	8.27	8.35	13.91	10.48
33	0.14	2.53	0.72	0.05	1.33
34	2.07	2.07	2.07	2.07	0.00
35	13.66	13.36	13.66	13.51	0.02
36	5.43	5.43	4.96	4.96	0.07

37	3.45	3.45	3.45	3.45	0.00
38	3.45	3.45	3.45	3.45	0.00
39	5.02	3.06	3.38	5.02	1.09
40	1.33	1.24	1.89	0.14	0.54
41	29.71	29.71	29.71	16.91	40.93
42	3.45	3.45	3.45	3.45	0.00
43	3.45	1.24	5.16	3.45	2.58
44	89.40	2.93	33.32	2.95	1661.67
45	3.45	3.45	3.45	3.45	0.00
46	3.45	2.07	2.25	5.16	2.03
47	8.88	8.88	8.88	8.88	0.00
48	2.07	2.07	3.45	3.45	0.63
49	3.45	3.45	0.08	0.61	3.25
50	8.82	8.82	8.82	8.82	0.00