

# **LiDAR-Driven Calibration of Microscopic Traffic Simulation for Balancing Operational Efficiency and Prediction of Traffic Conflicts**

by

Natalie Hany Farag

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Applied Science

in

Civil Engineering

Waterloo, Ontario, Canada, 2024

© Natalie Hany Farag 2024

## **Author's Declaration**

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## **Abstract**

Microscopic traffic simulation is a proactive tool for road safety assessment, offering an alternative to traditional crash data analysis. Microsimulation models, such as PTV VISSIM, replicate traffic scenarios and conflicts under various conditions, thereby aiding in the assessment of driving behavior and traffic management strategies. When integrated with tools like the Surrogate Safety Assessment Model (SSAM), these models estimate potential conflicts. Research often focuses on calibrating these models based on traffic operation metrics, such as speed and travel time, while neglecting safety performance parameters.

This thesis investigates the effects of calibrating microsimulation models for both operational metrics including travel time and speed, and safety metrics including traffic conflicts and Post-Encroachment Time (PET) distribution, using LiDAR sensor data. The calibration process involves three phases: performance calibration, performance and safety calibration, and only-safety calibration. The results show that incorporating safety-focused parameters enhances the model's ability to replicate observed conflict patterns. The study highlights the trade-offs between operational efficiency and safety, with adjustments to parameters like standstill distance improving safety outcomes without significantly compromising operational metrics. Furthermore, there is a substantial difference in the calibrated minimum distance headway for the safety model, highlighting the trade-off between operational efficiency and safety. While the operational calibration focuses on optimizing flow, the safety calibration prioritizes realistic conflict simulation, even at the cost of reduced flow efficiency. The research emphasizes the importance of accurately simulating real-world driver behavior through adjustments to parameters like the probability and duration of temporary lack of attention.

## **Acknowledgements**

I want to express my utmost gratitude to Dr. Chris Bachmann and Dr. Liping Fu, whose scholar guidance made this moment feasible. Their mentorship assisted me with thinking more constructively at every research stage.

Thank you, Mickey Dang, for building the LiDAR sensors extraction code, and Farid Mullally for building the Google Distance Matrix API code. Your contributions made my transition to graduate school easier.

My cornerstone, my family, thank you for your constant prayers. No words can describe my feelings except for a big hug. Thank you, Gedo (Grandpa) Saeed, for being my first cheerleader.

Lastly, thank you God.

## Table of Contents

|                                                           |     |
|-----------------------------------------------------------|-----|
| Author's Declaration.....                                 | ii  |
| Abstract.....                                             | iii |
| Acknowledgements.....                                     | iv  |
| List of Figures.....                                      | ix  |
| List of Tables .....                                      | xi  |
| 1 Introduction.....                                       | 1   |
| 1.1 Background.....                                       | 1   |
| 1.2 Problem Statement.....                                | 2   |
| 1.3 Research Objective.....                               | 3   |
| 1.4 Significance and Contribution.....                    | 4   |
| 1.5 Research Scope.....                                   | 4   |
| 1.6 Structure of Thesis.....                              | 5   |
| 2 Literature Review.....                                  | 6   |
| 2.1 Traditional Road Safety Analysis and Management ..... | 6   |
| 2.2 Current Road Safety Techniques.....                   | 7   |
| 2.2.1 Crash Data-Based Technique.....                     | 7   |
| 2.2.2 Crash Model-Based Technique .....                   | 9   |
| 2.3 Traffic Conflict-based Technique.....                 | 12  |
| 2.3.1 Conflict-Based Technique Evolution.....             | 12  |
| 2.3.2 Conflict Data Collection Methods.....               | 15  |
| 2.4 Surrogate Safety Measures .....                       | 18  |
| 2.4.1 Time-to-Collision.....                              | 18  |
| 2.4.2 Post-Encroachment Time .....                        | 19  |
| 2.4.3 Gap Time.....                                       | 20  |
| 2.4.4 Time Headway .....                                  | 20  |

|       |                                                          |    |
|-------|----------------------------------------------------------|----|
| 2.4.5 | Deceleration Rate to Avoid a Crash.....                  | 21 |
| 2.4.6 | Proportion of Stopping Sight Distance.....               | 21 |
| 2.5   | Simulation-Based Conflict Technique.....                 | 23 |
| 2.5.1 | Methods for Calibrating Simulation Models.....           | 27 |
| 2.5.2 | Validation Methods.....                                  | 31 |
| 2.6   | Research Gaps .....                                      | 33 |
| 3     | Methodology .....                                        | 34 |
| 3.1   | Data Collection and Network Building .....               | 36 |
| 3.1.1 | Field Data .....                                         | 36 |
| 3.1.2 | Network Building.....                                    | 36 |
| 3.2   | Performance Model Calibration .....                      | 38 |
| 3.2.1 | Driving Behavior.....                                    | 39 |
| 3.2.2 | Vehicle Performance.....                                 | 43 |
| 3.3   | Performance Model Validation .....                       | 45 |
| 3.4   | Parameter Selection .....                                | 46 |
| 3.4.1 | Effective Parameters by Literature Review.....           | 46 |
| 3.4.2 | Sensitivity Analysis.....                                | 47 |
| 3.5   | Performance and Safety Model Calibration .....           | 49 |
| 3.5.1 | Standstill Distance.....                                 | 50 |
| 3.5.2 | Minimum Headway.....                                     | 50 |
| 3.5.3 | Look-Ahead Distance (Maximum/Minimum) .....              | 51 |
| 3.5.4 | Temporary Lack of Attention (Duration/Probability) ..... | 51 |
| 3.5.5 | Desired Position at Free Flow .....                      | 52 |
| 3.6   | Only Safety Model Calibration.....                       | 52 |
| 3.7   | Conflicts Validation.....                                | 52 |
| 3.8   | Effectiveness of Calibration .....                       | 53 |

|       |                                                         |    |
|-------|---------------------------------------------------------|----|
| 3.8.1 | Error Degree.....                                       | 53 |
| 3.8.2 | Effectiveness Rate.....                                 | 53 |
| 3.8.3 | Overall Assessment.....                                 | 54 |
| 4     | Results.....                                            | 55 |
| 4.1   | Field Data .....                                        | 55 |
| 4.2   | Performance Model Calibration .....                     | 58 |
| 4.2.1 | Driving Behavior.....                                   | 59 |
| 4.2.2 | Vehicle Performance.....                                | 63 |
| 4.3   | Performance Model Validation .....                      | 71 |
| 4.4   | Parameter Selection.....                                | 75 |
| 4.5   | Performance and Safety Model Calibration .....          | 76 |
| 4.5.1 | Standstill Distance.....                                | 77 |
| 4.5.2 | Minimum Distance Headway.....                           | 77 |
| 4.5.3 | Look-Ahead Distance (Maximum) .....                     | 78 |
| 4.5.4 | Temporary Lack of Attention (Duration/Probability)..... | 78 |
| 4.5.5 | Desired Position at Free Flow .....                     | 78 |
| 4.6   | Conflicts Validation.....                               | 78 |
| 4.6.1 | Distribution of Post-Encroachment Time (PET).....       | 78 |
| 4.6.2 | Distribution of Conflict Types .....                    | 83 |
| 4.6.3 | Spatial Distribution of Conflicts .....                 | 84 |
| 4.7   | Effectiveness of Calibration .....                      | 85 |
| 4.7.1 | Error Degree.....                                       | 85 |
| 4.7.2 | Effectiveness Rate.....                                 | 86 |
| 4.7.3 | Overall Assessment.....                                 | 90 |
| 4.8   | Discussion.....                                         | 92 |
| 4.9   | Limitations.....                                        | 93 |

|       |                                                                                            |     |
|-------|--------------------------------------------------------------------------------------------|-----|
| 4.9.1 | Constraints of LiDAR Data.....                                                             | 93  |
| 4.9.2 | Microsimulation Limitations.....                                                           | 93  |
| 5     | Conclusion .....                                                                           | 95  |
| 5.1   | Contributions and Key Findings.....                                                        | 95  |
|       | References.....                                                                            | 97  |
|       | Appendix A – LiDAR Data Pipeline Python Code.....                                          | 118 |
|       | Appendix B – LiDAR Checkpoint Data Distribution Python Code .....                          | 128 |
|       | Appendix C – Minimum Desired Speed Python Code .....                                       | 140 |
|       | Appendix D – Maximum Desired Speed and Desired Acceleration/Deceleration Python Code ..... | 151 |
|       | Appendix E – VISSIM Travel Time Compiler Python Code .....                                 | 160 |
|       | Appendix F – VISSIM Speed Compiler Python Code.....                                        | 170 |
|       | Appendix G – SSAM Conflicts Coordinates Conversion and Cleanup Python Code .....           | 180 |
|       | Appendix H – VISSIM COM Python Code.....                                                   | 188 |

## List of Figures

|                                                                                                                 |    |
|-----------------------------------------------------------------------------------------------------------------|----|
| Figure 1 Highway Safety Manual Roadway Safety Management Process (Highway Safety Manual: User Guide, 2022)..... | 7  |
| Figure 2 Road Safety Pyramid (Hyden, 1987).....                                                                 | 13 |
| Figure 3 Schematic Diagram of TTC Calculation (Zheng et al., 2024).....                                         | 19 |
| Figure 4 Schematic Diagram of PET Calculation (Zheng et al., 2024).....                                         | 20 |
| Figure 5 SSAM Methodology Flowchart .....                                                                       | 23 |
| Figure 6 Area according to the final reached position (Park et al., 2019) .....                                 | 29 |
| Figure 7 Initial Golden-Section Search Direction (Park et al., 2019).....                                       | 30 |
| Figure 8 Final Golden-Section Search Direction (Park et al., 2019).....                                         | 30 |
| Figure 9 Artificial Neural Network Representation.....                                                          | 30 |
| Figure 10 Thesis Framework .....                                                                                | 35 |
| Figure 11 VISSIM Road Network Layout.....                                                                       | 36 |
| Figure 12 VISSIM Parameters.....                                                                                | 39 |
| Figure 13 Car-Following Model .....                                                                             | 40 |
| Figure 14 VISSIM Default Desired Speed Distribution for 50 km/h .....                                           | 43 |
| Figure 15 Boulevard du Saint Maurice Aerial Image.....                                                          | 55 |
| Figure 16 LiDAR Locates along Boulevard du Saint Maurice.....                                                   | 57 |
| Figure 17 AM Peak Vehicle Speed Distribution .....                                                              | 58 |
| Figure 18 PM Peak Vehicle Speed Distribution .....                                                              | 58 |
| Figure 19 AM Peak Acceleration/Deceleration Distribution.....                                                   | 58 |
| Figure 20 PM Peak Acceleration/Deceleration Distribution .....                                                  | 58 |
| Figure 21 AM Peak Time Headway Distribution .....                                                               | 58 |
| Figure 22 PM Peak Time Headway Distribution.....                                                                | 58 |
| Figure 23 AM Peak Standstill Distance Distribution.....                                                         | 59 |
| Figure 24 PM Peak Standstill Distance Distribution .....                                                        | 59 |
| Figure 25 AM Peak Lane Changing Deceleration Distribution.....                                                  | 61 |
| Figure 26 AM Peak Following Deceleration Distribution.....                                                      | 61 |
| Figure 27 PM Peak Lane Changing Deceleration Distribution .....                                                 | 61 |
| Figure 28 PM Peak Following Deceleration Distribution .....                                                     | 61 |
| Figure 29 AM Peak Minimum Distance Headway Distribution.....                                                    | 62 |
| Figure 30 PM Peak Minimum Distance Headway Distribution .....                                                   | 62 |
| Figure 31 AM Peak Safety Reduction Factor Distribution.....                                                     | 62 |
| Figure 32 PM Peak Safety Reduction Factor Distribution .....                                                    | 62 |
| Figure 33 Northbound AM Peak MANE Comparison for Different Speed Thresholds...                                  | 64 |
| Figure 34 Southbound AM Peak MANE Comparison for Different Speed Thresholds...                                  | 64 |
| Figure 35 Northbound PM Peak MANE Comparison for Different Speed Thresholds ...                                 | 65 |
| Figure 36 Southbound PM Peak MANE Comparison for Different Speed Thresholds ...                                 | 65 |
| Figure 37 Northbound AM Peak Desired Speed Cumulative Probability .....                                         | 66 |
| Figure 38 Southbound AM Peak Desired Speed Cumulative Probability .....                                         | 66 |

|                                                                                      |    |
|--------------------------------------------------------------------------------------|----|
| Figure 39 Northbound PM Peak Desired Speed Cumulative Probability.....               | 67 |
| Figure 40 Southbound PM Peak Desired Speed Cumulative Probability.....               | 67 |
| Figure 41 LiDAR Insight Platform Speed Candlestick Chart .....                       | 68 |
| Figure 42 Calibrated Desired Acceleration from LiDAR for AM Peak .....               | 69 |
| Figure 43 Calibrated Desired Deceleration from LiDAR for AM Peak .....               | 70 |
| Figure 44 Calibrated Desired Acceleration from LiDAR for PM Peak.....                | 70 |
| Figure 45 Calibrated Desired Deceleration from LiDAR for PM Peak.....                | 71 |
| Figure 46 AM Peak LiDAR API Observed Travel Time Distribution.....                   | 72 |
| Figure 47 AM Peak Google Matrix API Observed Travel Time Distribution .....          | 72 |
| Figure 48 AM Peak Simulated Travel Time Distribution for Performance Calibration ... | 72 |
| Figure 49 AM Peak Simulated Travel Time Distribution for Default Model .....         | 72 |
| Figure 50 PM Peak LiDAR API Observed Travel Time Distribution .....                  | 73 |
| Figure 51 PM Peak Google Matrix API Observed Travel Time Distribution.....           | 73 |
| Figure 52 PM Peak Simulated Travel Time Distribution for Performance Calibration ... | 73 |
| Figure 53 PM Peak Simulated Travel Time Distribution for Default Model .....         | 73 |
| Figure 54 AM Peak LiDAR API Observed Speed Distribution .....                        | 74 |
| Figure 55 AM Peak Google Matrix API Observed Speed Distribution.....                 | 74 |
| Figure 56 AM Peak Simulated Speed Distribution for Performance Calibration.....      | 74 |
| Figure 57 AM Peak Simulated Speed Distribution for Default Model.....                | 74 |
| Figure 58 PM Peak LiDAR API Observed Speed Distribution.....                         | 75 |
| Figure 59 PM Peak Google Matrix API Observed Speed Distribution .....                | 75 |
| Figure 60 PM Peak Simulated Speed Distribution for Performance Calibration .....     | 75 |
| Figure 61 PM Peak Simulated Speed Distribution for Default Model .....               | 75 |
| Figure 62 Combined Conflicts PET Distribution per Intersection .....                 | 80 |
| Figure 63 Conflict Type PET Distribution per Intersection.....                       | 81 |
| Figure 64 Conflict Type Distribution per Intersection.....                           | 84 |
| Figure 65 Observed Conflicts Heat Map .....                                          | 85 |
| Figure 66 Default Conflicts Heat Map.....                                            | 85 |
| Figure 67 Performance Calibrated Conflicts Heat Map .....                            | 85 |
| Figure 68 Performance and Safety Calibrated Conflicts Heat Map .....                 | 85 |
| Figure 69 Only Safety Calibrated Conflicts Heat Map.....                             | 85 |

## List of Tables

|                                                                                        |    |
|----------------------------------------------------------------------------------------|----|
| Table 1 Drawbacks of Conflict Indicators .....                                         | 22 |
| Table 2 Previous Simulation-Based Technique Studies.....                               | 25 |
| Table 3 Checklist of Calibrated Parameters by each Model Stage .....                   | 38 |
| Table 4 Percentile Combinations for Desired Acceleration/Deceleration .....            | 44 |
| Table 5 Cut-off Points for Desired Acceleration/Deceleration Calculation.....          | 45 |
| Table 6 Summary of Conflicts-Effective Parameters based on Literature.....             | 47 |
| Table 7 Sensitivity Analysis Tested Ranges.....                                        | 48 |
| Table 8 Sensitivity Analysis Parameters Combinations.....                              | 49 |
| Table 9 LiDAR Sensor Data Description.....                                             | 56 |
| Table 10 Maximum and Accepted Deceleration of Lane Changing Model.....                 | 61 |
| Table 11 Desired Acceleration/Deceleration MANE Results at Different Percentiles ..... | 69 |
| Table 12 Parameter Selection Results.....                                              | 76 |
| Table 13 Performance and Safety Calibrated Model Parametric Values .....               | 77 |
| Table 14 Operational MANE Results .....                                                | 86 |
| Table 15 AM Peak Parameter Effectiveness Rate.....                                     | 87 |
| Table 16 PM Peak Parameter Effectiveness Rate .....                                    | 88 |
| Table 17 AM Peak Model Effectiveness Rate .....                                        | 90 |
| Table 18 PM Peak Model Effectiveness Rate.....                                         | 90 |
| Table 19 Modelled Corridor Results Summary .....                                       | 91 |
| Table 20 Comparison of Conflict Ratios Summary .....                                   | 93 |

# 1 Introduction

## 1.1 Background

In 2022, approximately 118,853 individuals in Canada sustained injuries in road traffic incidents (Canada, 2024), resulting in financial strain totaling \$36 billion according to the latest published Canadian vehicle collisions statistics. These costs were divided into the following categories: 3.18 percent attributed to traffic delays, 37.58 percent to injuries, and 44.01 percent due to fatalities (Canada, 2023). In response, governmental agencies worldwide have been making significant investment into various road safety improvement programs and projects. Initiatives such as Vision Zero aim to eliminate traffic-related fatalities and serious injuries through the implementation of various safety measures, including infrastructure improvements, public awareness campaigns, and policy interventions. Essential to these efforts are Roadway Safety Management (RSM) programs, which provide a structured framework for preparing and incorporating policies, preventive strategies and new safety measures necessary to produce desired road safety performance (Alfonsi et al., 2016; Bliss & Breen, 2009; Muhlrads, 2009).

RSM programs are the overall process of identifying sites with potential for safety improvement, diagnosing conditions at the site, evaluating conditions, and identifying potential treatments (HSM, 2022). This includes prioritizing and programming treatments and subsequently evaluating the effectiveness of these treatments in reducing crashes. Safety funds can then be used as efficiently as possible based on the identified locations.

In the realm of road safety assessment, three distinct methods have emerged. First, the traditional crash-based technique, which relies on historical crash data to quantify safety using crash frequency and severity. It can be used in conjunction with additional information by surveying collision type, driver behavior, and environmental conditions. Statistical techniques build on this foundation by utilizing advanced models to predict crash likelihood, by considering roadway characteristics and addressing the biases inherent in short-term crash data fluctuations (Persaud et al., 2001; Wang, 1994). In addition to improving the accuracy of crash predictions, statistical methods are also integral to other safety evaluation techniques. Second, the conflict-based technique, which is a proactive approach, involving the measurements of *‘an observable situation in which two or more road users approach each other in time and space to such extent that there is a risk of collision if their movement remains unchanged’* (Amundsen & Hyden, 1977). This technique is used for identifying and analysing conflict events involving braking or evasive maneuvers and generating Surrogate Safety Measures (SSMs) (Hyden, 1987; Parker Jr. & Zegeer, 1989). Third, the simulation-based technique, which utilizes traffic simulation software to model the movements and interactions of road users which are then used to compute surrogates of different conflict types (Gettman & Head (2003). Generally, trajectories of the road users exported from a simulation model are post-processed to identify the conflicts of interest and compute the relevant surrogate safety measures. One

common post-processor is the Surrogate Safety Assessment Model (SSAM), which was developed by Federal Highway Administration (FHWA) (Gettman et al., 2008) and includes the functionality of visualizing different conflict types through heat maps.

Every method used to assess roadway safety has limitations that must be acknowledged. For instance, the crash-based technique, while reactive in nature, demands extensive data to identify true unsafe locations and estimate the effectiveness of implemented safety countermeasures (Cui et al., 2022; Da Costa et al., 2015). Being data intensive, this technique is best-suited for facilities with established traffic surveillance systems (Abdel-Aty & Pande, 2007). The statistical-based technique addresses the regression-to-mean bias present in the crash-based method (Paliotto et al., 2024); however, it faces challenges in balancing predictive accuracy with uncovering underlying causality, necessitating careful consideration in model development and interpretation (Mannering et al., 2020). Furthermore, the presence of unobserved heterogeneity can lead to biased and inconsistent parameter estimates in statistical-based analysis, requiring more sophisticated approaches (Mannering et al., 2016). The conflict-based technique requires extensive field observations or the use of camera vision or other technologies to capture conflict events (Zheng et al., 2014). It also heavily relies on the experience of trained observers or tracking algorithms to identify conflicts, given the various evasive actions and the lack of standardized predetermined SSM thresholds used in classifying conflicts and severity (Autey et al., 2012; Lu et al., 2021). Lastly, simulation-based techniques depend on the accuracy of the simulation model, which may not fully represent real-world conditions due to simplifications in behavioral models and the intensive calibration process (Lu, 2007). Microsimulation models may oversimplify vehicle movements and fail to accurately capture the complexity of collisions. This thesis research focuses on simulation-based road safety analysis, with a particular attention to challenges related to model calibration and validation.

## **1.2 Problem Statement**

The incorporation of traffic simulation models into road safety analysis offers some potential advantages compared to traditional approaches. Unlike traditional approaches that often rely on historical crashes or observed traffic conflicts, which may not capture all critical safety events or are limited to the existing road network, traffic and environmental conditions and safety countermeasures, traffic simulation models have the potential to simulate critical traffic situations and near-miss incidents under a much wider range of conditions (Siebke et al., 2022). Additionally, simulation models allow for the evaluation of safety impacts associated with alternative designs and traffic management strategies before their implementation (Astarita et al., 2011). The detailed modeling of vehicle interactions and driver behaviors in a simulation model can lead to a more comprehensive understanding of contributing factors to safety issues.

Recent research has increasingly utilized microsimulation models, such as PTV VISSIM, to analyze system performance and road safety. These models provide representations of vehicle interactions and can simulate various traffic scenarios to predict potential conflicts and crashes. Cafiso et al. (2018) discussed the benefits of integrating microsimulation with the SSAM to enhance safety predictions. SSAM is particularly effective in estimating traffic conflicts, which serve as proxies for potential crashes. Studies by Essa & Sayed (2015, 2020) showed that simulated traffic conflicts can accurately reflect real-world conditions, highlighting the usefulness of these models in safety analysis. Dadashzadeh et al. (2019) emphasized the importance of calibrating driver behavior parameters to enhance the accuracy of safety predictions, especially in complex traffic scenarios. Similarly, Liang et al. (2020) stressed the sensitivity of simulated conflicts to changes in driver behavior parameters, emphasizing the need for precise understanding and calibration of these behaviors.

Nevertheless, the application of traffic simulation models presents challenges. Accurately capturing realistic driver tendencies is crucial, yet a significant challenge due to the complexity and stochastic nature of human driving behavior (Yang, 2012a). Consequently, calibration and validation against observed data are imperative and further research is needed to validate and enhance the accuracy of these simulation models in representing real-world conditions. For example, microsimulation models can be inadequate for safety analysis as the SSMs incorporated may be limited to capturing only certain types of conflicts. Additionally, these models often systematically avoid simulating realistic vehicle interactions that could lead to collisions. Lastly, there is an ongoing debate and lack of consensus on the most appropriate SSMs to use in simulation models and how to interpret their thresholds concerning safety risks.

Despite the increasing use of traffic simulation models for determining safety performance, existing efforts on calibration of simulation models primarily focus on fine-tuning simulation models based on traffic performance metrics, overlooking safety performance parameters and traffic conflicts (Al-Ghandour et al., 2011; Fan et al., 2013a; Li et al., 2016). This highlights the urgent need for further exploration into developing comprehensive metrics to assess the safety implications of calibration decisions for simulation models.

### **1.3 Research Objective**

The goal of this thesis research is to investigate the challenges of applying a simulation model for road safety analysis with a specific focus on issues related to model calibration. The specific objectives of the research include:

1. Synthesizing literature on the simulation-based technique.

2. Exploring the critical surrogate safety measures that could be derived from a simulation model and identify the key model parameters that affect these safety performance metrics.
3. Assessing the impact of safety-focused calibration on overall network operational performance by examining the consistency and potential trade-offs between different performance measures, such as travel time, speed, and post-encroachment time (PET).
4. Developing a systematic procedure and guidelines to incorporate both safety and operations related metrics into the calibration process of traffic simulation models. The thesis will include a systematic comparison of the outcomes of safety-focused calibration with traditional operations-focused calibration methods.

#### **1.4 Significance and Contribution**

The thesis focuses on improving the accuracy of traffic simulation models for assessing road safety. Unlike traditional methods that rely only on historical crash data, this research employs traffic simulation models to create a more dynamic and comprehensive approach to safety evaluation.

The main objective is to enhance the accuracy of traffic simulation models such as VISSIM by integrating historical data from LiDAR sensors. This integration ensures that the models more accurately represent travel time, speed, and potential conflict scenarios, providing transportation planners and engineers with more reliable tools for predicting and mitigating safety issues. A unique aspect of this thesis is the inclusion of high-resolution trajectory data from LiDAR sensors in the calibration process, allowing for a more detailed analysis of vehicle interactions and driver behaviors crucial for understanding and improving traffic safety.

The thesis aims to contribute to building a systematic comparison of safety-focused calibration outcomes with traditional operations-focused calibration methods. This comparison highlights the benefits and challenges of integrating safety metrics into traffic simulation models, providing valuable guidance for future studies and practical applications. The findings can inform the development of guidelines and best practices for transportation agencies, enabling them to make data-driven decisions that enhance road safety.

#### **1.5 Research Scope**

The focus of this thesis is on calibrating a microscopic simulation model to enhance its ability to predict and evaluate conflicts using historical data collected by LiDAR sensors installed along an urbanized collector section on Boulevard du Saint Maurice in Trois-Rivières, Québec. The study area includes a corridor of six signalized intersections located within an urban mixed commercial-residential environment. The research involves

temporal analysis, including the AM peak and PM peak periods, to gain a comprehensive understanding of traffic dynamics. Additionally, various crossing and longitudinal conflict types for vehicular road users are being analyzed to gain insights into safety concerns across different scenarios.

The thesis contributes a two-stage calibration process that optimizes microsimulation model parameters using LiDAR data, aiming to enhance the accuracy of simulation in predicting potential conflicts.

The LiDAR sensors primarily detect crossing conflicts and have limitations in identifying other types of conflicts. Although efforts have been made to review recorded videos and categorize conflicts, the system is only able to identify conflicts with PET of less than 2 seconds, which may result in an incomplete representation of conflict types. Furthermore, while an algorithm was developed to categorize conflicts as lane changes or rear-end based on conflict geographical coordinates, there was still some uncertainty in the classification process. Therefore, to ensure consistency and reliability in the analysis, this thesis combines rear-end and lane change conflicts and categorizes observed conflicts as either crossing or longitudinal conflicts.

## **1.6 Structure of Thesis**

This thesis is structured into several chapters:

Chapter 1 serves as an introduction, outlining the thesis area and providing background information relevant to the research problem.

Chapter 2 explores the existing literature pertinent to this thesis, examining traditional and contemporary road safety assessment methods, advancements in data collection methods over time, and previous methodologies used in assessing traffic conflicts.

Chapter 3 details the methodology for calibrating a microscopic simulation model, including data collection using LiDAR sensors and the development of a VISSIM network model. This chapter also describes the two-stage calibration process and validation steps. The calibrated model is validated by comparing simulated outputs with observed conflicts and traffic performance measurements, concluding with an assessment of the accuracy and distribution of simulated conflicts for each conflict type against PET.

Chapter 4 presents the results obtained by applying the proposed methodology. The calibrated model is validated by comparing simulated outputs with observed conflicts and traffic performance measurements, concluding with an assessment of the accuracy and distribution of simulated conflicts for each conflict type using PET.

Chapter 5 summarizes the key findings and conclusions.

## 2 Literature Review

The topics reviewed in this literature review focus on techniques to evaluate roadway safety. Historically, roadway safety assessments primarily relied on crash data analysis. However, approaches using crash data have limitations due to the reactive nature of using past crashes. In response, analysts have shifted interest toward proactive methods, such as predicting potential conflicts using statistical models. Simultaneously, microsimulation models (for example, VISSIM) have become vital for evaluating the operational impacts of roadway designs. Over the past few decades, there has been a growing interest in leveraging microsimulation models for predicting traffic conflicts through the SSAM (Cafiso et al., 2018), but the quality of the data used in these models significantly influences the accuracy of the roadway safety assessments (Mehdizadeh et al., 2020). This literature review focuses on the evaluation techniques for roadway safety through definitions and contributed research cases. The goal of this chapter is to provide a key summary of the presented information and present research gaps that this thesis aims to address.

### 2.1 Traditional Road Safety Analysis and Management

Roadway Safety Management (RSM) comprises a set of tasks for preparing and implementing road safety policies (Alfonsi et al., 2016). RSM is a quantitative and systematic process for managing road safety by analyzing and addressing various contributing factors to road accidents on existing roadways (*Highway Safety Manual*, 2009).

A systematic approach to RSM is significant because it provides a clear framework for addressing road-safety issues. This approach is achievable by defining the burden and nature of road casualties, gaining support from decision makers, establishing road safety policies based on studied targets and problems, and ensuring effective funding and countermeasures (Varhelyi, 2016). As illustrated in Figure 1, RSM consists of a six-step process. The process is initiated by reviewing the road network to identify risks through Network Screening. Followed by evaluating historical crash data and field conditions were through the diagnosis step to identify the crash patterns. Afterwards, factors contributing to crashes and potential countermeasures are identified and evaluated in terms of benefits and costs through the Countermeasure Selection and Economic Appraisal stages. Finally, through the last two stages of the cycle – Project Prioritization and Safety Effectiveness Evaluation – whereas multiple improvement projects with justified finances are evaluated based on objectives such as cost, mobility, and environmental impacts, followed by each site being evaluated by the effectiveness of the countermeasures.

The complexity of the RSM system, combined with the limited evidence on the relationship between its components and road safety outcomes (Alfonsi et al., 2016; Dupont et al., 2011), necessitates the use of reliable safety analysis techniques. The techniques are essential for quantifying the associated risks with transportation facilities

and for evaluating the effectiveness of safety measures – such as policies, programs, and projects – in reducing crashes and injuries (Papadimitriou et al., 2012).

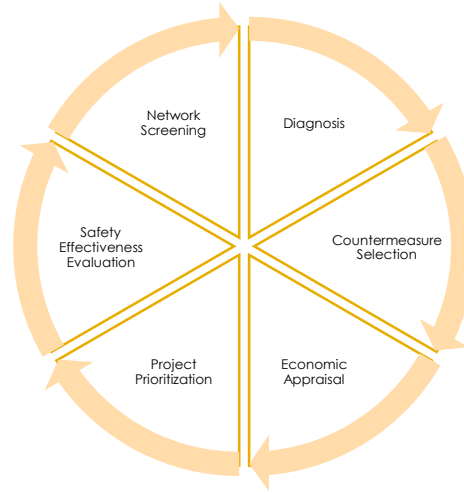


**Figure 1 Highway Safety Manual Roadway Safety Management Process (Highway Safety Manual: User Guide, 2022)**

## 2.2 Current Road Safety Techniques

### 2.2.1 Crash Data-Based Technique

The use of historical crashes for road safety analysis is the traditional core of traffic safety evaluation. It relies on statistical analysis of historical crash data to examine the safety and performance of a roadway facility and quantifies the effects based on crash frequency or severity (Mahmud et al., 2016; St-Aubin, 2011).

In this crash-based approach, traffic safety is measured by the observed crash frequency, which is the basic number of recorded crashes in a period of preference for a specific site location (*Highway Safety Manual*, 2009). The site characterized by a crash rate above a set threshold, as per Equation 2.1, is used to identify sites potentially requiring countermeasures. Typically, before-and-after evaluations are employed in observational studies (Elvik, 2002).

$$\text{Crash Rate} = \frac{\text{Average Crash Frequency in a Period}}{\text{Exposure in Same Period}} \quad (2.1)$$

The observed crash analysis provides valuable insight into crashes on roadways. However, depending solely on such a technique, it may not provide a complete understanding of the underlying causes (*Highway Safety Manual*, 2009). When used in conjunction with other information, such as annual crash comparison, collision type, collision severity, surveying driver behavior, and environmental conditions, the analyst can identify patterns that may have been unapparent through a single approach. Notably, the efforts towards a complete picture of crashes have caused engineers to address effective strategies for road safety.

For instance, studies have suggested that geometric road design accuracy, traffic calming measures, and pavement characteristics are road elements that can affect road safety through gathered crash data analysis (Lee et al., 2006; Montoya-Alcaraz et al., 2020). Lee et al. (2006) quantified the safety benefits of ramp metering using a log-linear crash prediction model, and the results demonstrated a 5 percent to 37 percent crash reduction compared to the no-control base. Montoya-Alcaraz et al. (2020) obtained historical crash data consisting of location, type of vehicle, and type of accident, followed by processing the captured information in a Geographic Information System (GIS) to identify highway sections with the highest degree of accidents. In this case, crash-based analysis aided in justifying the importance of signs, safety devices, and preventive markings in dangerous curves, as well as more speed radar technology and the implantation of safety barriers.

However, the reliance of the crash-based technique on historical crash data as the primary source to conduct road safety assessment is presumed to be unsuitable as crash statistics are frequently unavailable, uncomprehensive, biased, and require long collection cycles. The limitations of the crash-based technique are summarized as follows:

**Unavailability:** The reporting of crashes by police can vary, leading to challenges in accessing and trusting data used for safety studies. Some countries have a reporting threshold that includes only crashes involving injuries or significant property damage (Zegeer & Deen, 1978), resulting in potential under-registration of minor crashes (Berntman, 1994). In fact, many crashes, particularly those without injuries, are often unreported. Research findings from James (1991) and Heydari et al. (2019) suggest that developed countries under-report fatalities by approximately 2 to 12 percent, while developing countries may under-report 17 to 56 percent of road crashes (Heydari et al., 2019; Periyasamy et al., 2013; Salifu & Ackaah, 2012).

**Data Quality:** Crash reports are typically completed by police officers at the scene of an accident to provide insight into the potential causes of crashes. These reports serve as documentation of crucial details, such as road conditions, involved vehicles, impacted individuals, environmental characteristics, and consequences. However, crash statistics cannot be used to assess road safety situations in most cases (Yang, 2012b), nor can they provide a complete picture of safety issues related to unsafe interactions and near-misses (Mallory et al., 2021). While crash reports include location and time to facilitate dataset connection, a significant challenge emerges due to inconsistencies in spatial and temporal attributes across multiple datasets hence difficulty to accurately analyze the information (Imprialou & Quddus, 2019). Since reporting agencies can be different organizations in which do not primarily focus on road safety, direct matching of dataset is often unprioritized (Hawas et al., 2012). For instance, crash locations might be reported using coordinates; traffic data could reference a base map; geometry data might be derived from

aerial images or a detailed base map; and weather conditions could be linked to nearby coordinates.

**Collection Cycle:** Proper identification of black spots, which are network locations with high accident occurrence, can eventually optimize treatment, resulting in economic and societal benefits (Nguyen et al., 2016). To minimize year-to-year fluctuations in the data, it is recommended to gather crash periods of three to five years to ensure data validity and reliability (Chin & Quek, 1997; Nguyen et al., 2016). Nguyen et al. (2016) suggested that the collection of traffic data for that period could identify black spots. However, throughout this period, variations in road characteristics and the surrounding environment would occur, leading to irrelevant road safety assessment and countermeasures (Abdulhafedh, 2017).

**Regression-to-the-Mean (RTM):** RTM is a statistical phenomenon that occurs when a non-random sample is selected from a population, and two variables of interest exhibit imperfect correlation. This effect reflects the tendency of extreme values observed to regress towards the population mean upon subsequent measurement (Barnett et al., 2005). In a study conducted by Griffith & Flowers (1996), six quasi-experimental designs were assessed to determine the effectiveness of highway safety programs. All experimental designs were evaluated on a before-and-after basis using a crash-based technique. Evaluations that considered naive, multiple, and yoked comparisons resulted in the inability to account for RTM and the generation of bias. This result is expected since the crash-based technique assumes a linear relationship. Although of the technique's simplicity, changing the surrounding environment and crash randomness can negatively impact the relevance of the safety assessment. Due to these defects, difficulties are encountered such as the separation of treatment effects, in addition to the transferability of a site treatment implementation to an adjacent site. As site ought to have the same geometry and environment in which is impossible to obtain. Ultimately, non-linear functions can often better characterize relationships between crash frequencies and explanatory variables (Abdulhafedh, 2017; Lord & Persaud, 2000).

### 2.2.2 Crash Model-Based Technique

Statistical Road Safety Modeling (SRSIM) initially involves creating a statistical model using historical accident data and characteristics related to various road segments, intersections, or infrastructure components (Hauer, 2004). The outcome of SRSIM is an equation that estimates the anticipated frequency of accidents while considering the traits on the other side. The objectives of SRSIM can be categorized into two aspects: A) estimating the expected accident frequency of an infrastructure element based on its traits, and B) assessing the change in the expected accident frequency resulting from a modification in an infrastructure element's trait. Although these two objectives may seem similar, they differ significantly in essence. While aspect A) is straightforward, B) presents various challenges.

Crash prediction models use regression models to identify factors contributing to crashes while improving the resulting RTM bias from the observed crash frequency in Section 2.2.1 and to estimate a reliable expected average crash frequency with altering roadway conditions (*Highway Safety Manual*, 2009). The selection of variables to be incorporated into a statistical model for road safety modeling is contingent upon circumstances and data accessibility. Nonetheless, according to Hauer's (2004) recommendations, factors associated with traffic volume, geometric design, and roadside development are deemed essential. Hauer further emphasized that the choice of variables should be influenced by both theory and empirical evidence, cautioning against the inclusion of an excessive number of variables, as it can result in overfitting. A statistical-based approach following this mathematical function is represented as the predictive model in Equation 2.2.

$$N_{\text{Predicted}} = N_{\text{SPF}} \times (AMF_{1x} \times AMF_{2x} \times \dots \times AMF_{yx}) \times C_x \quad (2.2)$$

The predictive model reflects the following basic elements:

- *Safety Performance Functions (SPFs)* are statistical base models used to estimate the average crash frequency for certain specified base conditions and are calculated by considering the Average Annual Daily Traffic Volume (AADT), length of roadway segment, and site condition in terms of geometric design and geographic information.
- *Accident Modification Factors (AMFs)* represent relative changes in crash frequency owing to site condition changes. Hence, AMF is an effectiveness-measuring ratio between expected average crash frequency resulted in condition 'a' in comparison to a condition 'b.' Therefore, AMFs serve as a tool for evaluating the countermeasures explained in Section 2.1.
- *The Calibration Factor (C)* is a calibration factor to adjust for local conditions of site, which is then multiplied by the predicted crash frequency determined by the SPF.

As the Empirical Bayes (EB) Method is a function of the predictive model with a weighted factor ( $w$ ) reflecting the adjustment of SPFs prediction varying from 0 to 1, the overall equation to calculate the expected average crash frequency for the study period is given by Equation 2.3 below:

$$N_{\text{Expected}} = w \times N_{\text{Predicted}} + (1 - w) \times N_{\text{Observed}} \quad (2.3)$$

The variables contributing to the predictive model address RTM bias, as the ultimate aspect of the method concentrates on the long-term expected average crash frequency (*Highway Safety Manual*, 2009). Consequently, the method accounts for a fundamentally non-linear relationship between crash frequency and traffic volume, which is a limitation of crash-based techniques. Furthermore, due to the nature of SPF being based

on a negative binomial distribution, the modeling is more suited for variable crash data compared to the normal distribution used in crash-based. The predictive method is also transferable. Hence, the reliance on the availability of limited crash data is reduced for a study site, as the predictive model can be used for adjacent comparable sites by modifying the necessary factors contributing to the study site.

Hauer (2004) highlighted various difficulties associated with utilizing statistical models to determine the impact of road design factors on safety. First, the challenge is to maintain consistent outcomes when the models are based on diverse datasets. Another challenge is the requirement of careful selection of variables and mathematical functions as the use of an excessive number of variables can lead to overfitting. Calibration and validation are required to effectively represent the study site, and periodic updates are necessary to account for ongoing changes in traffic volume, geometric design, and other relevant factors. In a study conducted by Huang et al. (2008) through the Bayesian hierarchical modeling approach, it was resulted 28.9 percent of unexplained variations in individual severity due to between-crash variance, in which strongly suggest the beneficence of calibration and validation as otherwise random effects between crashes would result in a biased inaccurate assessment. Finally, it is crucial to recognize that the time consumed to estimate the predicted crash frequency based on infrastructural traits can be unproblematic. However, assessing the change in crash frequency owing to the modification of infrastructure traits is fraught with difficulties.

The effectiveness of the statistical model relies heavily on the reliability of the roadside crash database. Yet the presence of significant issues such as underreporting and overall quality of crash information hinders the potential efficacy of crash statistical models (Long et al., 2018). Although the comprehensive statistical method offers the advantage of incorporating a range of factors that influence crash severity, it relies on the expertise of domain specialities and is thus susceptible to subjectivity.

Causality examines the relationship between specific factors (such as driving behavior, road design, or vehicle characteristics) and accident severity, in which is crucial for developing effective countermeasures and policies. However, establishing causality poses challenges due to two main issues. Firstly, a sufficiently large crash dataset is required to enable proper statistical modeling (Savolainen et al., 2011). Limited data often necessitates simpler models with fixed parameters, hindering detailed road safety assessment. Richer datasets allow for boarder range of variables, enhancing predictive capabilities and understanding of causality (Mannering et al., 2020). Second, endogeneity arises when explanatory variables are influenced by injury severity outcomes (Savolainen et al., 2011; Yasmin et al., 2022). For instance, individuals not wearing seatbelts may appear more frequently in severe crashes due to their association with aggressive driving (Abay et al., 2013). Similarly, airbags may act as an endogenous variable if vehicles equipped with airbags are disproportionately owned by aggressive drivers (Winston et al.,

2006). Although airbags reduce severe accident probability, this benefit may be offset by drivers adopting riskier behaviours. Such endogeneity can distort statistical estimates, leading to biased models.

## **2.3 Traffic Conflict-based Technique**

### **2.3.1 Conflict-Based Technique Evolution**

The traffic conflict-based technique was firstly proposed by General Motors in the late 1960's. Observing an intersection for traffic conflicts is a systematic method of evaluating potential accidents by tallying the observed impending accident situations that are defined as traffic conflicts (Perkins & Harris, 1968). This method uses more than 20 objective criteria for traffic conflicts to obtain comprehensive accident patterns at signalized and unsignalized intersections. By contrast, accident histories provide limited information on the causes of accidents and are often inadequate for analysis. By dynamically evaluating an intersection without waiting for an accident history to evolve, traffic conflict studies can uncover the initial causes of incidents and provide more comprehensive information than accident histories.

In 1977, Amundsen and Hyden defined the Traffic Conflict Technique (TCT), with the goal of overcoming crash-based technique limitations, affiliated RTM, and irrelevant diagnosis, as outlined in Section 2.2.1. The TCT was defined as follows (Amundsen & Hyden, 1977):

“An observable situation in which two or more road users approach each other in space and time to such an extent that there is a risk of collision if their movements remain unchanged.”

Although the definition of TCT was modified in later publications, the basic elements remain as follows:

- The collision course must be presented in conflict.
- The severity of conflict is interpreted based on the onset of an evasive action.
- Distinctions between severe and non-severe conflicts should be established. However, severe conflicts consist of frequent observable non-severe conflicts. Therefore, addressing minor interactions that lead to non-severe conflicts can potentially reduce the associated crashes and injuries.

Interaction between road users occurs at varying probabilities and degrees of severity, as illustrated in Figure 2 (Hyden, 1987). Severity is defined as the proximity to a collision and potential consequences of a crash. In this study, Hyden adapted the safety pyramid for a TCT with a broken-down traffic process into elementary events and encounters. Thus, the base of the pillar is in a stage of conflict, leading to a top peak characterized by well-defined crashes and impacts.

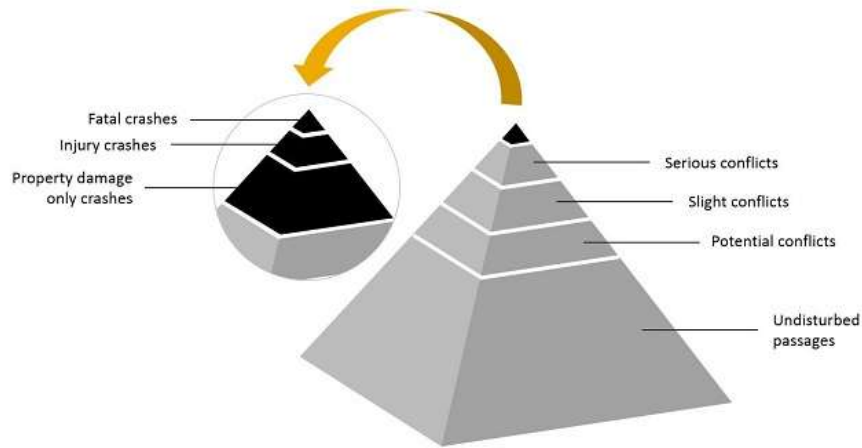


**Figure 2 Road Safety Pyramid (Hyden, 1987)**

Consequently, it was deemed necessary to establish thresholds encapsulating traffic interaction events and severity classification, resulting in the creation of the oldest Surrogate Safety Measure (SSM), known as time-to-collision (TTC). The TTC required observers to stay alert for longer time periods, as well as to be trained in a similar manner for consistent detection of conflicts and judgment of speeds and distances (Laureshyn & Várhelyi, 2018). Throughout a study of urbanized areas with speed limits of 50 km/h or lower, a TTC equaling to 1.5 seconds or less signifies the occurrence of serious conflict, as derived from video recording analyses (Hyden, 1987).

In the 2000s, the modern conflict-based technique advanced with the use of multiple SSM as discussed in Section 2.4, and improved conflict detection through automated computer image recognition software for real-time road safety assessment (Jiang et al., 2021; Orsini et al., 2021). Similar to earlier conflict-based methods, contributive behavioural and environmental variables require analyzing, such as:, with improved conflicts detection method through automatic computer image recognition software to assess road safety in real-time (Jiang et al., 2021; Orsini et al., 2021). Like older concepts of conflict-based technique, contributing factors other than speed require analyzing, such as:

- Geometric design.
- Type of traffic regulation.
- Categories of road users (car-to-car, car-to-bicycle, car-to-pedestrian).
- Type of traffic control (signalized or stop-controlled).
- Road classification (urban or rural).

Orsini et al. (2021) proposed an improved technique is to predict conflict-prone situations based on real-time traffic data called Real-Time Conflict Prediction Model (RTConfPM). The RTConfPM structure consists of the following features:

- Developed offline using historical traffic, weather, and road data as inputs, as well as conflict/crash data for calibration.
- Applied in real time using collected traffic, weather, and road data at the present intervals.
- Predicted future unsafe events based on collected patterns.
- Pertained an opportunity to apply an intervention strategy to prevent a predicted upcoming unsafe event.

RTConfPM is a machine-learning model that utilizes a dataset of conflicts to predict unsafe situations in real time. Specifically, RTConfPM is trained using a random forest algorithm on a dataset of vehicle-by-vehicle information obtained using radar, consisting of speed, time gap, time stamp, and vehicle class. Further, RTConfPM achieves over 93 percent accuracy, recall, and specificity in identifying unsafe situations, defined as with a high degree of accuracy at a 5-minute intervals with at least three conflicts where TTC is below 0.78 seconds. This makes it an effective tool in situations where spatial or temporally accurate crash data may be unavailable or unreliable (Orsini et al., 2021).

The identification of traffic conflicts based on evasive actions has historically posed challenges due to the limitation of early technologies in capturing these interactions. As a result, observers were equipped with an extensive list of potential evasive actions, making the process both exhaustive and prone to human error depending on partook training. Even when all evasive actions were identified, the dependency on simple individualistic observation raised concerns about the accuracy and reliability of the collected data. Furthermore, the definition of evasive actions can be inconsistent, as manoeuvres such as braking or lane changes might be executed either as a precautionary action or to enhance driving comfort, regardless of whether a crash occurs (Allen et al., 1978; Chin & Quek, 1997; Glauz & Migletz, 1980; Zheng et al., 2014).

Traffic conflict analysis, established in the work of Amundsen and Hyden (1977), emphasizes the temporal and spatial proximity between road users to be a basis for various SSMs. However, the lack of consensus on the most appropriate SSM continues to present challenges (Guido et al., 2011). Each measure incorporates a specific sensitivity to safety, leading to variability in conflict detection. Conflicts are typically recorded when specific SSM values fall below a predetermined threshold (Huang et al., 2013), yet these thresholds are often derived from historical research which potentially reduce relevance for modern intersections and roadway geometries. Consequently, relying on generalized thresholds may limit the advancement of traffic conflict analysis techniques (Zheng et al., 2014).

The validity of conflict models in representing crash risk remains a subject of debate. Some researchers argue that traffic conflicts and crashes represent distinct outcomes, influenced by the effectiveness of evasive actions or threshold set for SSMs (Guettinger, 1982; Jiang et al., 2021). From this perspective, traffic conflicts serve as a

complementary indicators of crash risk rather than direct equivalents. Alternatively, another perspective suggests a proportional relationship between conflicts and crashes, quantified through a distribution coefficient derived from maximum likelihood estimation (Jiang et al., 2021).

The effectiveness of conflict-based techniques, whether as complementary crash indicators or as surrogate safety measures, depends heavily on the chosen data collection method. Conflicts occur far more frequently than crashes, making its data collection resource intensive. Collection traffic conflicts can be gathered through either direct observation or microsimulation. The capability of conflicts to predict crash risk relies on the accuracy of the chosen data collection process outlined in Section 2.3.2. However, the selected data collection method may overlook critical factors, including (Sayed, 2021):

- Assumed representation of road users as centroid, rather than size (i.e., vehicular size).
- The identification of conflict measures involves modelling the expected travel path undertaken by the road user. This leads to a propagating naïve extrapolation; however, the avoiding action requires a complex computational probabilistic approach.

## 2.3.2 Conflict Data Collection Methods

### 2.3.2.1 Field Observation

Field observations, including on-site and video-recording observations, were collected at the survey sites. Analysts are first trained on the operational and technical aspects of recognizing traffic conflicts in terms of type and severity. Subsequently, the trained analysts are equipped with the necessary materials and sent to the study site or review site footage.

Field observation was initially proposed as a method for collecting conflict data because of its simplicity and ability to capture more contextual and first-hand information compared with video recordings, such as verifying roadway geometry and documenting unique operational characteristics in a technical memorandum (FDOT, 2021). Comparative studies indicate that conflicts observed in the field are more valid than conflicts determined by objective measures when relating them to crashes (Kruysse, 1991; Shinar, 1984). Field observation technique had been successfully applied in numerous studies (Ismail et al., 2009; Kronprasert et al., 2021; Svensson & Hydén, 2006; van der Horst et al., 2014). However, field observation manually or through video recordings is associated with reliability issues caused by inter- and intra-observer variability, additionally, the method is relatively expensive and labor-intensive (Zheng et al., 2014).

### 2.3.2.2 Computer Vision Technique

Ballard & Brown (1982) defined computer vision as: “the enterprise of automating and integrating a wide range of processes and representations used for vision perception

... such as image processing, statistical pattern classification, geometric modelling and cognitive processing.” The computer vision technique relies on video sensors as the main source of data (Ismail, 2010). The utilization of computer vision techniques to acquire traffic conflict data relies upon the implementation of computer algorithms designed to track moving objects and accordingly detect traffic conflicts from videos (Zheng et al., 2014). This extensive collective information can be used to analyze lane changing and car-following behaviors (Hidas & Wagner, 2004). To ensure optimal data quality, site footage should be recorded from an elevated position, providing high-resolution attributes such as stability, clarity, and minimum shadow interference from surrounding objects. Ismail et al. (2009), Laureshyn (2010), and Saunier & Sayed (2007) utilized computer vision techniques to evaluate road safety by leveraging its two components: a video-processing module for object detection and tracking, and an interpretation module for information extraction and conflict detection.

The computer vision technique enhances automation and objectivity, thereby minimizing the dependence on human intervention (Ismail, 2010). Video cameras and sensors, which are integral to this technique, are extensively utilized in traffic monitoring due their cost-effectiveness. This approach facilitates large-scale processing of traffic conflict data with increased efficiency and reduced potential for human error (Abdel-Aty et al., 2023; Saunier et al., 2010). Moreover, it improves the accuracy of conflict detection, identifying incidents that may not be discernible to human observers. Additionally, the real-time analysis capabilities inherent in computer vision provide valuable opportunities to develop methodologies for immediate intervention and response in traffic conflicts (Sayed et al., 2013a).

The use of computer vision for traffic conflict analysis is an evolving field requiring advancements in object tracking and classification algorithms (Sayed et al., 2013b; Opromolla & Fasano, 2021). These methods show promise as alternatives to traditional techniques by linking conflict data with crash records (Sacchi et al., 2013). Experimental analyses across diverse environments and targets are necessary to refine tracking process, while challenges like detecting low-contrast objects may be addressed through active light sources or color schemes (Opromolla & Fasano, 2021). Recent innovations, such as Sequence-to-sequence Recurrent Neural Networks (RNNs), enable vehicle trajectory predictions up to three seconds ahead for proactive conflict detection (Abdelraouf et al., 2022). Similarly, Gaussian Mixture Models (GMM) and Mclust optimize conflict detection and automate surrogate safety threshold (Ding et al., 2024).

Camera vision technique can encounter occlusion issues in heavy traffic congestion, leading to potential inaccuracies in object detection and conflict identification (Ngeni et al., 2024). For optimal coverage and reduced errors, mounting heights of 9.1 metres to 15.2 metres are preferred for installing the cameras, especially when they are positioned at the side of the roadway (Mimbela & Klein, 2007). However, cameras are

typically mounted on signal poles at a height of 5 metres, which is the recommended pole height as outlined in the Ontario Traffic Manual Book 12 Traffic Signals (Ministry of Transportation in Ontario, 2012). Therefore, errors increase as the mounting height decreases. Additionally, the computational demands of processing large volumes of video data, along with advanced object detection and tracking algorithms, pose challenges for real-time applications (Abdel-Aty et al., 2023). For instance, deep learning-based models for object tracking and video stitching significantly increase the computational complexity.

#### 2.3.2.3 LiDAR Data

Light detection and ranging (LiDAR) is a remote sensing technology that sends laser light to the target and measures the reflected light back in order to recognize the variation in wavelength and arrival time of the reflected light (Mehendale & Neoge, 2020). In this way, information about the collected objects can be obtained without making physical contact. In comparison to traditional surveying of roads and infrastructure, LiDAR's capabilities extend to producing highly accurate images of roads at high speed with minimal disruption. Another advantage is the laser sensor component of LiDAR, which is unaffected by extreme weather conditions, ensuring reliability and comprehensive distance measurements in adverse environments (Bhattarai et al., 2023; Sun et al., 2018).

Roadside LiDAR sensors have emerged as an approach for collecting real-time high-resolution traffic data (HRTD) from all road users (Sun et al., 2018). These infrastructure-based LiDAR sensors capture the surroundings in a three-dimensional point cloud data format with known positional attributes, which can be processed to generate vehicle trajectories in real time. Because of the relatively prohibitive cost compared to other sensors and camera radars; the use of LiDAR is limited.

Although the current application of LiDAR systems is in autonomous vehicles, roadside traffic applications differ in their use in several key aspects (Sun et al., 2018). For instance, roadside traffic detection requires LiDAR sensors to be installed at a fixed location, rather than on a moving vehicle. Second, LiDAR should be able to constantly detect and track vehicles, even in cases where the vehicle shape is not well-defined (shaped by 3D LiDAR points). This necessitates a scalable and robust data processing algorithm to track vehicles with fewer LiDAR points. Third, LiDAR should be able to determine the vehicle speed independently without relying on other types of sensors. Minimizing the use of various sensors is vital to avoid maintenance labor and higher budgets.

Studies covering roadside LiDAR-based trajectory data for proactive safety analysis are at the start of surging. Bhattarai et al. (2023) presented a methodology to detect rear-end conflicts at signalized intersections with the aid of roadside LiDAR sensors to evaluate correlation with crashes. Whereas the LiDAR sensors gathered high-resolution vehicle trajectory data then data is processed and clustered, afterwards subsequently employed to establish SSMs for evaluating conflict risk at each frame captured by the

sensor. Through integrating time-based, deceleration-based, and severity-based SSMs, the proposed methodology facilitated rear-end conflict risks to diagnose safety conditions and surpass the reliance on historical crash records. The approach covered in this study can potentially be developed through analyzing other conflict types such as lane-change and crossing; other traffic control types such as stop-controlled and roundabouts.

Other studies covered the potential capabilities of LiDAR technology to extract information about roadside assets in which can be a potential road hazard.

## **2.4 Surrogate Safety Measures**

Surrogate Safety Measures (SSMs) are quantitative measures for indicating conflicts, which is built upon examining temporal or spatial proximity of road users in anticipation to a point of collision (Kronprasert et al., 2021). These indicators provide critical insights into the severity and frequency of safety-related interactions, allowing analysts to evaluate potential traffic risks effectively. Nevertheless, it is essential to acknowledge that the ability of SSMs to classify actual collisions is a matter of ongoing research.

Conflict indicators offer several key advantages as compared to crash-based safety measures (Ismail, 2010): 1) the occurrences of conflicts are more frequent than collisions, accomplishing a proactive method in preventing collisions and injuries; 2) the quantitative characteristic overcomes subjectivity associated with traditional observation-based techniques; 3) the capability to provide insight into the severity of traffic conflicts that can allow comprehensive safety evaluation; and 4) the adoption in numerous studies enables validation and cross-comparisons.

In the realm of conflict indicators, SSMs, several fundamental problems have been identified (Chin & Quek, 1997). These problems arise due to inconsistent and clear definition of “evasive action,” leading to uncertain assumptions about road users’ behaviours, undertaken actions, and questionable reliance on conflict indicators in measuring safety. The following subsections define various SSMs, while Table 1 lists major drawbacks of these indicators used to evaluate severity of vehicle-to-vehicle conflicts.

### **2.4.1 Time-to-Collision**

Time-to-Collision (TTC) is defined as the time required for two vehicles to collide if they continue at their present speeds on the same path (Kronprasert et al., 2021). In other words, TTC quantifies the time it would take for a following vehicle to reach the position of the leading vehicle – referred to as the conflict point – assuming both vehicles maintain their current velocities, known as conflict speeds. TTC is one of the most notable and widely used SSM as it directly measures the available time for road users to avoid a potential collision. Typically, lower TTC value suggests that a collision is imminent unless an evasive action is taken. Mathematically, the TTC is calculated as the ratio of the distance

between the following and leading vehicles to the relative velocity between them, as shown in Equation 2.4:

$$TTC = \frac{|d_2 - d_1|}{|v_2 - v_1|} \quad (2.4)$$

Whereas  $d_2$  is the position of the following vehicle (m);  $d_1$  is the position of the leading vehicle (m);  $v_2$  is the velocity of the following vehicle (m/s); and  $v_1$  is the velocity of the leading vehicle (m/s).

Figure 3 presents a schematic diagram of the TTC calculation for a through vehicle and a right-turn vehicle.

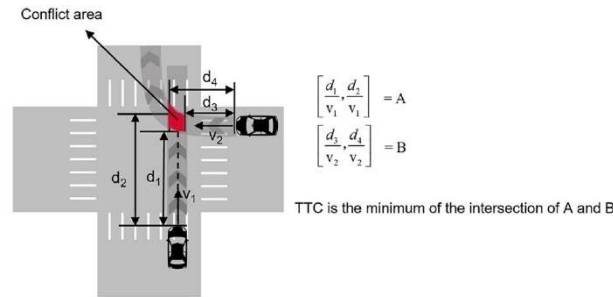


**Figure 3 Schematic Diagram of TTC Calculation (Zheng et al., 2024)**

The calculation of TTC is mostly applied towards rear-end conflicts whereas the vehicles are traveling in the same lane and direction. In such case, the parameters of Equation 2.4 are treated as one-dimensional problem. Studies such as Ward et al. (2015) proposes a generalized TTC calculation method for diverse traffic scenarios incorporating vehicles states including position, speed, heading, and yaw rate, through computing two TTC values: 1) straight-line TTC by assuming straight trajectories and constant speeds, and 2) curvilinear TTC by accounting for yaw rates and orientation changes.

TTC can be computed at every timestamp during a vehicle interaction, resulting in multiple TTC values for the pair of vehicles. Typically, a minimum TTC over the observation period is reported, as it represented the closest approach and highest risk.

A related metric worth noting is Time-to-Accident (TA), which measured the TTC at the precise moment an involved road user takes an evasive action. Both TTC and TA rely on threshold values to distinguish between severe and non-severe conflicts.

#### 2.4.2 Post-Encroachment Time

Post-Encroachment Time (PET) measures the time difference between a vehicle leaving an encroachment region and an approaching vehicle entering the same region (Peesapati et al., 2018). PET relies on two timestamps; the time when the first vehicle ends encroachment over the area of conflicts and the time when the second vehicle enters the area of conflicts. Hence, the PET can be presented through Equation 2.5 as follows:

$$PET = t_2 - t_1 \quad (2.5)$$

Whereas the  $t_1$  and  $t_2$  values represent the time in seconds when the leading vehicle and following vehicle, respectively, cross (s): time recorded for vehicle 2 to pass the same conflict area.

Figure 4 presents a schematic diagram of the PET calculation for a through vehicle and a right-turn vehicle.

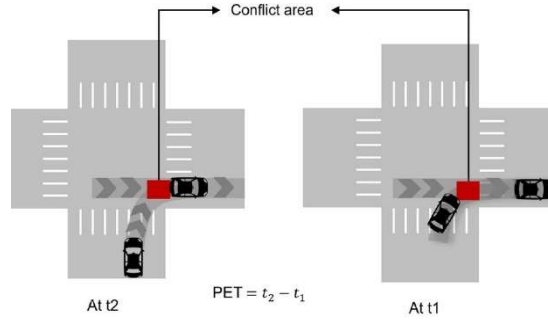


**Figure 4 Schematic Diagram of PET Calculation (Zheng et al., 2024)**

A PET value of 0 indicates a crash occurrence, while non-zero indicates crash proximity. Thus, lower PET indicated higher probability of collision. Nevertheless, PET does not capture the initial cause of conflict, nor the evasive actions taken by the drivers; the measure only provides proximity of interaction to a collision.

SSAM applies a default PET threshold of 5 seconds, a value not explicitly justified in the literature. In contrast, the American Association of State Highway and Transportation Officials (AASHTO) Green Book (7) recommends a minimum time gap for intersection sight distance ranging from 5.5 to 12 seconds, depending on factors such as traffic control type, design vehicle, and the intended manoeuvre. The SSAM default PET threshold being lower than the recommended minimum time gap suggests that the threshold may indicate insufficient temporal separation for safe manoeuvring, potentially underestimating the safety needs outlined in intersection design standards.

#### 2.4.3 Gap Time

Gap time is a PET variant, calculated at each time step through projecting movement of interacting road users in terms of space and time (Ismail et al., 2009). With that being said, the parameter is essential to define priority rules (Huang et al., 2013). The default minimum gap time on VISSIM is defined as 3 seconds.

#### 2.4.4 Time Headway

Headway is an indicator used to determine the severity of traffic incident, through measuring elapsed time between front of lead and following vehicles passing at same location (Mahmud et al., 2017). As general rule of thumb, the US recommends a headway of less than 2 seconds for safe car-following. Meanwhile Germany uses the term “half the

speedometer” meaning the headway distance should be 40 metres for a car travelling at 80 km/h, which translates to 1.8 seconds. Time Headway is calculated per Equation 2.6 below:

$$H = t_2 - t_1 \quad (2.6)$$

Whereas  $t_2$  represents the time of following vehicle to pass certain location; and  $t_1$  represents time of lead vehicle to pass same location.

#### 2.4.5 Deceleration Rate to Avoid a Crash

Deceleration Rate to Avoid a Crash (DRAC) was first defined by Almqvist et al. (1991) as the differential speed between a following vehicle and its corresponding leading vehicles, divided by their closing time. The closing time accounts for the relative positions of the following and leading vehicles at a given moment, while considering the length of. Whereas the leading vehicle. In this context, the leading vehicle initiates an action (i.e., braking for a traffic light or stop sign and changing lanes and/or accepting a gap), while the following vehicle must react promptly avoid a dangerous interaction.

#### 2.4.6 Proportion of Stopping Sight Distance

Proportion of Stopping Sight Distance (PSD) is defined as the ratio between the remaining distance to potential point of collision (RD) to the minimum acceptable stopping distance (MSD) (Mahmud et al., 2017). Hence, PSD equation is presented as follows:

$$PSD = \frac{RD}{MSD} \quad (2.7)$$

The MSD represents the shortest distance required for a vehicle traveling at a given speed to come to a complete stop, assuming it decelerates at the maximum acceptable rate (Guido et al., 2011). It is calculated using Equation 2.8:

$$MSD = \frac{V^2}{2d} \quad (2.8)$$

Whereas  $V$  represents the approaching velocity (m/s); and  $d$  is maximum acceptable deceleration rate (m/s<sup>2</sup>).

Table 1 provides a comprehensive overview of the limitations associated with conflict indicators commonly used in studies.

**Table 1 Drawbacks of Conflict Indicators**

| <b>Conflict Indicator</b>                    | <b>Drawback</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Time-to-Collision</b>                     | <ul style="list-style-type: none"> <li>• Disregards the actual acceleration and deceleration until the collision has occurred (Ozbay et al., 2008).</li> <li>• Only assumes if the speed of following vehicle is larger than leading, a collision will occur resulting in ignoring many potential conflicts (Mullakkal-Babu et al., 2017; Ozbay et al., 2008).</li> <li>• Assumes constant speed and path, making it suitable primarily for rear-end conflicts, yet fails to capture complexity of real-world driving behaviour and decision-making process (Hou et al., 2014; Zheng &amp; Sayed, 2019).</li> <li>• Overlooks driver reaction time, a critical factor in collision avoidance, by assuming instantaneous reaction which is unrealistic. Typically, a critical TTC value lower than the perception and reaction time should be considered unsafe (Mahmud et al., 2017). Whereas, the minimum TTC is approximately 1.6 seconds for low level conflict (Sayed et al., 1994), yet the average brake reaction time is approximately 2.3 seconds (McGehee et al., 2000).</li> </ul> |
| <b>Time-to-Accident</b>                      | <ul style="list-style-type: none"> <li>• Lacks considering reaction time (Hyden, 1987)..</li> <li>• Requires road users to be on a collision course (Hyden, 1987).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Post-Encroachment Time</b>                | <ul style="list-style-type: none"> <li>• Only to be calculated when following vehicle passes through the common area, meaning the indicator is only applicable to post-conflict estimation (Davis et al., 2011).</li> <li>• Regarded as most-suitable for only crossing conflicts at intersections (Jiang et al., 2021).</li> <li>• Easily misjudges rear-end conflicts, for instance when speed of following vehicle is slower than leading vehicle, a PET will yet be generated although collision might not occur (Jiang et al., 2021).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>Gap Time</b>                              | <ul style="list-style-type: none"> <li>• Lacks representation of complex behaviours such as ‘pushing’ into queues in congested traffic conditions and various types of interactive yielding (Archer, 2005).</li> <li>• Inconsistent estimation as it is determined by driver’s comfortability (Archer, 2005). In which might lead to misrepresentation of the presence and severity for conflict between two vehicles.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Time Headway</b>                          | <ul style="list-style-type: none"> <li>• Inconsistent relation between the duration of time headway and occurrence of traffic conflicts nor accident involvement (Vogel, 2003).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Deceleration Rate to Avoid Crash</b>      | <ul style="list-style-type: none"> <li>• Fails to accurately identify the potential traffic conflicts due to its inconsideration of essential factors in the braking process (i.e., vehicle braking capability over time for prevailing road and traffic conditions) (Cunto, 2008; Meng &amp; Weng, 2011).</li> <li>• Unsuitable for lateral movements (Mahmud et al., 2017).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Proportion of Stopping Sight Distance</b> | <ul style="list-style-type: none"> <li>• Indicates higher percentage of vehicles interaction and time exposure to conflicts compared to other measures, thus tends to be less focused on specific unsafe areas (Guido et al., 2011).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

## 2.5 Simulation-Based Conflict Technique

The Simulation-Based technique can utilize SSAM, developed by the Federal Highway Administration (FHWA), to estimate traffic conflicts by using simulated vehicle trajectories (Gettman et al., 2008; Gettman & Head, 2003). The SSAM model is supplemented with the outputted trajectory files from microscopic simulation models such as VISSIM, AIMSUN, PARAMICS, and TEXAS. In SSAM, various traffic conflict indicators are calculated such as TTC, PET, deceleration rate, maximum speed, and speed differential. Then based on specific thresholds for TTC and PET, SSAM can identify conflicts as well as represent them both visually and statistically for the three manoeuvre types: rear-end, lane-change, and crossing. Figure 5 summarizes the methodology undertaken for SSAM. Notably, the SSAM approach has been gaining traction in safety evaluation (Al-Ghandour et al., 2011; Essa & Sayed, 2015b; Fan et al., 2013b; Li et al., 2016; Liang et al., 2020; Rajeswaran et al., 2023; Shahdah et al., 2014; Vasconcelos et al., 2014; Wang & Stamatiadis, 2014).

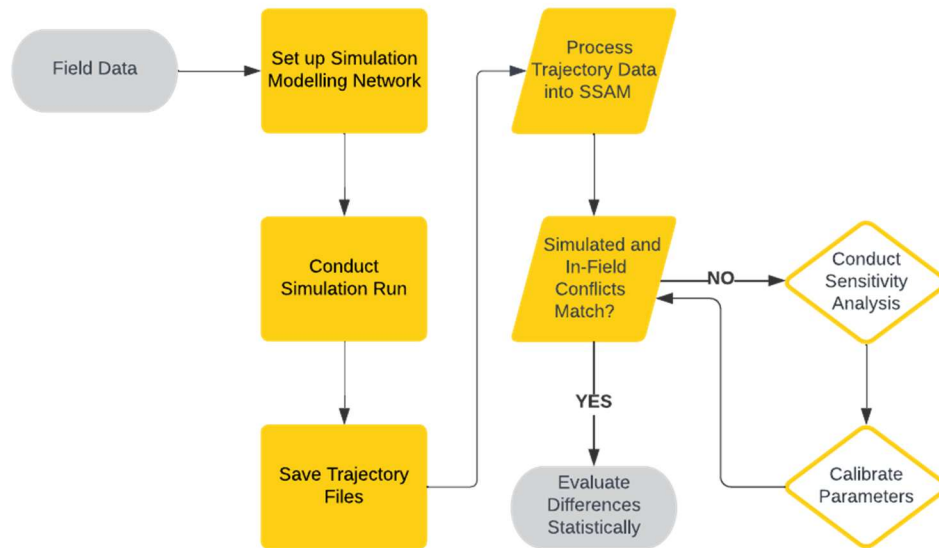


**Figure 5 SSAM Methodology Flowchart**

Alternatively microscopic simulation software can be directly utilized to validate the network by comparing simulated against observed network performance. This process involves evaluating metrics such as travel time, speed, and conflicts to ensure accuracy. For instance, Bachmann et al. (2011) presented a redefined TTC to enhance the accuracy of traffic conflict analysis using PARAMICS microsimulation model to evaluate truck-only highway in the Greater Toronto Area. Similarly, (El-Tantawy et al., 2009) evaluated truck lane restriction strategies on downtown Toronto's Gardiner Expressway using PARAMICS microsimulation. They utilized algorithms for conflict detection by recording relevant attributes for each conflict type, with TTC serving as the surrogate safety measure.

However, there are two primary challenges associated with employing simulation-based technique in road safety studies. First, the driving behaviour is modeled to avoid a collision, rendering it difficult to represent vehicle interactions and near misses in a fully accurate manner. Second, microsimulation models contain numerous parameters which can significantly influence the outcomes of simulated conflicts.

Table 2 summarizes additional studies that have contributed to the road safety assessment field through Simulation-Based technique to obtain conflicts, in a chronological order. Therefore, these limitations emphasize the requirement to consider validating and calibrating the microsimulation model to ensure a correlated relationship between real-world and simulated conflicts with the aim of securing a reliable safety study.

**Table 2 Previous Simulation-Based Technique Studies**

| Authors                    | Study Area/Geometry                                                   | Conflict Type                           | Performance Measure | Safety Measure Indicator                                    | Calibration Targets                         | Calibration Method                                                                                                                                                             | Validation Method                                                                                                                                                                                     | Selected Parameters                                                                                                                       |
|----------------------------|-----------------------------------------------------------------------|-----------------------------------------|---------------------|-------------------------------------------------------------|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| (Dijkstra et al., 2010)    | 569 combinations of signalized and unsignalized intersections         | Lateral, Frontal, Rear-end              | S-PARAMICS          | TTC                                                         | N/A                                         | SAS Procedure GENMOD                                                                                                                                                           | Simulated conflicts through PARAMICS were compared to observed crashes using Poisson and Negative Binomial.                                                                                           | N/A                                                                                                                                       |
| (Al-Ghandour et al., 2011) | 80 approaches of roundabouts                                          | Approach, Slip-lane, Merging, Meeting   | SSAM                | TTC                                                         | Gap Acceptance                              | SAS Procedure GENMOD                                                                                                                                                           | T-tests were conducted to compare the significance of differences between model performances. A statistical analysis of TTC was performed to evaluate normal distribution.                            | N/A                                                                                                                                       |
| (Fan et al., 2013b)        | Freeway                                                               | Rear-end, Lane-change, Ramp-lane change | SSAM                | TTC, PET, Deceleration Rate, Max. Speed, Speed Differential | Travel Time                                 | Genetic Algorithm (GA) was used to automate the calibration process. Minimized Average Percentage Error (MAPE) was used for two-stage calibration (travel time and conflicts). | Linear regression analysis was conducted to compare between observed and simulated to observe statistical influence. The Spearman rank correlation coefficient was used to compare safety ranking.    | Standstill Distance, Headway Time, Threshold for entering following state, Safety Distance Reduction Factor, Minimum Gap Time.            |
| (Vasconcelos et al., 2014) | Urbanized intersections (t-intersection, four-legged, and roundabout) | Rear-end                                | SSAM                | TTC, Speed Differential                                     | Deemed unnecessary                          | Deemed unnecessary                                                                                                                                                             | First method through comparison between simulated conflicts and predicted injury accidents from analytic models. Second method through comparison between simulated conflicts and observed conflicts. | N/A                                                                                                                                       |
| (Essa & Sayed, 2015b)      | 2 urban signalized intersections                                      | Rear-end                                | SSAM                | TTC                                                         | Arrival Type (Platoon Ratio), Desired Speed | 2-Stage Calibration was used. First is MAPE for average delay time. Second is GA being applied to estimate best values of the selected parameters.                             | Adjacent intersection was calibrated in similar process.                                                                                                                                              | Standstill Distance, Headway Time, -ve and +ve Following Thresholds, Reduction Factor, Start upstream of stop line, Desired Deceleration. |
| (Li et al., 2016)          | Freeway                                                               | Rear-end, Lane-change                   | SSAM                | TTC                                                         | Travel Time                                 | MAPE was used to calculate the difference between observed and simulated travel time, until within accepted range. Similar process was followed for conflicts.                 | Comparison of field and predicted Hourly Conflict Risk Index (HCRI).                                                                                                                                  | Ramp Traffic Volume, Acceleration Lane Length.                                                                                            |

**Table 2 Continued Previous Simulation-Based Technique Studies**

| Authors                   | Study Area/Geometry                                                                               | Conflict Type         | Performance Measure          | Safety Measure Indicator | Calibration Targets                          | Calibration Method                                                                                                                                                                          | Validation Method                                                                                                                                                                                                                                                      | Selected Parameters                                                                                                                                                                                                                                                                                    |
|---------------------------|---------------------------------------------------------------------------------------------------|-----------------------|------------------------------|--------------------------|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| (Khavas et al., 2017)     | Freeway                                                                                           | N/A                   | N/A                          | N/A                      | Free Flow Speed, Speed at Capacity, Capacity | Van Aerde equation was used to estimate targets, then top 10 impacting parameters were ranked using Elementary Effect (EE) and selected. Then Neural Network (NN) was used for utilization. | Simulated outputs related to traffic stream parameters were compared to the original set of desired traffic stream parameters. Also, a randomly selected subset of traffic stream characteristics along with field data from another freeway were used for validation. | Acceleration Deceleration (Own and Trail), Safety Distance Reduction Factor, Standstill Distance, Headway Time, -ve and +ve Following Thresholds.                                                                                                                                                      |
| (Essa & Sayed, 2020b)     | 2 urban signalized intersections                                                                  | Rear-end              | Real-time                    | TTC                      | Arrival Type (Platoon Ratio), Desired Speed  | 2-Stage Calibration was used. First is MAPE for average delay time. Second is GA being applied to estimate best values of the selected parameters.                                          | $Y = \sum_{i=1}^N \sum_{j=1}^C y_{ij}$ <p>The equation was used to calculate the predicted number of rear-end conflicts for approach per hour.</p>                                                                                                                     | Standstill Distance, Headway Time, -ve and +ve Following Thresholds, Reduction Factor, Start upstream of stop line, Desired Deceleration.                                                                                                                                                              |
| (Liang et al., 2020)      | Freeway                                                                                           | Rear-end, Lane-change | SSAM                         | N/A                      | Average Conflict Counts                      | T-test was used for parameter selection. Different parametric values were chosen by level of magnitude (low, medium, and high).                                                             | N/A                                                                                                                                                                                                                                                                    | Look ahead distance (Max.), Look ahead distance (Observed), Temporary lack of attention duration, Temporary lack of attention probability, Standstill distance, Headway time, Threshold for entering state following state, Standstill acceleration, General behaviour, Desired position at free flow. |
| (Rajeswaran et al., 2023) | 91 four-legged signalized intersections and 13 other signalized intersections for transferability | N/A                   | SSAM, Negative Binomial (NB) | PET                      | PET                                          | Calibration based on SSAM and meeting minimum threshold as observed.                                                                                                                        | CALIBRATOR software was used to calibrate models for the 13 intersections to assess transferability.                                                                                                                                                                   | N/A                                                                                                                                                                                                                                                                                                    |

### 2.5.1 Methods for Calibrating Simulation Models

The process of model calibration and validation is essential for the credibility of any microsimulation model, ensuring that computational results are reliable for real-world applications (Oketch & Carrick, 2005). Validation is defined as the extent achieved by model fundamental rules and relationship to reflect actual driving behaviour. Meanwhile, calibration consists of assigning the appropriate parametric values to portray local traffic conditions into model. The comparison of modeled and observed data for operation analysis can be computed through variety of mathematical methods including GEH statistic and Mean Absolute Percentage Error (MAPE) function (Essa & Sayed, 2020b; Oketch & Carrick, 2005). GEH and MAPE are represented by the following Equations 2.9 and 2.10 respectively below:

$$GEH = \sqrt{\frac{(M-O)^2}{0.5 \times (M+O)}} \quad (2.9)$$

Whereas  $M$  denotes simulated flows (vehicle/hour), and  $O$  denotes observed flow (vehicle/hour). Various GEH values indicate goodness of fit as outlined below:

|                           |                                                    |
|---------------------------|----------------------------------------------------|
| <b>GEH &lt; 5</b>         | Flows is adequate to be considered as a good fit   |
| <b>5 &lt; GEH &lt; 10</b> | Flows may require further investigation            |
| <b>10 &lt; GEH</b>        | Flows is inadequate to be considered as a good fit |

$$MAPE = \frac{\sum_{i=1}^n |y_i - E(Y_i)|}{\sum_{i=1}^n y_i} \quad (2.10)$$

Whereas  $y_i$  denotes the number of field-measured conflicts at the datapoint ( $i$ ) (i.e., conflicts for one approach for 1 hour);  $E(Y_i)$  is the number of conflicts predicted by SSAM or real-time safety models at datapoint ( $i$ ); and  $n$  denotes the sample size (i.e., data collection period in hours multiplied by number of approaches).

#### 2.5.1.1 Calibration Targets

##### 2.5.1.1.1 OD Demand

The procedure involves estimation of Origin-Destination (OD) trip matrix based on observed link volumes and turning movement counts at intersections (Oketch & Carrick, 2005). Fundamentally, the observed data must be balanced so that sums of related entities – volumes of destination and origin trips – are consistent.

##### 2.5.1.1.2 Queue Length

The procedure involves comparing collected queue data at the intersection of study with the simulated model results. The difference should be within an acceptable tolerance (Oketch & Carrick, 2005).

#### 2.5.1.1.3 Travel Time

Travel time observations are converted into a range based on the mean travel times of vehicles in the study direction of traffic. The modeled travel time values are then compared to check if they fall within the observed range (Oketch & Carrick, 2005).

#### 2.5.1.2 Calibration Approaches

##### 2.5.1.2.1 Genetic Algorithm

Genetic Algorithm (GA) is a heuristic optimization technique on mechanics and natural selection and evolution (Park & Qi, 2005). Generally, it possesses three operators including selection, crossover, and mutation (Dadashzadeh et al., 2019b). The procedure is a simplified model which was inspired by natural evolution. The genetic operators used by GA to find a suitable solution for the problem are summarized below along with interactions:

- *Selection*: initial population (generation) is selected by GA randomly, particularly, the parents of selected population (Dadashzadeh et al., 2019b).
- *Crossover*: GA combines the selected parents to produce offspring (children) for the next generation using multiple methods such as single-point, double-point, or uniform crossover (Dadashzadeh et al., 2019b). Each method is beneficial at certain stage, as an example, uniform is beneficial for initial population. The probability of applying crossover is within typical range between 0.7 and 1.0 (Leal et al., 2020).
- *Mutation*: the purpose of mutation is to maintain an adequate level of diversity in the populations (Leal et al., 2020).

In Park & Qi (2005), GA is used to calibrate the parameters of the VISSIM model. As part of the calibration process, critical parameters are identified based on sensitivity testing and feasibility tests, including simulation resolution, the number of observed preceding vehicles, the maximum look-ahead distance, the average standstill distance, saturation flow rate, minimum headway, minimum gap time, desired speed, and their appropriate ranges. Afterwards, the GA generates several individuals in the population, each representing a feasible set of parameters. The simulated results (such as travel time) can be obtained by incorporating these parameter sets into the simulation model. Fitness levels are assigned after each generation or run. In Equation 2.11 the fitness function compares the results of the simulation to those of the field survey:

$$FV = \frac{|TT_{field} - TT_{sim}|}{TT_{field}} \quad (2.11)$$

Where,  $FV$  denotes the fitness value,  $TT_{field}$  is the average travel time obtained from field data (s); and  $TT_{sim}$  is the average travel time obtained from simulation model (s).

#### 2.5.1.2.2 Golden-Section Search

The Golden Ratio-based Genetic Algorithm (GRGA) enhances real-coded genetic algorithms by incorporating Golden Ratio (GR) to improve solution accuracy (Zhang et al., 2015). Through the utilization of golden-section search, which is a developed version of Genetic Algorithm (GA), the golden-section search can address the common issue of populations converging to local minima.

In Park et al. (2019), golden-section search was applied to identify crash-free paths for lane changing by autonomous vehicles. The search was developed to find number of final design variables for minimum cost function, based on final vehicles' position and calculated optimal points. The golden-section search follows the procedure below.

Functional areas allowing lane change movements were determined, measuring from the moving obstacle on current lane to initiate the search after being divided to four zones presented in Figure 6: deceleration limit zone, possible zone, overtaking zone, and acceleration zone.

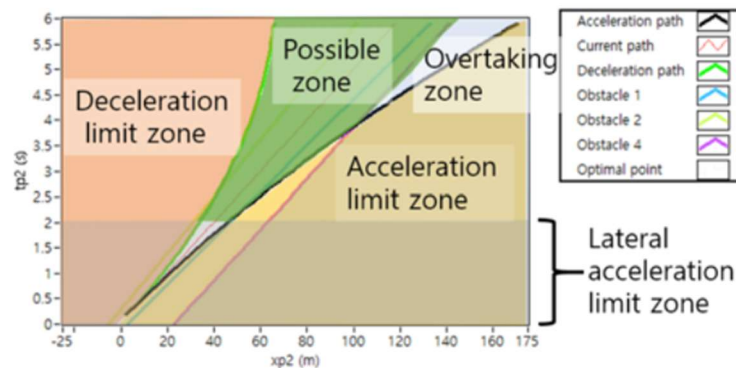


**Figure 6** Area according to the final reached position (Park et al., 2019)

Afterwards, the time within the search period and moving obstacles at the destination are divided as shown in Figure 7. As it had been visualized, the area is divided into three sections per number of cars on the lane planned to change to. The first search section starts applying golden-section search for the planned lane, then the search proceeds to using unconstrained optimization method with setting penalties on car collision and considering restrictions as visualized in Figure 8.

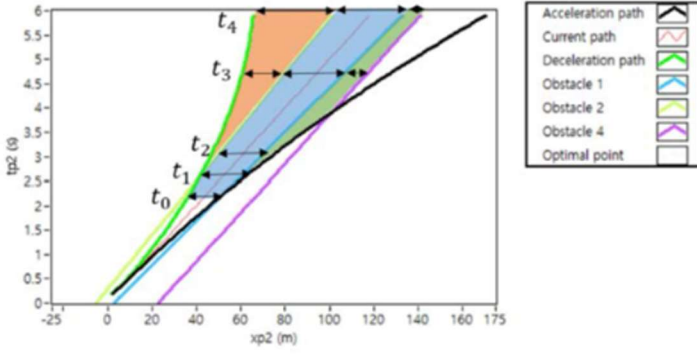


Figure 7 Initial Golden-Section Search Direction (Park et al., 2019)

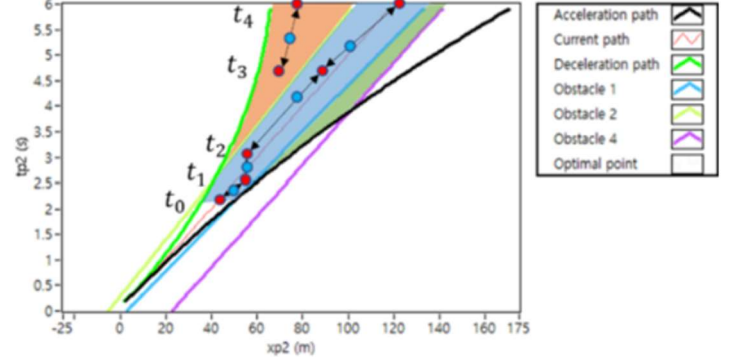


Figure 8 Final Golden-Section Search Direction (Park et al., 2019)

The figures above reference parameters, in which are explained as follows:  $t_0$  is the minimal time limited by the lateral speed;  $t_2$  is the time from the start of the lane change based on driver's preference;  $t_1$  is the average between  $t_0$  and  $t_2$ ;  $t_4$  is the longest time needed to change lanes.

#### 2.5.1.2.3 Artificial Neural Network

Artificial Neural Networks (ANN) is an Artificial Intelligence (AI) technique mimicking the function of human brain. As the ANN is flexible to cope, anticipate, and result with high predictability through findings patterns through experience (Vishnuvardhan et al., 2023). ANN is composed of connected artificial neurons and been organized into layers as Figure 9, whilst the technique is trained using weights between neurons to balance inputs and outputs (Agatonovic-Kustrin & Beresford, 2000).

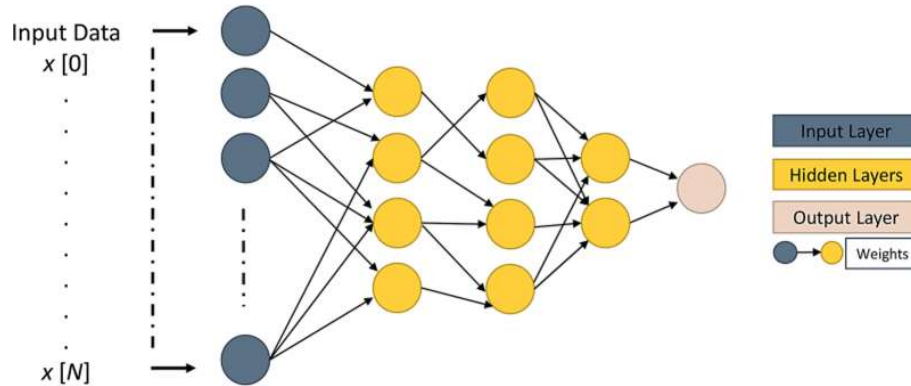


Figure 9 Artificial Neural Network Representation

In reference to Figure 9, the ANN is composed of three types of layers. The following summarizes each of the scopes individually:

- *Input Layer* is the first layers with obtained inputs to be transmitted to the next layer.
- *Hidden Layer* resides between the input and output layer and is composed of computations undertaken.

- *Output Layer* is usually only consisting of one output in ANN, and it is accountable of obtaining the results.

Ištoka Otković et al. (2023) proposes a methodology to train ANN to establish a calibration of the microsimulation traffic model. Databases of simulation results for observed traffic indicators were inputted in VISSIM, to be used for the first ANN layer – Input Layer – and initiate the training process. As a second step, the trained neural networks were evaluated based on their performance in training and testing datasets. During the validation process for the third step, neural networks were assessed to determine their ability to generalize to new unused datasets. Then, the produced predict-function of neural network with most accurate prediction of simulation results, gets programmed into the microsimulation model calibration. As an ultimate step, a comparison between modelled and measured values of traffic parameters is conducted and optimized.

## 2.5.2 Validation Methods

The validation process for SSAM involved a rigorous evaluation of field, model and software; consisting of three activities as described in the following subsections (Gettman et al., 2008).

### 2.5.2.1 Theoretical Validation

The theoretical validation efforts are to assess the capabilities of SSAM to represent the safety of intersections and interchanges in a series of comparison for eleven case studies including four-legged signalized intersections consisting of permitted and protected left-turns, median U-turn bays, t-intersections, interchanges with three-phase and four-phase timing, and roundabouts.

Theoretical validation revealed a trade-off of surrogate safety measures rather than preferable safer alternative designs. Notably, it was suggested for research to develop a “conflict index” or “safety index” acting as a weighting factor of observed conflicts at distinct types, frequencies, and severities.

### 2.5.2.2 Field Validation

The field validation scope investigated the direct accuracy of surrogate safety assessment, through analyzing eighty-three four-leg urban signalized intersections. These intersections were modeled in VISSIM, simulated, and assessed with SSAM. Afterwards, a comparison between the simulated conflicts and actual recorded crashes was evaluated using five statistical tests.

Field validation concluded correlation between simulated conflicts provided by SSAM and the recorded crash data, except during path-crossing maneuvers where it was under-represented in the simulation. As well as it was found a correlation between conflicts and crashes in reference to ADT volumes.

#### 2.5.2.3 Sensitivity Analysis

Sensitivity analysis conducted the assessment on five of the previous eighty-three intersections using four simulation systems: AIMSUN, PARAMICS, TEXAS, and VISSIM, with the goal of determining bias of surrogate safety measures.

As a result of sensitivity analysis, modeled intersections via TEXAS had the highest conflict frequency – ten times higher than VISSIM. Simulation models show different range of driving behaviour when avoiding conflicts. Also, a bias towards rear-end conflict events was instituted, although multiple speed limits were inputted. The various microsimulation software shows inconsistent default driving behaviour and modeling assumptions at higher congestions levels.

## 2.6 Research Gaps

Past research on simulation-based study mainly focuses on adjusting simulation models based on traffic performance metrics, often overlooking safety performance parameters and estimated conflicts (Al-Ghandour et al., 2011; Fan et al., 2013a; Li et al., 2016). This emphasizes the need for further investigation into creating comprehensive metrics for assessing the safety implications of calibration decisions for simulation models while balancing operational performance. A significant portion of the research studying simulated conflicts with SSAM, as shown in Table 2, heavily relies on camera vision techniques or commonly uses TTC as a surrogate metric. However, this reliance on camera vision may not fully consider the limitations of occlusion problems and computational complexity (Section 2.3.2.2). On the other hand, research that focuses on vehicle-vehicle conflicts at signalized intersections uses LiDAR data processed through machine learning algorithms to extract important PET values and conflict angles (Ansariyar, 2023). This presents an opportunity for further exploration, especially in examining the feasibility of using microscopic simulation models to develop safety-focused predictive models. While research has started to explore the aspect of simulating conflicts using LiDAR for conflict data collection, it has mainly concentrated on individual intersections rather than entire corridors (Ansariyar & Taherpour, 2023). This raises questions about whether observed conflicts at one intersection are affected by conditions at preceding intersections within the same corridor, suggesting potential opportunities for investigating whether microscopic observed parameters extracted from LiDAR data along a corridor can contribute to the development of predictive models.

### 3 Methodology

The literature review highlights the significant role of simulation-based techniques in road safety assessment, in predicting traffic performance, and managing potential issues encountered with the calibration datasets.

The analysis of past methodologies shows a strong focus on operational performance measures such as speed, travel time, and queue lengths when calibrating simulation models. However, this approach often overlooks the complex dynamics of road safety. Another gap in the literature is the traditional treatment of vehicles only as centroids, leading to approximations of vehicle interactions and SSMs. Additionally, previous studies relying on camera vision for conflict data collection faced issues with occlusion and computational complexity, while using Time-to-Collision (TTC) tended to overestimate rear-end conflicts. Furthermore, the use of LiDAR sensor technology for simulation-based safety assessment calibration is new. Investigating road corridors rather than focusing solely on individual intersections may offer a more comprehensive understanding of the cumulative effects of calibration on operational and safety performance. The methodology proposed in this thesis, delineated through a seven-step process as illustrated in Figure 10, is designed to achieve the mentioned research objectives in the Introduction chapter.

The first step involves collection of trajectory data (Section 3.1), including traffic volumes, turning movement counts, signal indications, and conflicts. The collected trajectory data from LiDAR API is processed to extract travelled distance, speed, acceleration and deceleration, and headway, considering real-time vehicular dimensions. In the subsequent step (Section 3.2), the simulation model is calibrated by adjusting three parameter groups – vehicle performance, car-following model, and lane-changing model - based on metrics derived from generated LiDAR trajectory data. The third step (Section 3.3) involves model validation through travel time, using the Google Distance Matrix API. Following this, in the fourth step (Section 3.4), model parameter are identified that have a statistically significant effect on simulated conflicts. The fifth step (Section 3.5) focuses on calibrating the parameters identified to influence simulated conflicts to closely replicate observed conflicts. The sixth step (Section 3.6) creates a separate calibrated model based only on conflicts-effective parameters. The seventh step (Section 3.7) consists of validating the calibrated models by comparing simulated and observed conflicts in terms of PET distributions and heat maps for each intersection of corridor. Lastly, effectiveness of parameters and simulated models are assessed in the final step (Section 3.8), in addition to an overall modelled corridor overview in term of changes in 85<sup>th</sup> percentile PET and speed and travel time errors.

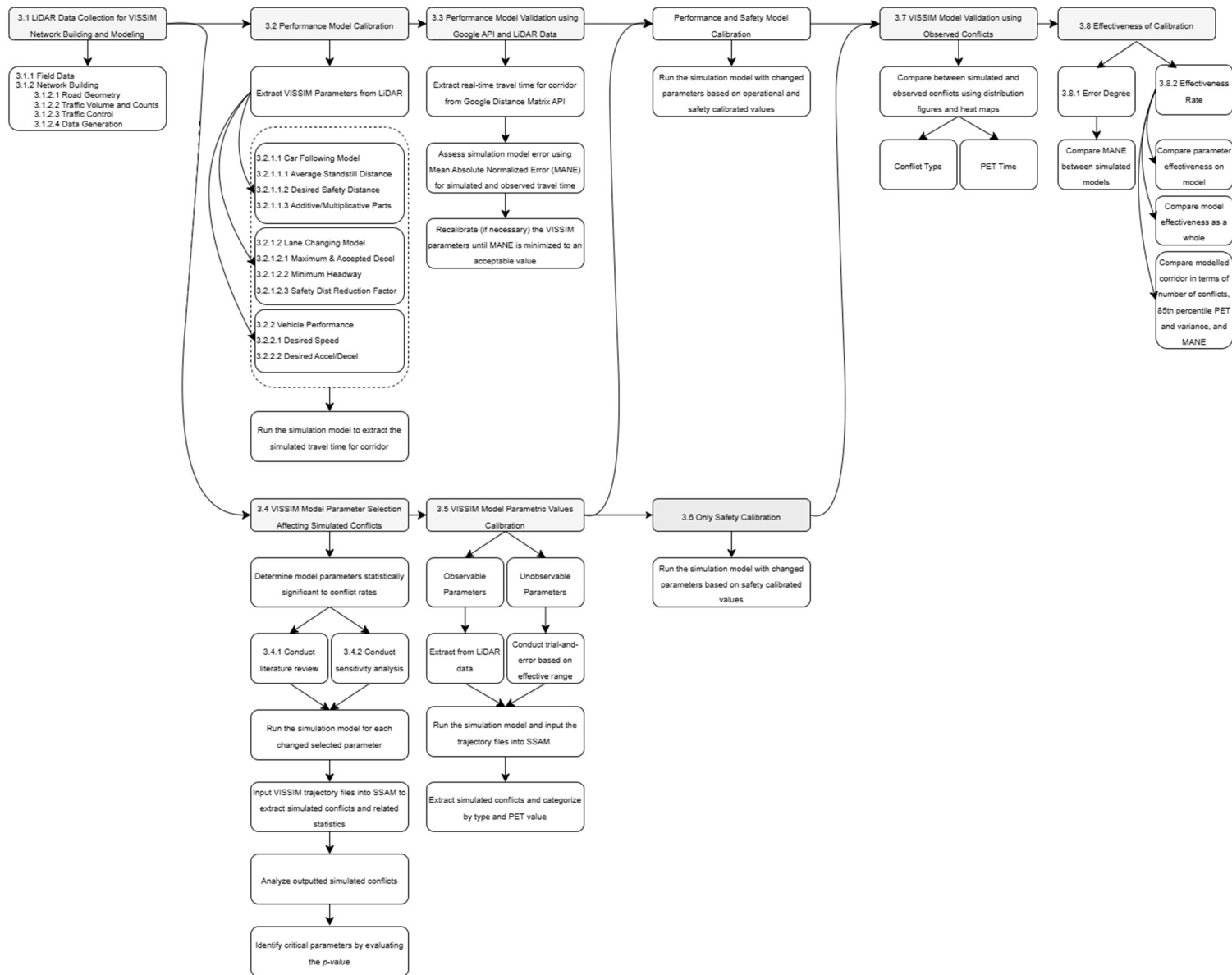


Figure 10 Thesis Framework



#### 3.1.2.2 Traffic Volume and Turning Movement Counts

LiDAR sensor data can be extracted through two methods: real-time access via the LiDAR API or the web-based Insight Platform. Real-time extraction involves processing raw data from each sensor using Python to filter outliers, classify road user, and cluster information by vehicle IDs and geocoordinates to determine volumes, turning movement counts, and headways. Alternatively, the Insight Platform provides preprocessed data, including traffic volumes, signal phasing, and conflicts, which is accessible one day later.

For this thesis, traffic volumes and Turning Movement Counts (TMC) for July 2022 were sourced from the Insight Platform, recorded at 15-minute intervals. The vehicle volume data was further categorized into AM and PM peak hours, with an average volume calculated for each peak hour. While API processing was initially explored, the Insight Platform offered a more reliable approach, avoiding issues such as misclustering of illegal turning movements.

#### 3.1.2.3 Traffic Control

The timing plans for the study area were acquired from the Insight Platform, which is directly connected to the traffic controllers. These plans contain comprehensive details, including the duration of green, yellow, and red phases for each signalized intersection. However, the operational downtime experienced by the Insight Platform during thesis period posed a significant challenge. This downtime impeded the ability to consistently verify and validate the timing plans throughout the study. Consequently, reliance on previously recorded signal timing data was imperative to ensure the accuracy of the simulation models. Proactive measures were taken to cross-reference the available data with field observations of intersections with similar background. This approach aimed to mitigate the impact of downtime on the overall accuracy and reliability of the thesis' findings. The signal timing plans were then incorporated into the VISSIM Traffic Signal Controller Program.

#### 3.1.2.4 Data Generation

The VISSIM model utilizes a range of network objects, such as data collection points, vehicle travel times, and queue counters, to collect data on outputs like vehicle speeds, travel time, delay time, and queue lengths. These outputs are crucial for simulating real-world measurements and determining the necessity for model calibration.

The placement of VISSIM network objects, including data collection points and vehicle travel times, mirrors the locations used in the Google Matrix API, ensuring consistency. The simulation encompasses a period of 3600 seconds in addition to a warm-up phase of 900 seconds representing the 1-hour run of extracted trajectory from the LiDAR sensors and 15 minutes of warm-up period prior to recording. The simulation also includes 30 simulation runs and 30 random seeds, effectively representing a month of variation.

This thesis focuses on gathering average travel time and speed in both eastbound and westbound directions of the Boulevard du Saint Maurice corridor to assess the model's performance during the initial calibration process. Given Boulevard du Saint Maurice corridor is characterized as a major road, it serves as a benchmark for comparing simulated travel time and speed with field measurements and conducting the necessary calibration. Further information on the travel time and speed calibration and validation processes are detailed in Section 3.2 and Section 3.3. Subsequently, the trajectory files generated by VISSIM are inputted into SSAM, a conflict-detection software that calculates various critical traffic conflict indicators, including PET. Additional details on conflict calibration and validation processes are outlined in Section 3.4 to Section 3.6.

Table 3 provides a concise summary of the calibrated parameters for each of the three calibrated simulation models. Each parameter is listed in the rows, with corresponding checkmarks (✓) whether it was calibrated during performance, performance and safety, and only safety calibrations.

**Table 3 Checklist of Calibrated Parameters by each Model Stage**

| Parameter                               | Performance | Performance and Safety | Only Safety |
|-----------------------------------------|-------------|------------------------|-------------|
| Standstill Distance                     | ✓           | ✓                      | ✓           |
| Minimum Headway                         | ✓           | ✓                      | ✓           |
| Additive parts of safety distance       | ✓           | ✓                      | -           |
| Multiplicative parts of safety distance | ✓           | ✓                      | -           |
| Temporary lack of attention duration    | -           | ✓                      | ✓           |
| Temporary lack of attention probability | -           | ✓                      | ✓           |
| Desired Position at free flow           | -           | ✓                      | ✓           |
| Safety Distance Reduction Factor        | ✓           | ✓                      | -           |
| Accepted Deceleration (Lane Changing)   | ✓           | ✓                      | -           |
| Accepted Deceleration (Following)       | ✓           | ✓                      | -           |
| Maximum Deceleration (Lane Changing)    | ✓           | ✓                      | -           |
| Maximum Deceleration (Following)        | ✓           | ✓                      | -           |

### 3.2 Performance Model Calibration

In the initial phase of calibration, the VISSIM model is calibrated using data obtained from the LiDAR API to model real-world travel time and speed observations. Thus, the initial calibration involves adjusting model parameters encompassing driving behaviour and vehicle performance. Figure 12 displays the parameters extracted from the processed LiDAR API, highlighting the unobservable parameters in red. While the default VISSIM parameters offer a basic starting point for guiding the simulation model's behaviour regarding vehicle interaction, adjustments are necessary to ensure real-world accuracy.

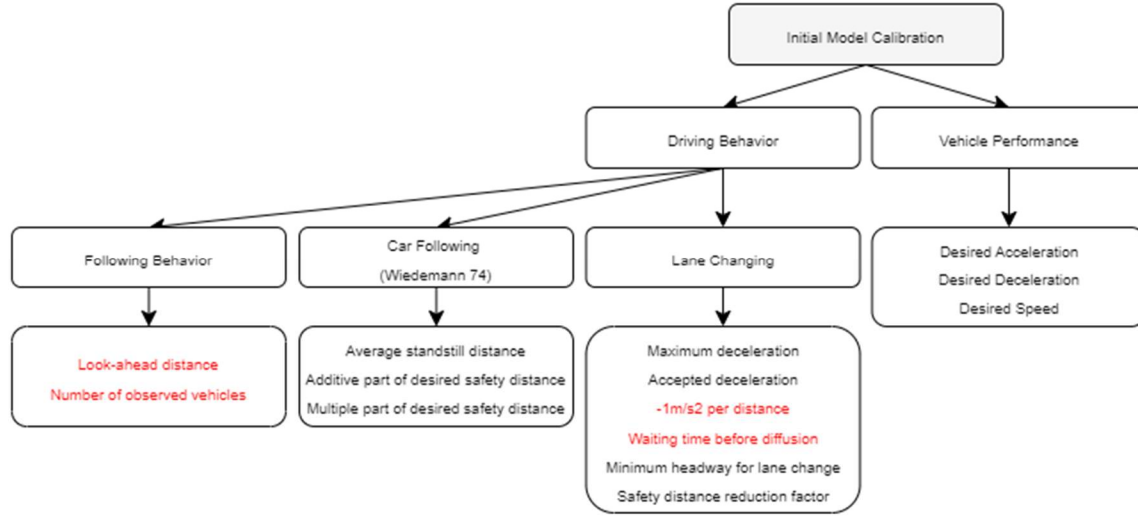


**Figure 12 VISSIM Parameters**

### 3.2.1 Driving Behavior

#### 3.2.1.1 Car-Following

The car-following model aims to capture the dynamics of vehicle interactions on the road. It was originally developed by Wiedemann in 1974 with adjustable parameters defining the thresholds for five states as shown in Figure 13. The concept behind the car-following model is to mimic the driver's reaction in the following vehicle. For instance, when the following vehicle approaches a slower-speed leading vehicle, it decelerates to avoid a potential crash. In VISSIM, the user can adjust two thresholds as shown in Figure 13: the distance between two stationary vehicles (known as standstill distance) represented as  $ax$ , and the minimum following distance contributing to a safe driving distance which is represented as  $d$ . After defining those two thresholds using processed LiDAR API, the additive and multiplicative parts of desired safety distance,  $bx$ , can be calculated through a linear programming formulation (Lei, 2021).

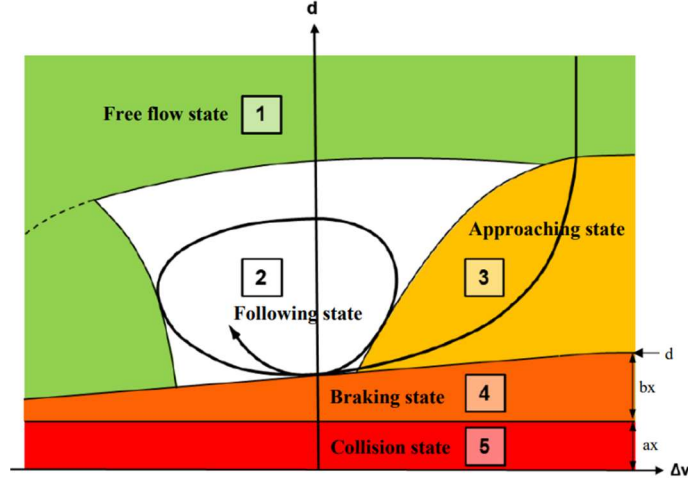


Figure 13 Car-Following Model

#### 3.2.1.1.1 Average Standstill Distance

Standstill distance is the measured distance between the rear bumper of the leading vehicle and the front bumper of the following vehicle when both vehicles are stationary. It is calculated using LiDAR API trajectory data by identifying the closest stationary vehicle to each vehicle based on geocoordinates and timeframe, then subtracting the vehicle size from the measured distance. Negative standstill distances may occur due to switched placements of vehicles or inaccuracies in measured distance if vehicle IDs change along the traveling path. During the AM peak, approximately 1 in 3.4 entries had a negative standstill distance, while during the PM peak, about 1 in 2.5 entries were invalid. These outliers along with distances exceeding 10 metres, were cleaned from the dataset to ensure data integrity.

#### 3.2.1.1.2 Desired Safety Distance

The desired safety distance is defined as the minimum safety following distance maintained by drivers. To calculate the desired safety distance, the length of leading vehicle is subtracted from the distance headway, as demonstrated in Equation 3.1. The distance headway can be obtained from the processed LiDAR API by specifying the distance gap in metres between the centers of the leading and following vehicles. Afterwards, the 10<sup>th</sup> percentile value of the following distances was measured in keeping with literature with similar study environments (Lei, 2021).

$$\text{Minimum Safety Following Distance} = \text{Space Headway} - \text{Leading Vehicle} \quad (3.1)$$

#### 3.2.1.1.3 Additive and Multiplicative Parts of Desired Safety Distance

The calculation of  $bx$  is determined using Equations 3.2 and 3.3 which establish the relationship between average standstill distance, desired safety distance, the additive and multiplicative parts of the safety distance, and the speed of the following vehicle. Equation 3.3 is then transformed into Equation 3.4 by rearranging the known values. In this context,  $d$  represents the desired safety distance in metres,  $ax$  is the standstill distance,  $bx_{add}$  is the

additive part of safety distance,  $b_{mult}$  is the multiplicative part of safety distance,  $z$  represents a randomly selected value between 0 and 1 that a truncated normally distributed around 0.5 with a standard deviation of 0.15, and  $v$  represents the following vehicle speed in metres per second.

$$d = ax + bx \quad (3.2)$$

$$bx = (bx_{add} + bx_{mult} \times z)\sqrt{v} \quad (3.3)$$

$$\frac{bx}{\sqrt{v}} = (bx_{add} + bx_{mult} \times z) \quad (3.4)$$

The right-side of Equation 3.4 follows a normal distribution with mean of  $bx_{add} \pm 0.5bx_{mult}$ . According to a study by Lei (2021), the most effective method to determine  $bx_{add}$  and  $b_{mult}$  is by deriving Equation 3.5 as the standard deviation and then minimizing it as the objective function. Additionally, the parameter ranges should be adjusted to accurately reflect field speed data and estimated saturation flow rates (Park & Qi, 2005). Specifically, the coefficient  $bx_{add}$  should range between 1.0 to 5.0, while  $b_{mult}$  should range between 1.0 to 6.0.

Minimize:

$$\sigma \frac{bx}{\sqrt{v}} = 0.15bx_{mult} \quad (3.5)$$

Subject to:

$$\mu \frac{bx}{\sqrt{v}} = bx_{add} + 0.5bx_{mult} \quad (3.6)$$

$$1 \leq bx_{add} \leq 5 \quad (3.7)$$

$$1 \leq bx_{mult} \leq 6 \quad (3.8)$$

### 3.2.1.2 Lane Changing

The lane change model in VISSIM is classified into two primary situations: free lane change and necessary lane change (Mohamed et al., 2021). Each situation undergoes distinct evaluation within the model. For free lane change, the model evaluates the time gap between the leading vehicle and following in meeting the desired safety distance and minimum time headway criterion. On the other hand, for necessary lane changes, the model assesses the vehicle queue while considering the lead time to collision between the leading and following vehicles in a certain lane.

This research aims to calibrate the lane change model using data obtained from LiDAR sensors, with attention to lane-changing movements in general. Initially, the LiDAR sensors employed are unable to detect or visualize lane edges and limits. Consequently, a Python script was developed to identify travel lanes using geographic coordinates from Google Maps as reference points for lane limits. This approach enables

determining the vehicle's position, orientation, and whether a lane change has occurred. Notably as well, LiDAR sensors are limited in their ability to distinguish between different types of lane changes, which falls into necessary lane change caused by traffic conditions and free lane change made at the discretion of the driver.

Studies on lane change duration, such as those conducted by Salvucci & Liu (2002) and Lee et al. (2004), provide valuable insights into the temporal dynamics of driver behaviour during lane changes. Salvucci & Liu (2002) reported an average lane change duration of 5.14 seconds based on driving simulator experiments examining lane changing control and eye movement patterns. Meanwhile, Lee et al. (2004) reported 6.3 seconds from real-world observations of sixteen sampled commuters, with a range varying from 3.5 seconds to 6.5 seconds on urban streets. These benchmarks are beneficial in defining expected ranges when measuring time headway, acceleration, and deceleration of lane changing vehicles from collected LiDAR data.

The calibration process applied these insights by comparing with observed travel times to identify valid lane change events and filter out irrelevant movements. For instance, movements that deviated significantly, such as extended lane changing acceleration, were flagged as noise behaviors. Thereby, the process aims to improve the ability to replicate traffic conflicts.

#### 3.2.1.2.1 Maximum and Accepted Deceleration

The maximum and accepted deceleration establish the upper and lower bounds of deceleration for lane-changing and following vehicles. These bounds determine the required distance between the two vehicles to prevent potential collisions.

Establishing a normal distribution for deceleration rates for both lane-changing and following vehicles involves capturing the 85<sup>th</sup> percentile for maximum deceleration and the 15<sup>th</sup> percentile for accepted deceleration within the majority range. Vehicles within this deceleration range are more likely to be changing lanes freely rather than adhering to safe driving practices. According to Lei (2021), setting the 85<sup>th</sup> percentile for maximum deceleration rate and the 15<sup>th</sup> percentile for accepted deceleration rate results in the lowest error rate between simulated and observed data. Hence, similar percentiles are utilized for this thesis since similar corridor conditions are observed.

#### 3.2.1.2.2 Minimum Headway

The term "minimum headway" pertains to the required minimum distance between two vehicles following a lane change for a safely completed manoeuvre while maintaining a speed-dependent safety distance. To determine the minimum headway, the distance between a vehicle that has changed lanes, and the following vehicle was measured. Specifically, for this thesis, the minimum headway was derived from the processed LiDAR API by averaging the most frequent and lowest headway bin from a distribution histogram.

### 3.2.1.2.3 Safety Distance Reduction Factor

The lane change model employs a safety distance reduction factor as a multiplier that reduces the minimum safety distance required between lane-changing and following vehicles. The safety reduction factor was calculated by finding the average ratio of minimum and maximum distance headways between lane-changing and following vehicles. Only vehicle records with a maximum distance headway of less than 10 metres have been included in this calculation to align with the distribution vehicular following distances observed from the processed LiDAR API.

## 3.2.2 Vehicle Performance

### 3.2.2.1 Desired Speed

The desired speed is the travelled speed affected by road design and driving aggressiveness level. In VISSIM, the default desired speed is represented by a uniform distribution, visually depicted as a straight-line cumulative distribution in Figure 14, based on a posted speed of 50 km/h. It is crucial to recognize external factors such as the composition of roadway cross-section (i.e., on-street parking, bike lanes, sidewalks, landscaping, and utility poles) and interactions with other drivers can impact the desired speed.

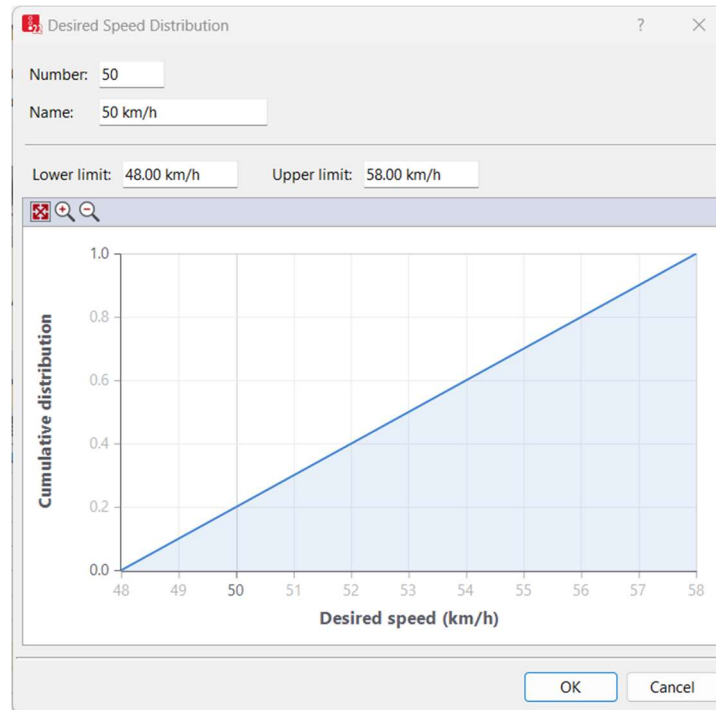


**Figure 14 VISSIM Default Desired Speed Distribution for 50 km/h**

The desired speed is influenced by vehicles in a free-flow state, occurring at a time headway of 6 seconds or greater (Boora et al., 2017). To modify the desired speed using LiDAR data, the processed API focused on vehicle trajectories with a 6-second time

headway threshold. A minimum speed threshold was also applied to filter out slowly travelling vehicles.

To specify the minimum desired speed, Mean Absolute Normalized Error (MANE) as shown in Equation 3.9 was used to calculate observed and simulated speed values travelled from the first to the last intersection of corridor. The speed with the lowest error was selected as the minimum desired speed. As for the maximum desired speed, it was determined based on best fit of the desired speed cumulative distribution with the highest fit based on R-square.

$$\text{Minimize MANE (v)} = \frac{1}{J} \sum_{j=1}^J \frac{|v_{obs,j} - v_{sim,j}|}{v_{obs,j}} \quad (3.9)$$

Whereas,

$v_{obs,j}$  is observed average speed for a given section  $j$  (m/s).

$v_{sim,j}$  is simulated average speed for a given section  $j$  (m/s).

$J$  is total number of speed collection sections.

### 3.2.2.2 Desired Acceleration/Deceleration

In the VISSIM simulation model, the desired acceleration and deceleration are functions of the vehicle's speed. Default values are set for minimum, median, and maximum desired acceleration and deceleration to accommodate different driving behaviours and vehicle characteristics. To determine the input values for minimum, median, and maximum desired acceleration and deceleration, multiple percentiles were experimented through simulation runs with the goal of finding lowest MANE. The minimum threshold corresponded to the 5<sup>th</sup>, 10<sup>th</sup>, and 15<sup>th</sup> percentiles; the median threshold was the 50<sup>th</sup> percentile; and the maximum threshold was the 85<sup>th</sup>, 90<sup>th</sup>, and 95<sup>th</sup> percentiles. In this thesis, nine test runs were conducted using combinations of percentiles for desired acceleration and deceleration, as shown in Table 4. The evaluation was done by comparing the observed and simulated acceleration and deceleration using MANE.

**Table 4 Percentile Combinations for Desired Acceleration/Deceleration**

| Simulation<br>Test # | Acceleration     |                  |                  | Deceleration     |                  |                  |
|----------------------|------------------|------------------|------------------|------------------|------------------|------------------|
|                      | Minimum          | Median           | Maximum          | Minimum          | Median           | Maximum          |
| 1                    | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> |
| 2                    | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> |
| 3                    | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> |
| 4                    | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> |
| 5                    | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> |
| 6                    | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> |
| 7                    | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> |
| 8                    | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> |
| 9                    | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> |

To ensure accurate results, acceleration and deceleration within lower ranges were excluded. This step was necessary to prevent these lower values from influencing the overall relationship between speed and acceleration/deceleration. The recommended cut-off points should be within the range of 0 to 1 m/s<sup>2</sup> (0 to -1 m/s<sup>2</sup> for deceleration) across each speed range. The specific cut-off points in this thesis are presented in Table 5.

**Table 5 Cut-off Points for Desired Acceleration/Deceleration Calculation**

| <b>Speed Interval (km/h)</b> | <b>Acceleration Thresholds (m/s<sup>2</sup>)</b> | <b>Deceleration Thresholds (m/s<sup>2</sup>)</b> |
|------------------------------|--------------------------------------------------|--------------------------------------------------|
| <b>0-10</b>                  | 0.1                                              | -0.5                                             |
| <b>10-20</b>                 | 0.1                                              | -0.5                                             |
| <b>20-30</b>                 | 0.1                                              | -1.0                                             |
| <b>30-40</b>                 | 0.1                                              | -1.0                                             |
| <b>40-50</b>                 | 1.0                                              | -1.0                                             |

### 3.2.2.3 Maximum Acceleration/Deceleration

The various vehicle types come with specific maximum and desired acceleration and deceleration values based on their category. This is essential for accurately representing their behaviour in a traffic simulation. For instance, heavy trucks and passenger vehicles have unique acceleration and deceleration traits that must be considered to achieve realistic simulation results. Hence, maximum acceleration and deceleration distributions were required to be adjusted to mirror the distinct performance capabilities and limitations of passenger vehicle category.

To ensure accuracy, the desired acceleration and deceleration values were replicated for use as maximum values, ensuring consistency with real-world data. The average maximum acceleration and deceleration values were determined based on the default relationship between desired and maximum parameters, allowing for refinement with processed LiDAR API data. These values were adjusted for different speed intervals.

## 3.3 Performance Model Validation

The Google Distance Matrix API is a tool for analyzing travel time and validating estimates from the microsimulation model for various modes of transportation, including driving, biking, transit, and walking. By entering longitude and latitude coordinates using programming languages such as Java, Python, Go Client, or Node.js, users can retrieve travel time information in JSON or XML format. To validate the calibrated simulation model based on travel time, the Google Distance Matrix API is utilized to gather observed travel times during specific peak hours. This involves collecting travel time data from 8:00 am to 9:00 am and 3:30 pm to 4:30 pm at 5-minute intervals over the week of July 9th, 2022, to July 15th, 2022.

Observed speed data is obtained from the LiDAR Insight Platform in 1-hour increments for each intersection during the same period. Due to differences in data collection increments and the inability to measure the entire corridor on the Insight Platform, adjustments are made by distributing recorded speeds within the first and second 30 minutes of peak hours. For instance, the speed at 8:00 am is duplicated every 5 minutes until 8:30 am, and the speed at 9:00 am is duplicated from 8:30 am to 9:00 am. Additionally, observed speed data is calculated using the distance and travel time obtained for each vehicle ID from the LiDAR API.

Since the observed data was collected over a week, the simulated speed and travel time - originally spanning 30 days - are averaged to fit a 7-day week. The averaging process involved grouping the simulated values for each corresponding day. For instance, if a Monday occurred four times during the simulation period, the simulated speed and travel time values were averaged across four runs. For speed, the MANE is calculated by comparing simulated and observed values at each intersection. In the case of travel time, the MANE is calculated by comparing simulated corridor-wide travel time with those recorded by the Google Matrix API. The total MANE is then obtained by summing individual MANE values for both speed and travel time.

Travel time and speed distribution figures are created to validate the initial calibration model and compare the data from different sources with the simulated data. For instance, speed is calculated from the Google API by dividing the travel time by the corridor length of 860 metres, using geocoordinates set up in the Google API script. The LiDAR API is also used to extract travel time and speed data for 1-week, with data cleaned to include only vehicle IDs traveling the entire corridor. These distribution figures are created for the whole corridor over a 1-week duration, comparing the initially calibrated model with the default model, Google Matrix API, and LiDAR API.

### **3.4 Parameter Selection**

The parameter selection covered in this methodology aims to determine the VISSIM parameters effecting simulated traffic conflicts. The proposed process involves two key steps: literature review (Section 3.4.1) and sensitivity analysis (Section 3.4.2).

#### **3.4.1 Effective Parameters by Literature Review**

The initial step entails conducting a comprehensive review of existing studies that have specifically addressed the adjustment of parameters for conflicts in microsimulation models. This process is crucial for eliminating certain parameters based on engineering judgment and experience documented in the literature. By analyzing commonly studied parameters and impacted conflict types, the aim is to establish the foundation for selecting the most influential parameters. Table 6 contains a comprehensive list of parameters previously investigated for calibrated simulated conflicts, which will be the primary focus of the next step in this thesis as detailed in Section 3.4.2.

**Table 6 Summary of Conflicts-Effective Parameters based on Literature**

| Parameter                               | Reference                                                                                                               | Covered in Initial Calibration |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| Standstill Distance                     | (Fan et al., 2013a), (Essa & Sayed, 2015a), (Khavas et al., 2017), (Essa & Sayed, 2020a), (Liang et al., 2020)          | Yes                            |
| Minimum Headway                         | (Abuamer et al., 2019), (Brill & Brink, 2016), (Foomani, 2022), (Liang et al., 2020), (Ulak et al., 2019)               | Yes                            |
| Headway Time                            | (Fan et al., 2013a), (Essa & Sayed, 2015a), (Khavas et al., 2017), (Essa & Sayed, 2020a), (Liang et al., 2020)          | No                             |
| Threshold for entering following state  | (Fan et al., 2013a), (Essa & Sayed, 2015a), (Khavas et al., 2017), (Essa & Sayed, 2020a), (Liang et al., 2020)          | No                             |
| Safety Distance Reduction Factor        | (Fan et al., 2013a), (Essa & Sayed, 2015a), (Khavas et al., 2017), (Essa & Sayed, 2020a)                                | Yes                            |
| Start upstream of stop line             | (Essa & Sayed, 2015a), (Essa & Sayed, 2020a)                                                                            | No                             |
| Desired Deceleration                    | (Essa & Sayed, 2015a), (Essa & Sayed, 2020a)                                                                            | Yes                            |
| Acceleration (Leading)                  | (Khavas et al., 2017)                                                                                                   | No                             |
| Acceleration (Following)                | (Khavas et al., 2017)                                                                                                   | No                             |
| Accepted Deceleration (Lane Changing)   | (Khavas et al., 2017)                                                                                                   | Yes                            |
| Accepted Deceleration (Following)       | (Khavas et al., 2017)                                                                                                   | Yes                            |
| Look-ahead distance (Maximum)           | (Liang et al., 2020), (Foomani, 2022), (Ulak et al., 2019), (Abuamer et al., 2019), (Deluka Tibljaš et al., 2018)       | No                             |
| Look-ahead distance (Observed)          | (Liang et al., 2020)                                                                                                    | No                             |
| Temporary lack of attention duration    | (Liang et al., 2020), (Ulak et al., 2019), (Brill & Brink, 2016), (Abuamer et al., 2019), (Deluka Tibljaš et al., 2018) | No                             |
| Temporary lack of attention probability | (Liang et al., 2020), (Ulak et al., 2019), (Brill & Brink, 2016), (Abuamer et al., 2019), (Deluka Tibljaš et al., 2018) | No                             |
| Standstill acceleration                 | (Liang et al., 2020)                                                                                                    | No                             |
| General Behaviour                       | (Liang et al., 2020)                                                                                                    | No                             |
| Desired position at free flow           | (Liang et al., 2020)                                                                                                    | No                             |

### 3.4.2 Sensitivity Analysis

Microscopic simulation models are intricate, leading to unique traffic network performance. Therefore, it is crucial to conduct sensitivity analysis to identify parameters affecting the model's outputs and discard unnecessary ones. The following steps are followed to perform sensitivity analysis in this thesis:

1. *Define the range of VISSIM parameters*, as specified in Table 7.
2. *Define the parameterization method*, which is determined to be one-way ANOVA test, where a selected parameter is modified while others remain at

default values. The modified values are based on the minimum and maximum values from the literature review, with a median value tested if the parameter proves ineffective.

3. *Run the simulation model* for each pair of parameters listed in Table 8, modifying each parameter within the pair for every run. The parameter pairings are chosen based on anticipated interactions, reflecting how one parameter may influence the other as inferred from their descriptions and typical behaviour.
4. *Analyze simulation results* for each modified selected parameter in terms of simulated PET.
5. *Identify the critical parameters*, by assessing if the outputted conflict rate is statistically significant compared to a simulation run with a default parametric value, using *p-values*. If the *p-value* is less than 0.05, then the parameter is considered statistically significant.

Each parameter was tested across three threshold scenarios, with 5 simulation runs per threshold to manage trajectory file size limitations in SSAM efficiently. A VISSIM COM script was developed to automate the calibration process, enabling parameter modification and simulation execution. This automation enables a reliable evaluation of parameter combinations while optimizing time management.

**Table 7 Sensitivity Analysis Tested Ranges**

| Parameter                               | Default        | Tested Thresholds |
|-----------------------------------------|----------------|-------------------|
| Standstill Distance                     | 1.5            | 0, 2.7, 10        |
| Minimum Headway                         | 0.5            | 0.1, 5, 500       |
| Safety Distance Reduction Factor        | 0.6            | 0.1, 2.55, 5      |
| Accepted Deceleration (Lane Changing)   | -1.0           | -0.01, -4, -10    |
| Accepted Deceleration (Following)       | -1.0           | -0.01, -4, -10    |
| Look-ahead distance (Maximum)           | 150            | 50, 100, 300      |
| Look-ahead distance (Minimum)           | 0              | 10, 30, 50        |
| Temporary lack of attention duration    | 0              | 0.1, 2.5, 5       |
| Temporary lack of attention probability | 0              | 5, 15, 25         |
| Desired position at free flow           | Middle of lane | Any, Left, Right  |
| Lateral Distance Driving                | 1.0            | 0.3, 0.6, 0.8     |
| Lateral Distance Standstill             | 0.2            | 0.3, 0.8, 1.0     |
| Reaction Time Distribution              | 0              | 0.5, 0.9, 2.0     |

**Table 8 Sensitivity Analysis Parameters Combinations**

| #  | Conflict-Effective Parameter Combinations |                                         |
|----|-------------------------------------------|-----------------------------------------|
|    | Parameter A                               | Parameter B                             |
| 1  | Standstill Distance                       | Minimum Headway                         |
| 2  | Minimum Headway                           | Safety Distance Reduction Factor        |
| 3  | Acceleration/Deceleration (Leading)       | Acceleration/Deceleration (Following)   |
| 4  | Look-ahead Distance (Maximum)             | Temporary Lack of Attention Duration    |
| 5  | Look-ahead Distance (Maximum)             | Look-ahead Distance (Observed)          |
| 6  | Look-ahead Distance (Minimum)             | Temporary Lack of Attention Duration    |
| 7  | Temporary Lack of Attention Duration      | Temporary Lack of Attention Probability |
| 8  | Minimum Headway                           | Desired Position at Free Flow           |
| 9  | Lateral Distance Driving                  | Lateral Distance Standstill             |
| 10 | Standstill Distance                       | Lateral Distance Standstill             |
| 11 | Look-ahead Distance (Minimum)             | Reaction Time Distribution              |
| 12 | Temporary Lack of Attention Duration      | Reaction Time Distribution              |

### 3.5 Performance and Safety Model Calibration

In the second step of model calibration, a trial-and-error technique is utilized to estimate conflict-effective parameter values, aiming to match observed conflicts. Parameters are adjusted iteratively within predefined ranges and compared to actual conflicts. The process begins by simultaneously adjusting two parameters, identifying combinations that bring simulated conflicts closer to observed ones. Once these values are set, another parameter is adjusted, ensuring that unchanged conflicts are effectively addressed. Each adjustment involves conducting 30 simulation runs. A VISSIM COM script was developed to automate parameter adjustments and test combinations efficiently. The resulting trajectory files from each adjustment are rigorously analyzed using SSAM to extract simulated conflicts. This analysis provides comprehensive conflict information, including vehicle position, conflict type, PET, as well as summaries of total conflicts, maximum PET, and average PET.

It is crucial to note that initial observations of conflicts from the LiDAR Insight Platform did not include rear-end and lane-change conflicts. This information was provided through CSV files containing vehicle movements during conflicts and recorded videos for PET less than 2 seconds. Based on the CSV files, there were conflicts initially labeled as crossing conflicts; however, a thorough investigation revealed that many of these conflicts

occurred along the approach leg once referencing geocoordinates, suggesting the presence of rear-end or lane-change conflicts. Additionally, challenges were encountered in using recorded videos of conflicts. The focus on PET of less than 2 seconds limited the range of conflicts studied, with videos capturing crossing conflicts and often overlooking rear-end conflicts. Moreover, overlapping conflicts were typically classified only as crossing conflicts, emphasizing the need for more comprehensive approaches to accurately identify and classify conflict types.

To differentiate between rear-end and lane-change conflicts from crossing conflicts, a Python code was developed in Google Collaboratory based on the position of the conflicts relative to pavement marking edges. Conflicts occurring within 0.75 metres from the lane edge were initially categorized as lane-change conflicts. However, to ensure more accurate results, the counts of rear-end and lane-change conflicts were combined into a single category, longitudinal conflicts. The adjustment of merging two types of conflicts into one, is made to address potential misclassifications that could arise from the limited view provided by LiDAR Insight Platform and the absence of specific labeling for non-crossing conflicts in the CSV files.

#### 3.5.1 Standstill Distance

The standstill distance, initially calibrated in Section 3.2.1.1.1 for network performance, is reused for the performance and safety calibration when focusing on conflicts. Liang et al. (2020) used values ranging from 0 metres for low magnitude to 5 metres for high magnitude. Foomani (2022) applied a range of 0 to 5 metres with 0.5-meter increments, while Ulak et al. (2019) specified 2 metres for mid-aged and senior drivers and 1.5 metres for young drivers. Essa & Sayed (2020) used standstill distances of 2.10 metres and 2.5 metres at different intersections, and Abuamer et al. (2019) used a range from 0.75 metres for low to 2.25 metres for high. To determine the calibrated value for standstill distance in this thesis, an increment of 0.18 metres is tested, along with its corresponding parameters, which had to pass the effectiveness check on conflicts from the outlined combinations in Table 8, each with its own specified increment. The proposed increment for standstill distance, as well as those covered in the following sections, is chosen to balance manageable simulation runs along with literature-based increments.

#### 3.5.2 Minimum Headway

The minimum headway was established during the initial calibration phase, as detailed in Section 3.2.1.2.2, and is calibrated again in the second calibration. This parameter also offers an alternative to the commonly used Headway Time (Gap Time) because the study employs the Wiedemann 74 model, which is best suited for urbanized intersections, as opposed to the previously used Wiedemann 99 model. To determine the calibrated value, Liang et al. (2020) used increments of 0.3 metres for low, 0.8 metres for medium, and 1.5 metres for high headway levels. Foomani (2022) tested a range from 0.1 metres to 5 metres with an increment of 0.1 metres. Brill & Brink (2016) explored a range

from 3 metres to 7 metres with 2 metres increments. Ulak et al. (2019) used 1.07 metres for mid-aged drivers, 0.76 metres for young drivers, and 1.83 metres for aging drivers. The range used by Abuamer et al. (2019) was from 0.5 metres to 1 metre. For this thesis, an increment of 0.32 metres is applied within the established effective range, and in combination with corresponding effective parameter from Table 8.

### 3.5.3 Look-Ahead Distance (Maximum/Minimum)

The minimum/maximum look-ahead distance refers to the closest/farthest distance a vehicle driver can see ahead to anticipate and react to other vehicles in front or to the sizes within the same link.

The calibration process to determine the maximum look-ahead distance entails a systematic trial-and-error approach to determine the value within specified thresholds. Following the methodology used by Lokesh et al. (2020), the interval employed is 25 metres, ranging from 175 up to 300 metres; Foomani (2022) used a range of 100 to 300 metres with an 8-metre interval; Liang et al. (2020) used magnitude levels of 200 metres for low, 300 metres for medium and 500 metres for high; while Ulak et al. (2019) used 91 metres for young to mid-aged drivers and 61 metres for seniors. For this thesis, the effective range based on sensitivity analysis thresholds, inspired by literature, was used to calibrate look-ahead distance. An increment of 17 metres, in combination with other effective parameters as outlined in Table 8, is employed within the determined effective range.

### 3.5.4 Temporary Lack of Attention (Duration/Probability)

Temporary lack of attention in microsimulation refers to a parameter that simulates a driver's temporary inattentiveness or distraction while driving. It represents a duration of time where a driver will not react to the preceding vehicle, except for emergency braking. During this "lack of attention" period, the driver ignores the vehicle in front and does not adjust their speed or the following distance accordingly. The temporary lack of attention parameter aims to replicate realistic driver behavior where momentary distractions or lapses in attention occur. Increasing the duration and probability values makes drivers less responsive to changes in the leading vehicle, potentially causing shockwaves or instabilities in traffic flow.

Through a study conducted by CDM Smith (2014), an investigation on the reliance of temporary lack of attention on the classification of the environment was pursued. In urban settings, the default duration is set at 0.5 seconds, while aggressive and mild urban driving scenarios result in durations of 0 seconds. Correspondingly, the probability of encountering a temporary lack of attention is 10 percent for the urban default case but dropped to 0 percent for aggressive and mild urban conditions. Liang et al. (2020) used a 1 percent probability for low magnitude, 3 percent for medium, and 5 percent for high magnitude. For the temporary lack of attention duration, Liang et al. (2020) used 1 second for low magnitude, 2 seconds for medium magnitude, and 4 seconds for high magnitude.

Ulak et al. (2019) reported different values based on age groups, using a 5 percent probability for mid-aged drivers, 25 percent for young drivers, and 20 percent for seniors. The durations were 1 second for mid-age, 2 seconds for young, and 1.5 seconds for senior drivers. Brill & Brink (2016) used a 5 percent probability and 1-second duration. Abuamer et al. (2019) utilized a range of 0 to 10 percent for probability and 0 to 2.5 seconds for duration. Deluka Tibljaš et al. (2018) used 8 percent probability and 1 second duration. For this thesis, increments of 0.5 percent for temporary lack of attention probability and 0.16 seconds for temporary lack of attention duration, combined with other effective parameters as outlined in Table 8, are used to calibrate within the conflict-effective range.

#### **3.5.5 Desired Position at Free Flow**

In traffic simulation models, the desired position at free flow parameter refers to the preferred lateral orientation of vehicle within its lane when not influenced by other vehicles, which occurs under free flow conditions. This thesis focused on adjusting the effective position parameter and comparing the model's output in terms simulated conflicts. A maximum of 3 calibration runs are conducted, utilizing the available positions provided by VISSIM and combined with corresponding effective parameter in Table 8.

### **3.6 Only Safety Model Calibration**

The only safety calibration uses conflict-based parameters calibrated in Section 3.5 to focus on safety outcomes, while other parameters are set to default except if those directly related to network building for the case study. It also incorporates the desired speed and acceleration/deceleration distributions from the performance calibration in Section 3.2 to represent the observed traffic flow. This model type is to evaluate solely the impact of safety parameters against performance and safety model per Section 3.5.

### **3.7 Conflicts Validation**

The validation of modelled conflicts focus on comparing simulated conflicts as outputted from each of the performance, performance and safety, and only safety models, against the observed data over a one-month period during the peak hours. The essence of this step is to evaluate the accuracy of the calibrated model in replicating real-world conflict scenarios. The validation process included analysis of several key aspects:

- Examination of the distribution of PET for each intersection, considering combined conflict types.
- Separate analysis of PET distribution of individual conflict types for each intersection, including crossing and longitudinal conflicts, allowing for an understanding of the PET distribution for different conflict types at each intersection.
- Study of the distribution of conflict types for each intersection to understand the range of conflict types observed during the validation period.

- Generation of combined conflict heat maps for the entire corridor, providing a visual representation of conflict hotspots and patterns along the corridor.

Overall, the validation process aimed to assess the effectiveness of conflict-focused calibration in accurately representing real-world conflict scenarios in the traffic simulation model by comparing simulated conflicts with observed data over a one-month period and analyzing PET distributions, conflict types, and heat maps.

### 3.8 Effectiveness of Calibration

#### 3.8.1 Error Degree

To evaluate improvements made to the calibrated model, the process involved comparing all simulation models, including the default, performance calibrated, performance and safety calibrated, and only safety calibrated, with observed speed and travel time data as the baseline. The performance calibration focused on refining operational parameters to better match the observed operational data. The performance and safety calibration integrated both operational and safety factors to create a more balanced model. The only safety calibration prioritized safety by incorporating conflict-based parameters. The effectiveness of each calibration was measured by calculating the MANE based on the observed speed and travel time, allowing for a clear comparison of improvements across the different models.

#### 3.8.2 Effectiveness Rate

The analysis assessed the impact of each calibrated parameter by comparing the key performance metrics – speed and travel time for operational, and average PET for safety – against results obtained using the default value of the same parameter. Each comparison was based on five simulation runs. The calculations were automated using a developed Python code on Google Collaboratory. The code computed the effectiveness rate for each calibrated parameter using Equation 3.10 for operational assessment and Equation 3.11 for safety assessment.

$$\text{Effectiveness Rate}_{\text{Operational}} = \frac{|t_{\text{calib}} - t_{\text{default}}|}{t_{\text{default}}} + \frac{|v_{\text{calib}} - v_{\text{default}}|}{v_{\text{default}}} \times 100 \quad (3.10)$$

Whereas,

$t_{\text{calib}}$  is calibrated average travel time across five simulation runs (s).

$t_{\text{default}}$  is default average travel time across five simulation runs (s).

$v_{\text{calib}}$  is calibrated speed across five simulation runs (m/s).

$v_{\text{default}}$  is default speed across five simulation runs (m/s).

$$\text{Effectiveness Rate}_{\text{Safety}} = \frac{|PET_{\text{calib}} - PET_{\text{default}}|}{PET_{\text{default}}} \times 100 \quad (3.11)$$

Whereas,

$PET_{\text{calib}}$  is calibrated average PET value across five simulation runs.

$PET_{\text{default}}$  is default average PET value across five simulation runs.

Additionally, the absolute difference between the default and calibrated values was calculated using Equation 3.12. Followed by calculating the percentage difference to quantify the magnitude of change relative to the default value, by using Equation 3.13.

$$\Delta_{\text{abs}} = |x_{\text{calib}} - x_{\text{default}}| \quad (3.12)$$

$$\Delta_{\text{percent}} = \frac{|x_{\text{calib}} - x_{\text{default}}|}{x_{\text{default}}} \times 100 \quad (3.13)$$

Whereas,

$\Delta_{\text{abs}}$  is the absolute difference.

$\Delta_{\text{percent}}$  is the percentage difference.

$x_{\text{calib}}$  is calibrated parameter value.

$x_{\text{default}}$  is default parameter value.

A similar approach was taken to assess the performance of the models rather than individual parameters. The results of travel time, speed, and average PET for each calibrated model were compared against one another and the default model to determine overall effectiveness.

### 3.8.3 Overall Assessment

The analysis of the simulation models involved a structured process to calculate and evaluate key metrics across the default model and the three calibrated versions. The focus is on both operational metrics including speed and travel time, as well as safety metric including conflict counts and PET. For each model and conflict type, the total number of conflicts was recorded to compare the predicts conflict frequency with observed data. The PET is analyzed by calculating the 85<sup>th</sup> percentile and variance for each conflict type to assess the severity and variability of conflicts. Additionally, the MANE for PET is computed, comparing model predictions against observed values to evaluate accuracy in simulating realistic traffic conditions.

## 4 Results

The purpose of this chapter is to present the results from a case study of applying the proposed methodology described in Section 3.

### 4.1 Field Data

The study area encompasses an 880-meter segment of Boulevard du Saint Maurice in Trois-Rivières, Québec. This urbanized collector section, situated within a mixed commercial and residential zone, includes six signalized intersections as shown in Figure 15: Rue Laviolette, Rue St Francois Xavier, Rue Ste Angele, Rue St Paul, Rue Nicolas Perrot, and Boulevard des Chenaux, all with a speed limit of 50 km/h. The area experiences moderate congestion during peak hours, making it an ideal candidate for assessing traffic safety and performance. Data collection was conducted during the morning peak hour from 8:00 am to 9:00 am and the afternoon peak hour from 3:30 pm to 4:30 pm over one week, from July 9th, 2022, to July 15th, 2022.



**Figure 15 Boulevard du Saint Maurice Aerial Image**

Raw CSV files exported from the LiDAR API were processed using Python in Google Collaboratory platform to cluster the data, visualize traffic trajectories and extract key metrics such as travel time, distance, and time headway. Each peak hour data collection was preceded by 15-minute warm-up period, ensuring complete vehicle trajectories and reducing the likelihood of erroneous movements. This warm-up period allowed the LiDAR sensors to reach thermal equilibrium, stabilizing their measurements and improving data accuracy (Sánchez & Pany, 2019). In the absence of such warm-up period, the LiDAR system could counter data inconsistencies or missing segments of vehicle trajectories, which could compromise the reliability of extracted travel time, acceleration, and deceleration patterns. Allowing the system to stabilize prior to recording can aid improving

data accuracy and ensure that the captured traffic patterns are representatives of typical real-world conditions. The subsequent 60-minute collection period covered the entire corridor, with 10 minutes allocated to recording data at each intersection daily, as the LiDAR API granted access to one intersection sensor at a time. along the corridor on each day of data collection.

The processed LiDAR API data provided essential attributes, including vehicle speed, acceleration, and deceleration, as well as space and time headways. However, database cleanup was necessary to rectify issues such as inconsistent timestamp formats, removal of trajectories with fewer than four recorded data points to retain meaningful trajectory information, and elimination of invalid vehicle IDs. The real-world trajectory data were gathered using LiDAR sensors offered comprehensive insights into vehicle movements and interactions, representing typical urban traffic patterns characterized by diverse road users. Table 9 lists the data types collected by LiDAR sensors and used for developing microscopic traffic simulation model.

**Table 9 LiDAR Sensor Data Description**

| Attributes       | Description                                                                                                                                                         |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>timestamp</b> | Date and time in standardized format, ISO 8601.                                                                                                                     |
| <b>id</b>        | Road user identification number.                                                                                                                                    |
| <b>centerX</b>   | Camera information of x-coordinate of road user that can be transformed to longitude.                                                                               |
| <b>centerY</b>   | Camera information of y-coordinate of road user that can be transformed to latitude.                                                                                |
| <b>width</b>     | Width of the bounding box detected for the object or road user.                                                                                                     |
| <b>length</b>    | Length of the bounding box detected for the object or road user.                                                                                                    |
| <b>rotation</b>  | Angle of the bounding box detected for the object or road user.                                                                                                     |
| <b>classType</b> | Road user type: pedestrian, car, van, truck, bus, SUV, construction vehicle, trailer, tractor, person sitting, cyclist, bicycle, person, motorcycles, rollerblader. |

Attributes such as vehicle speed, travel time, acceleration, time headway, and lane position were measured using extracted LiDAR data. These attributes were recorded across the entire corridor, including both northbound and southbound directions, as well as turns onto and off the corridor from intersecting roads. The measurement limits, based on the range of the LiDAR sensors, were determined from the centerline of the intersection: Rue Laviolette covers 60 metres, Rue St Francois Xavier covers 45 metres, Rue Ste Angele and Rue St Paul cover 30 metres, Rue Nicolas Perrot covers 55 metres, and Boulevard des Chenaux covers 50 metres. Figure 16 illustrates the locations of LiDAR sensors at each intersection as well as the covered area.

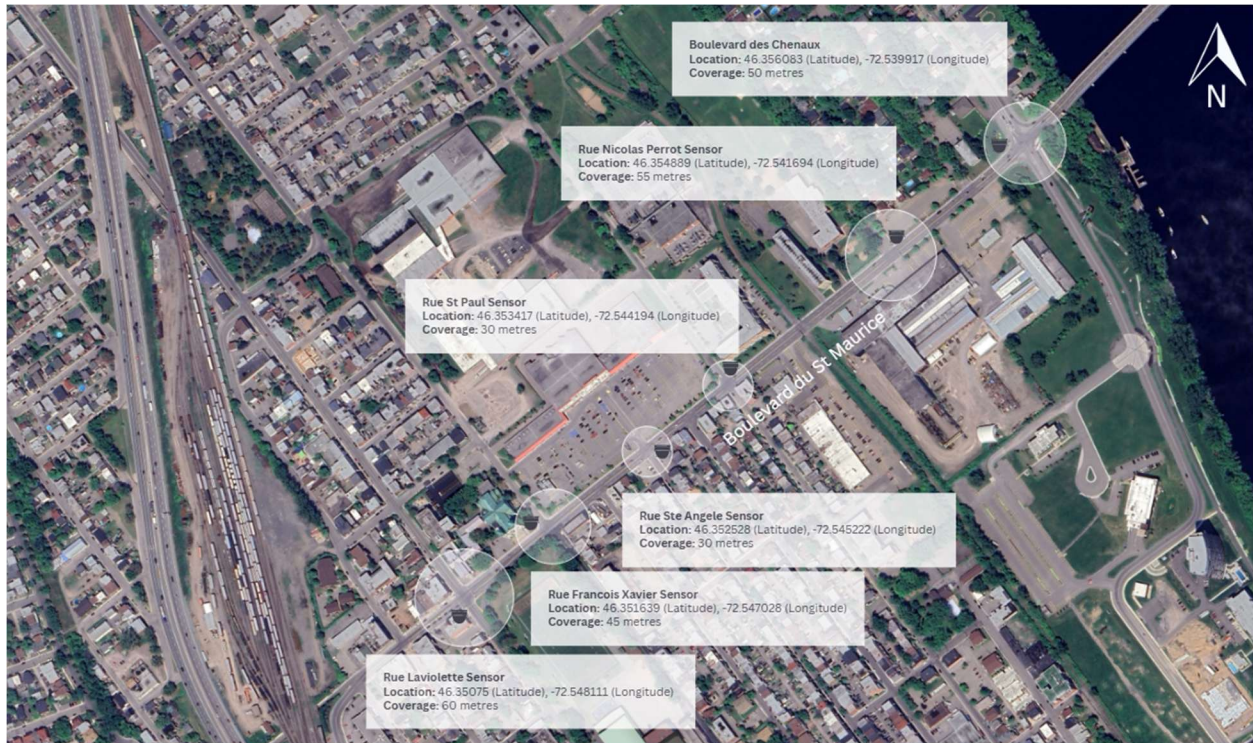


**Figure 16 LiDAR Locates along Boulevard du Saint Maurice**

Data was collected during both the morning and afternoon peak hours, with each hour representing one hour of daily data collection over the week from July 9<sup>th</sup> to July 15<sup>th</sup>, 2022. The distributions of these attributes are shown from Figure 17 to Figure 22, based on compiled data from the entire observed week as taken by the LiDAR sensors.

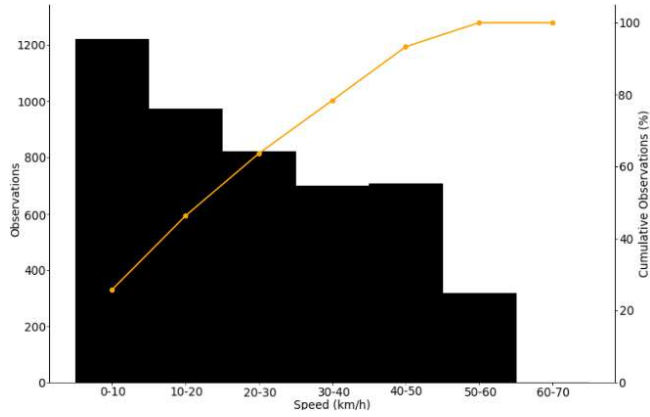


**Figure 17 AM Peak Vehicle Speed Distribution**

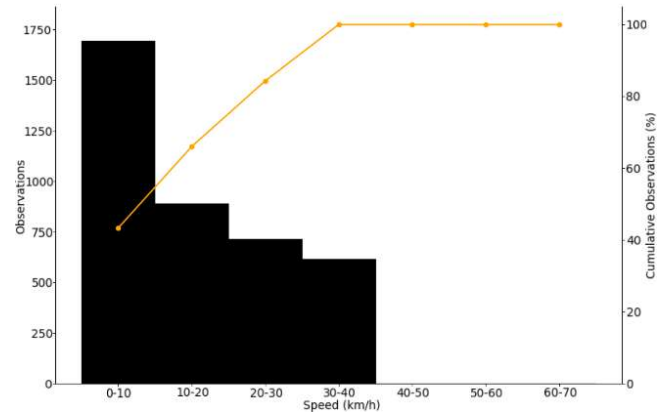


**Figure 18 PM Peak Vehicle Speed Distribution**

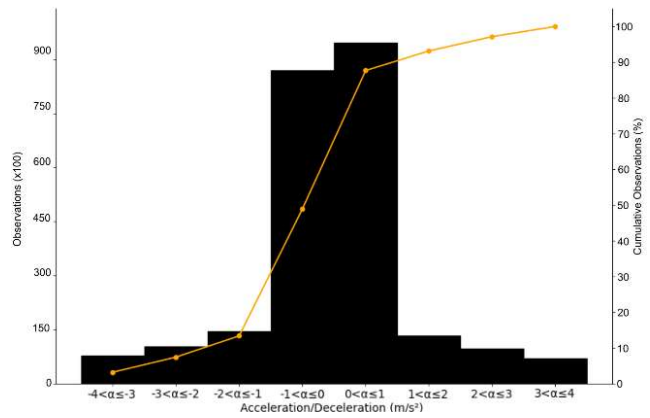


**Figure 19 AM Peak Acceleration/Deceleration Distribution**

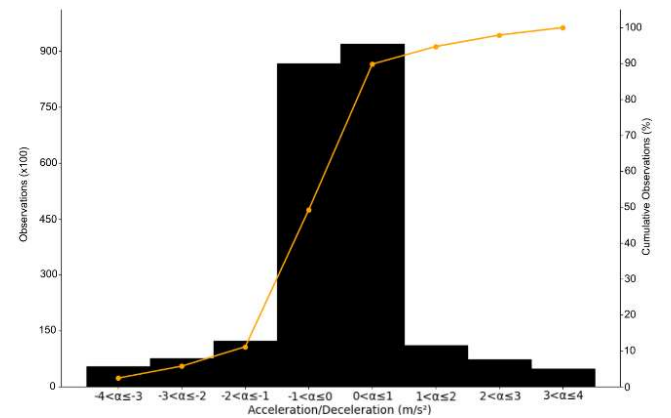


**Figure 20 PM Peak Acceleration/Deceleration Distribution**

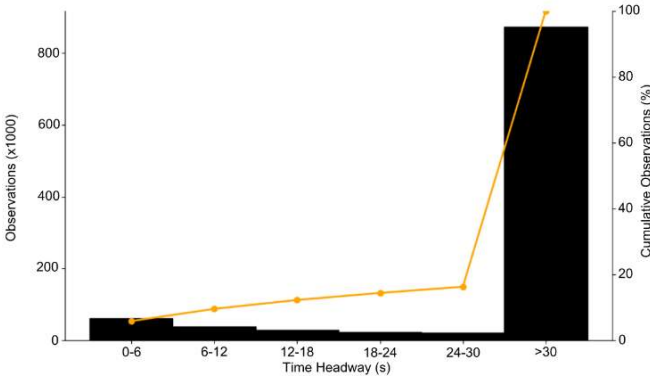


**Figure 21 AM Peak Time Headway Distribution**

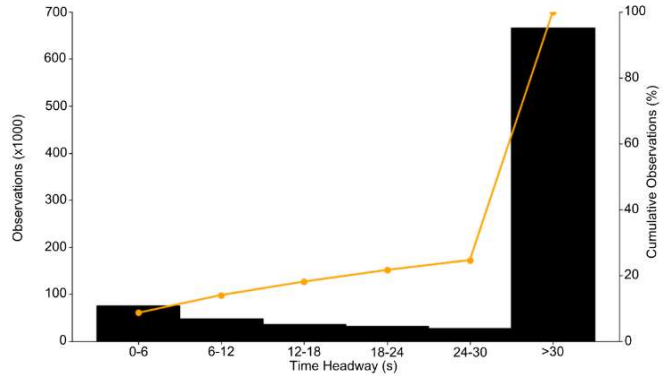


**Figure 22 PM Peak Time Headway Distribution**

## 4.2 Performance Model Calibration

Performance model calibration is essential to ensure that the simulation accurately reflects real-world driving behaviors and system performance characteristics.

#### 4.2.1 Driving Behavior

Driving behavior is a critical component in traffic simulation, influencing vehicle interactions and overall traffic flow.

##### 4.2.1.1 Car-Following

The car-following model is a key aspect of traffic dynamics, illustrating the mechanisms by which vehicles modulate speeds and sustain safe following distances across varying traffic conditions. In VISSIM, car-following is modelled using the Wiedemann 74, which is more suitable for urbanized corridors.

###### 4.2.1.1.1 Average Standstill Distance

The standstill distance, the gap maintained by vehicles when stationary, is a fundamental parameter in car-following models. LiDAR sensors were used along the study area to measure the vehicular movement and average standstill distance during AM and PM peak hours and produce distribution figures for standstill distances capturing data from 260 vehicles in the AM peak and 362 vehicles in the PM peak. To enhance the accuracy of the dataset, outlier standstill distances produced from the LiDAR system that are less than 0 metres or greater than 10 metres were omitted. The averaged standstill distances were found to be 1.6 metres for the AM peak and 1.9 metres for the PM peak, as shown in Figure 23 and Figure 24, which are lower than the default value of 2 metres. The slight increase in average standstill distance during the afternoon peak indicates a greater spacing between stationary vehicles. This highlighted standstill distance correlates with a higher safety distance reduction factor – covered in Section 4.2.1.2.3 – suggesting reduced pressure to reach destination on time compared to the morning peak hour, when punctuality for work is prioritized.

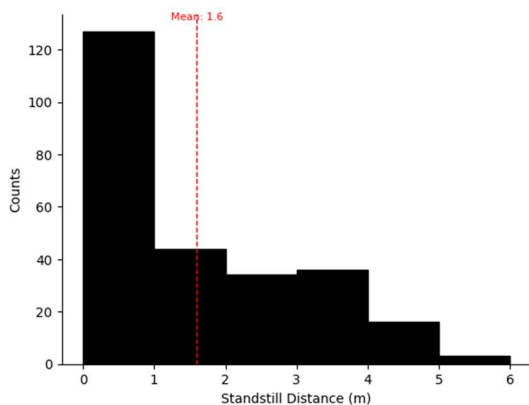


**Figure 23 AM Peak Standstill Distance Distribution**

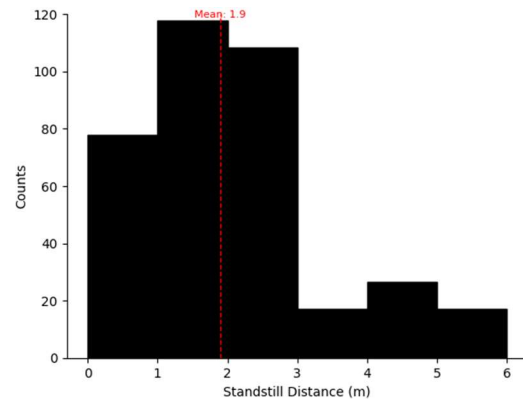


**Figure 24 PM Peak Standstill Distance Distribution**

###### 4.2.1.1.2 Additive and Multiplicative Parts of Desired Safety Distance

The desired safety distance, which consists of additive and multiplicative parts, ensures that vehicles maintain a safe following distance. The default values are 2 metres for the additive part and 3 metres for the multiplicative part of the safety distance. To

accurately reflect the driving behaviour between each pair of leading-following vehicles, a linear programming model as detailed in Section 3.2.1.1.3 is used to optimize the parts values through a minimization process. For both the AM and PM peak hours, the additive part of the desired safety distance has resulted to be 1 meter for both the AM and PM peak hours. Meanwhile, the multiplicative part varies between the two peak hours, with a value of 1.7 for the AM peak and 1.3 for the PM peak. The variation in the multiplicative part suggests that the afternoon peak hour experiences a higher traffic saturation flow compared to the morning peak hour.

#### 4.2.1.2 Lane Changing

Lane-changing behavior impacts traffic flow efficiency and safety, particularly in urban environments with frequent interactions.

##### 4.2.1.2.1 Maximum and Accepted Deceleration

Deceleration rates during lane-changing maneuvers are crucial for modeling vehicle dynamics and ensuring realistic simulation outcomes. The default value for the accepted deceleration is  $-1 \text{ m/s}^2$  for both leading and following vehicles, while the maximum deceleration is  $-4 \text{ m/s}^2$  for the leading vehicle and  $-3 \text{ m/s}^2$  for the following vehicle. The deceleration rate distribution figures for lane-changing and following vehicles were created based on the cleaned trajectory data. The 85<sup>th</sup> percentile for maximum deceleration and the 15<sup>th</sup> percentile for accepted deceleration were calculated, with maximum deceleration rates of  $-3.54 \text{ m/s}^2$  and  $-3.51 \text{ m/s}^2$  for the AM and PM peak hours respectively for lane-changing vehicles as shown in Figure 25 and Figure 27, and  $-3.52 \text{ m/s}^2$  and  $-3.18 \text{ m/s}^2$  for following vehicles as shown in Figure 26 and Figure 28. Accepted deceleration rates were  $-0.93 \text{ m/s}^2$  and  $-0.92 \text{ m/s}^2$  for lane-changing vehicles, and  $-0.92 \text{ m/s}^2$  and  $-0.89 \text{ m/s}^2$  for following vehicles.

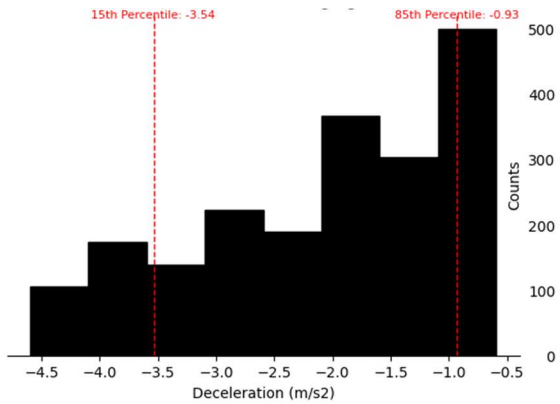


**Figure 25 AM Peak Lane Changing Deceleration Distribution**

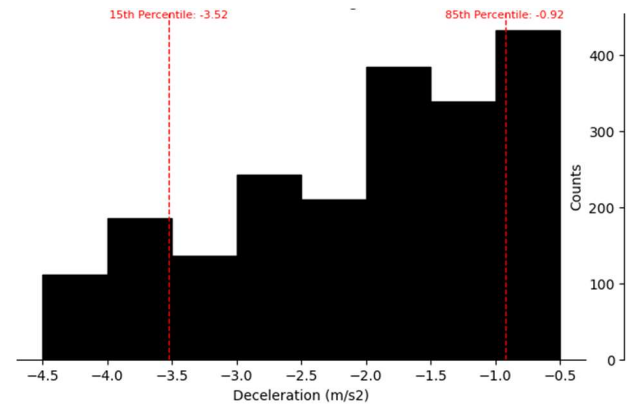


**Figure 26 AM Peak Following Deceleration Distribution**

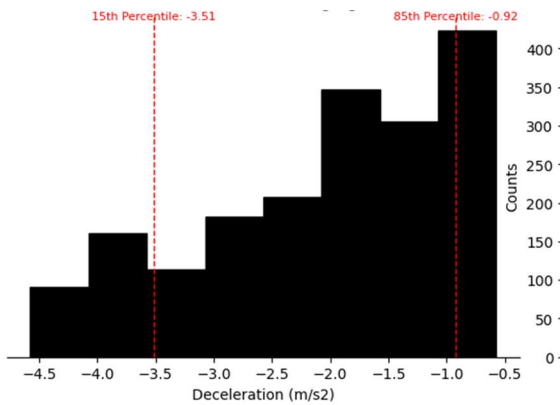


**Figure 27 PM Peak Lane Changing Deceleration Distribution**

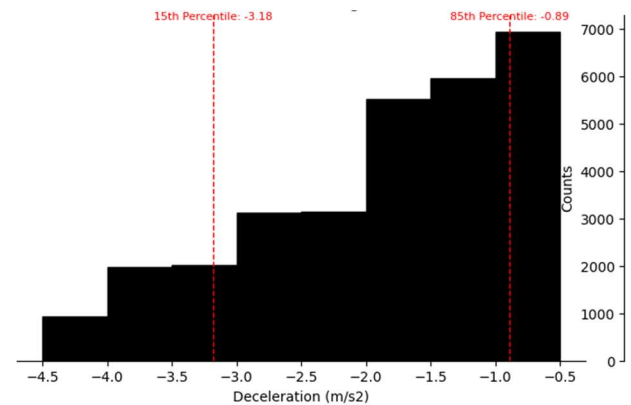


**Figure 28 PM Peak Following Deceleration Distribution**

Table 10 summarizes the resulted deceleration of lane changing and following vehicles at each peak hour.

**Table 10 Maximum and Accepted Deceleration of Lane Changing Model**

|                                                | Lane Changing Vehicle |       | Following Vehicle |       |
|------------------------------------------------|-----------------------|-------|-------------------|-------|
|                                                | AM                    | PM    | AM                | PM    |
| <b>Maximum Deceleration (m/s<sup>2</sup>)</b>  | -3.54                 | -3.51 | -3.52             | -3.18 |
| <b>Accepted Deceleration (m/s<sup>2</sup>)</b> | -0.93                 | -0.92 | -0.92             | -0.89 |

#### 4.2.1.2.2 Minimum Distance Headway

The minimum distance headway, defined as the closest permissible distance between two vehicles, is a key parameter in safety analysis. In VISSIM, the default value is 0.5 metres. This thesis calculates the distance headway using the Euclidean distance between vehicles, derived from latitude and longitude coordinates converted to metres while accounting for Earth's curvature. The resulting headway distribution ranged from 0 to 10 metres, with the lowest and most frequent bin falling between 0.5 to 0.75 metres. The

averaged minimum headways were 0.63 metres for the AM peak and 0.61 metres for the PM peak, as illustrated in Figure 29 and Figure 30.

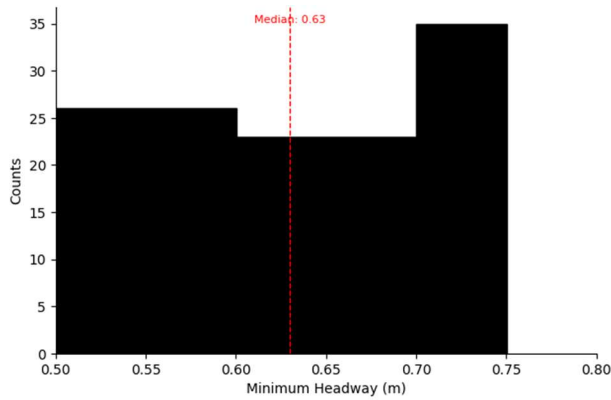


**Figure 29 AM Peak Minimum Distance Headway Distribution**

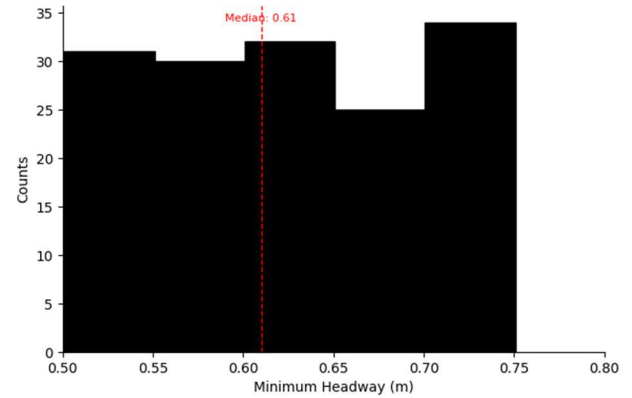


**Figure 30 PM Peak Minimum Distance Headway Distribution**

#### 4.2.1.2.3 Safety Distance Reduction Factor

The safety distance reduction factor adjusts the desired safety distance during lane changes to account for varying traffic conditions. The default value in VISSIM for the parameter is 0.6. To determine the appropriate safety distance reduction factor for each peak hour in this thesis, the ratio of the minimum and maximum headway between each pair of the lane changing and following vehicles was calculated. Headways exceeding 10 metres were omitted from the analysis to improve accuracy. The results indicated a safety distance reduction factor of 0.92 for AM peak hour and 0.93 for PM peak hour, as shown in Figure 31 and Figure 32.

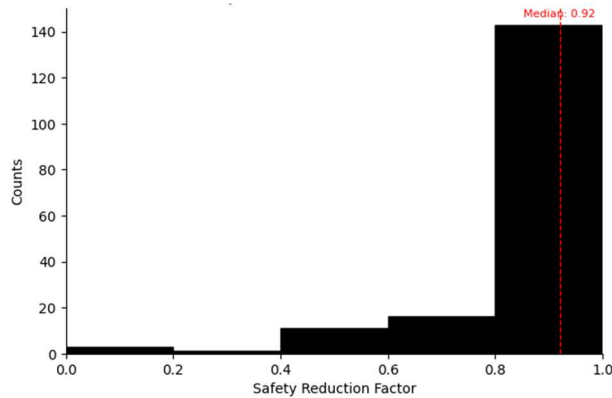


**Figure 31 AM Peak Safety Reduction Factor Distribution**

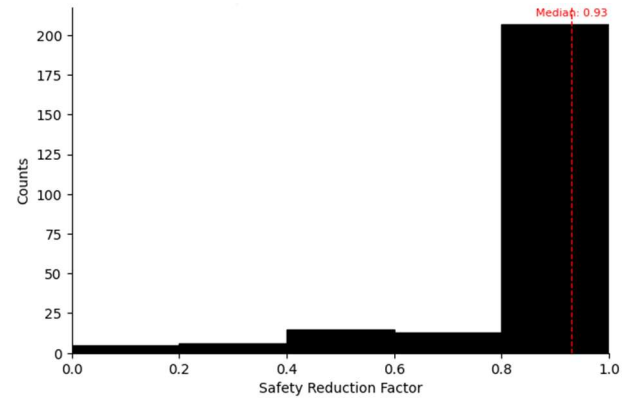


**Figure 32 PM Peak Safety Reduction Factor Distribution**

#### 4.2.2 Vehicle Performance

Vehicle performance parameters, such as speed and acceleration, are critical for accurate traffic simulation and model calibration.

##### 4.2.2.1 Desired Speed

The concept of desired speed is defined as the speed at which drivers typically prefer to travel under normal, non-congested traffic conditions. In VISSIM, for a speed limit of 50 km/h, the desired speed distribution ranges from 48 km/h to 58 km/h.

To establish the desired speed distributions, measures were taken to eliminate outliers and ensure that the data represents typical free-flow conditions. A minimum time headway of over 6 seconds is used to measure a free-flow state, while a minimum speed threshold is applied for the peak hours to exclude extremely slow-moving vehicles. To determine the average speeds, data within the 25<sup>th</sup> to 75<sup>th</sup> quantile range are considered. This approach offers an advantage of focusing on the interquartile range while also providing insight into medium to long-term speed trends with effectively mitigating rare cases. The findings from this approach indicate minimum speed thresholds for outlier removal of 35 km/h for the AM peak hour and 25 km/h for the PM peak hour.

The study utilized LiDAR sensor data to simulate and analyze various minimum desired speeds for the St-Maurice corridor. Each minimum desired speed is evaluated through 7 simulation runs representing a week. The simulated average vehicle speeds and travel times are compared with observed values using MANE to assess the error degree. Figure 33 to Figure 36 are generated to illustrate the minimum desired speed for both AM and PM peak hours, as well as for the northbound and southbound directions along the corridor. – along St-Maurice corridor. The analysis identified minimum desired speed thresholds of 35 km/h and 39 km/h for AM peak northbound and southbound movements, and 25 km/h and 30 km/h for PM peak northbound and southbound movements. These thresholds resulted in the lowest MANE, hence, providing necessary alignment between the simulated and observed travelling behavior.

Although the minimum desired speed extracted from LiDAR sensor data provides low MANE, it is essential to investigate differences when compared to LiDAR Insight Platform. Notably the Insight Platform lacks the capability to filter free-flow state, yet the average minimum speed for the corridor is calculated by averaging the minimum speeds recorded across the six intersections. The Insight Platform results in minimum speed of 34 km/h for northbound movement, with the lowest value of 29 km/h observed at Francois Xavier intersection; and 31 km/h for the southbound movement, with the lowest value of 27 km/h at each of Paul and Laviolette intersections. As the discrepancy is considered minor, the LiDAR sensor data can be confirmed as valid and suitable to use for minimum desired speed along the corridor.

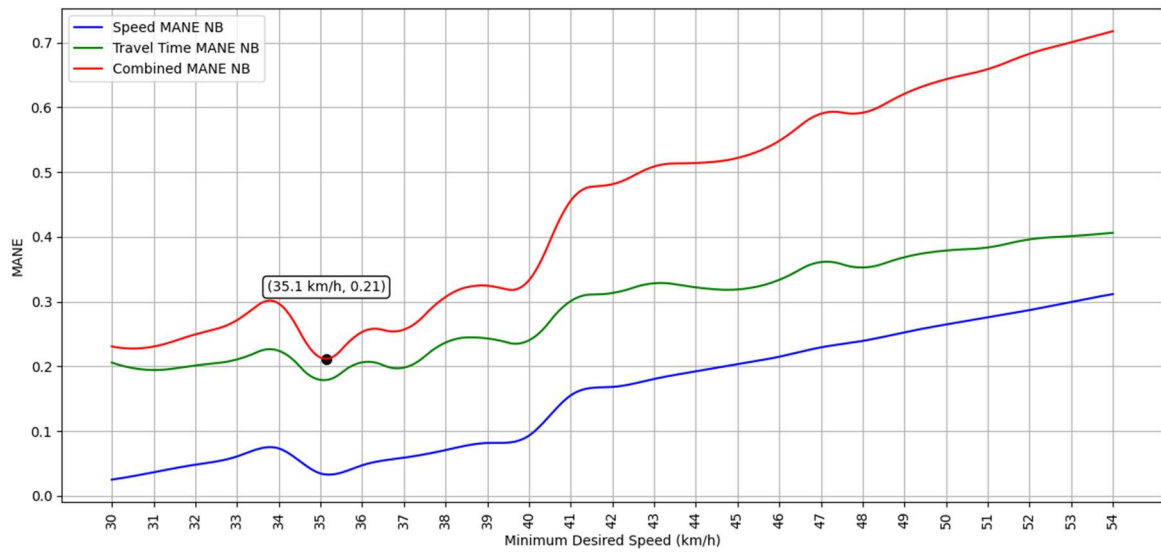


**Figure 33 Northbound AM Peak MANE Comparison for Different Speed Thresholds**

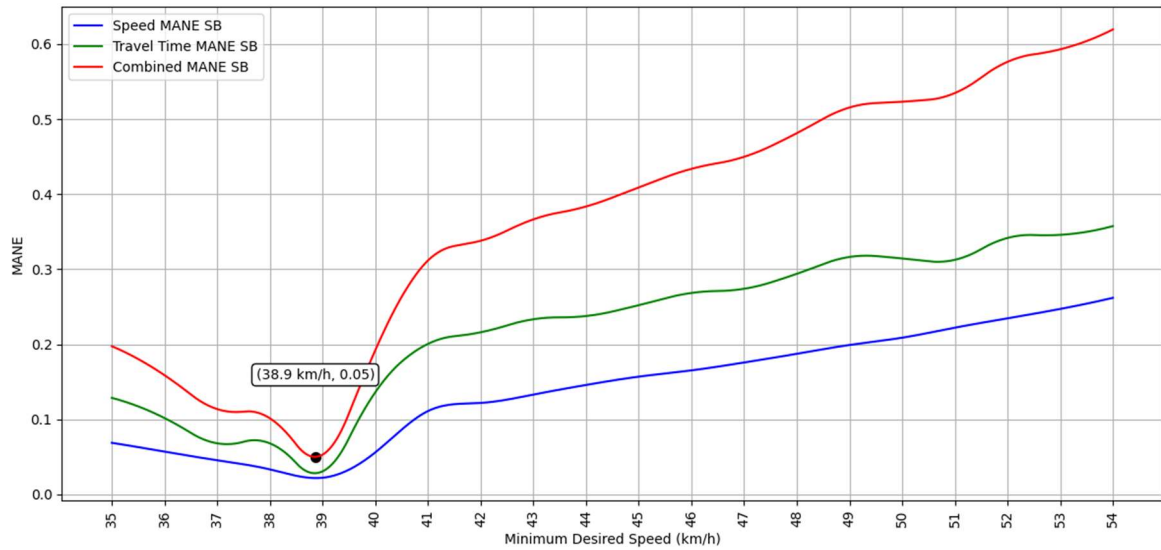


**Figure 34 Southbound AM Peak MANE Comparison for Different Speed Thresholds**

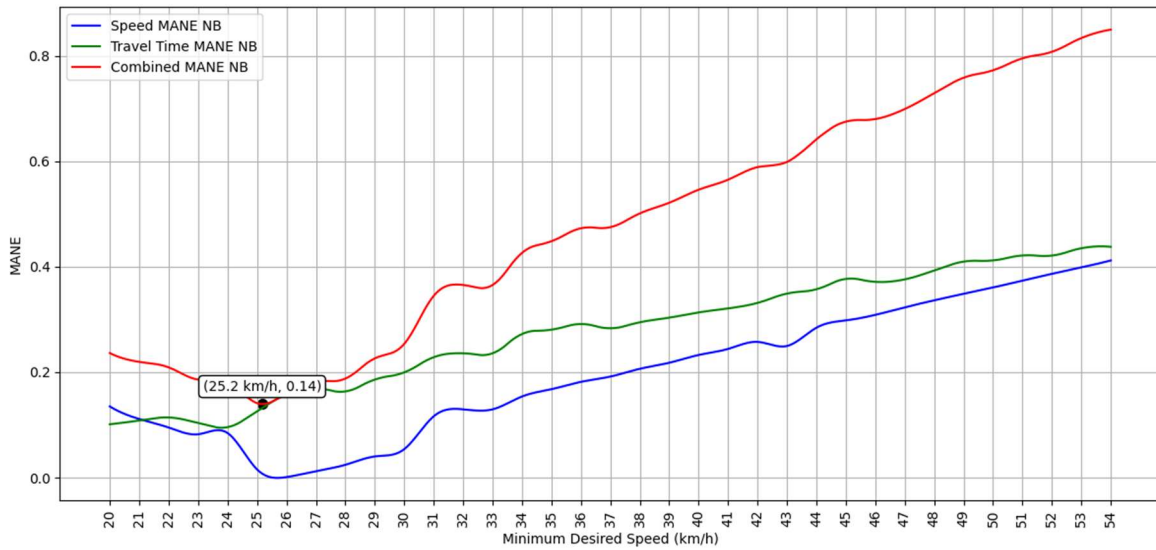


**Figure 35 Northbound PM Peak MANE Comparison for Different Speed Thresholds**

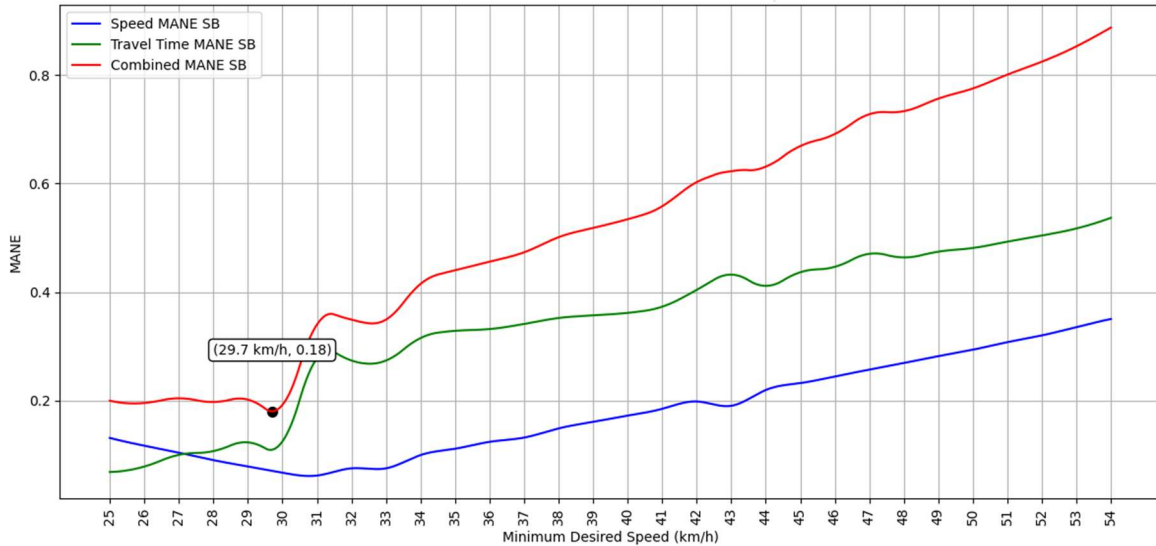


**Figure 36 Southbound PM Peak MANE Comparison for Different Speed Thresholds**

A probability plot is subsequently generated to illustrate the cumulative distribution of desired speeds by plotting the maximum speed of each recorded vehicle traveling in a free-flow state. The maximum desired speeds are determined through a regression model that best fits the observed speed data collected from LiDAR sensors. For the AM peak, the maximum desired speed is found to be 41 km/h for northbound movements and 50 km/h for southbound movements. For the PM peak, the maximum desired speed 42 km/h for both northbound and southbound movements.

It is worth noting that the observed maximum desired speeds, particularly those below the posted speed limit of 50 km/h, may be influenced by congestion in the area. Under such conditions, vehicles are constrained by traffic flow, preventing them from

reaching their full desired speeds. Additionally, some recorded speeds may reflect vehicles accelerating from lower speeds rather than driving at their actual maximum desired speeds. The cumulative probability distributions of desired speed are as shown in Figure 37 to Figure 40.

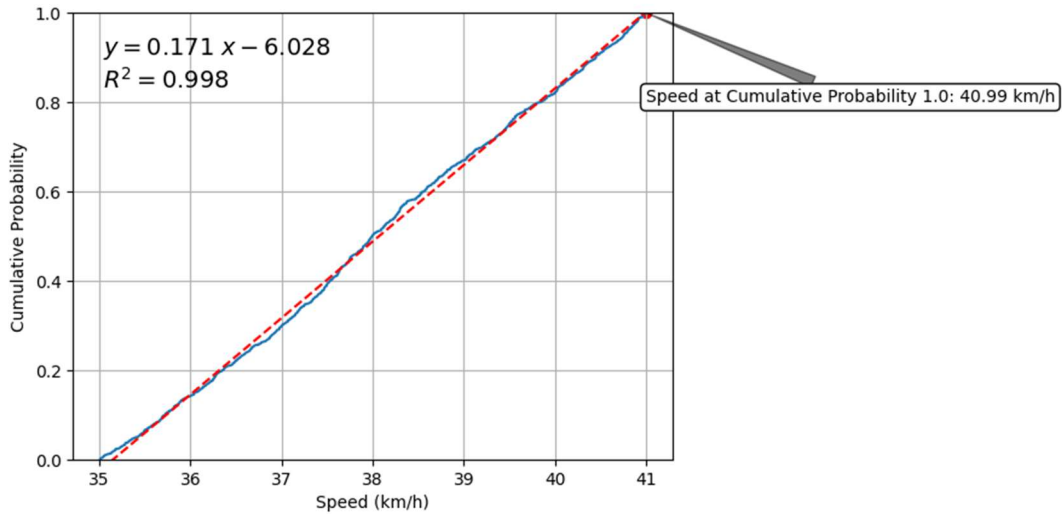


**Figure 37 Northbound AM Peak Desired Speed Cumulative Probability**

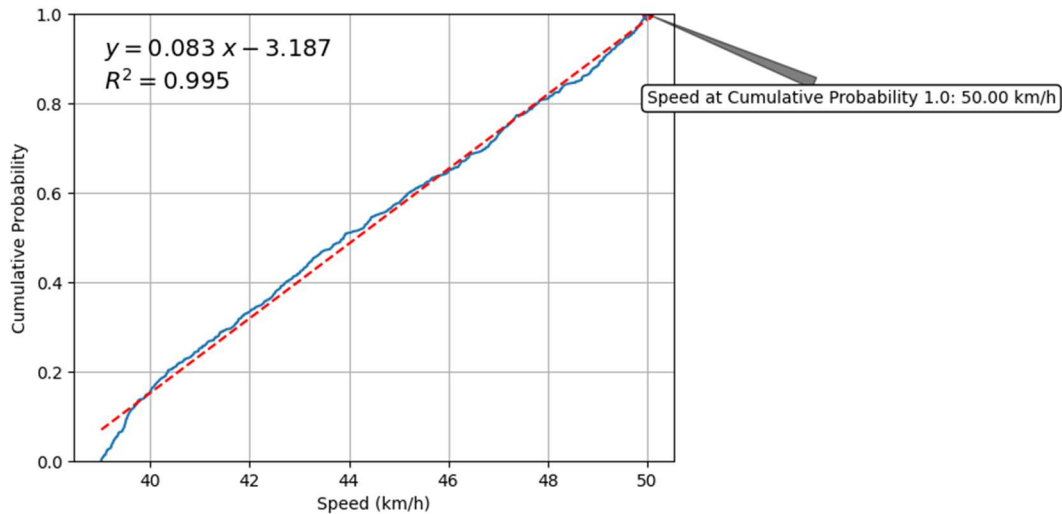


**Figure 38 Southbound AM Peak Desired Speed Cumulative Probability**

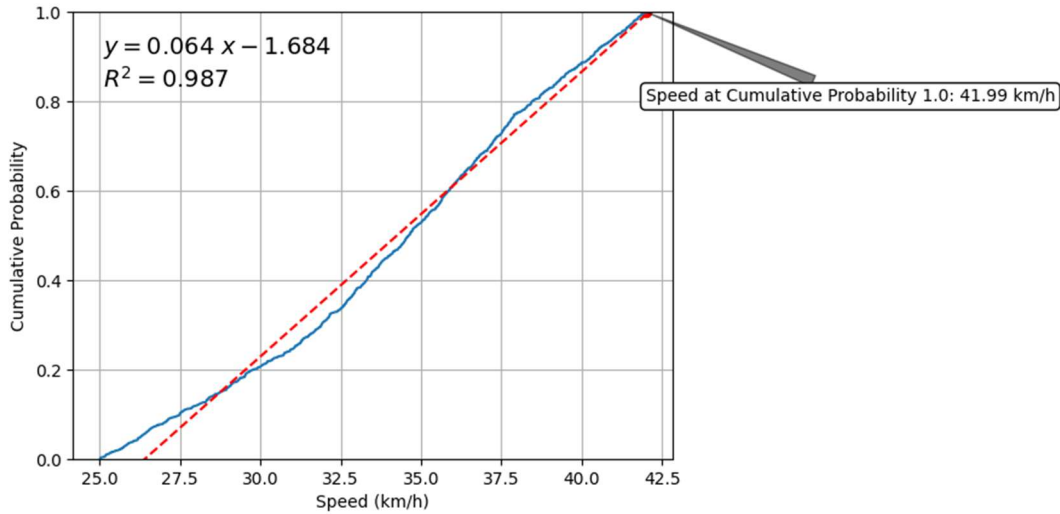


**Figure 39 Northbound PM Peak Desired Speed Cumulative Probability**

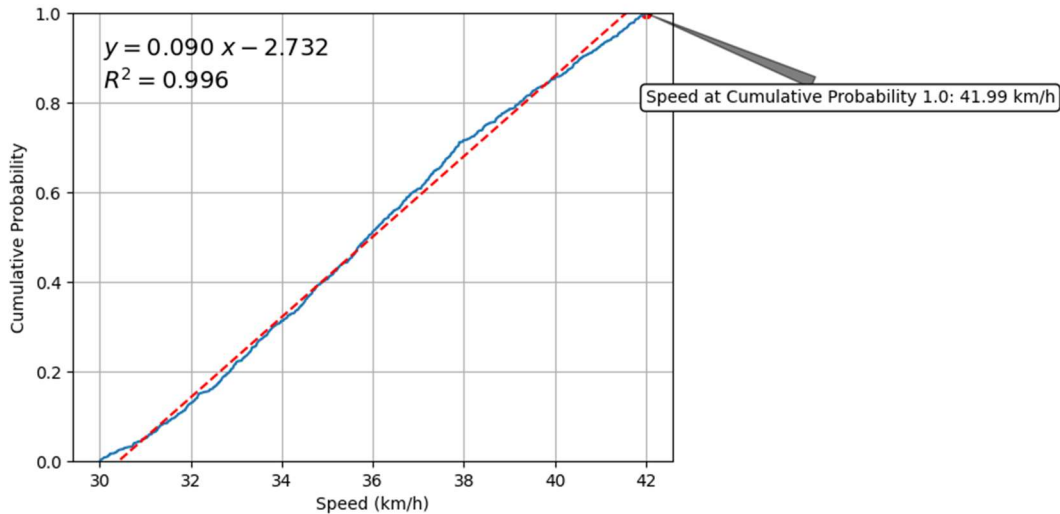


**Figure 40 Southbound PM Peak Desired Speed Cumulative Probability**

In comparison, the LiDAR Insight Platform as shown in Figure 41, indicates different maximum speed for the corridor than the raw LiDAR data. However, the Insight Platform does not provide peak hour specific maximum speeds; rather, it offers daily speed metrics. To address this limitation, the desired speed for each direction is determined by averaging the daily maximum speeds across the six intersections over the entire month of July. The averaged maximum speed is calculated to be 54 km/h for the northbound movement and 51 km/h for the southbound movement. Given the notable discrepancy between the LiDAR sensor data and the Insight Platform, both data sources are incorporated into microsimulation model to compare traffic operations with observed condition. The Insight Platform concludes the closest alignment with the observed speed and travel time; thus, the thesis adopts maximum desired speed of 54 km/h for northbound

and 55 km/h for southbound movement. Moreover, adopting the extracted values from the Insight Platform as the maximum desired speed is justifiable, as the maximum desired speed should be typically higher than the posted speed limits.

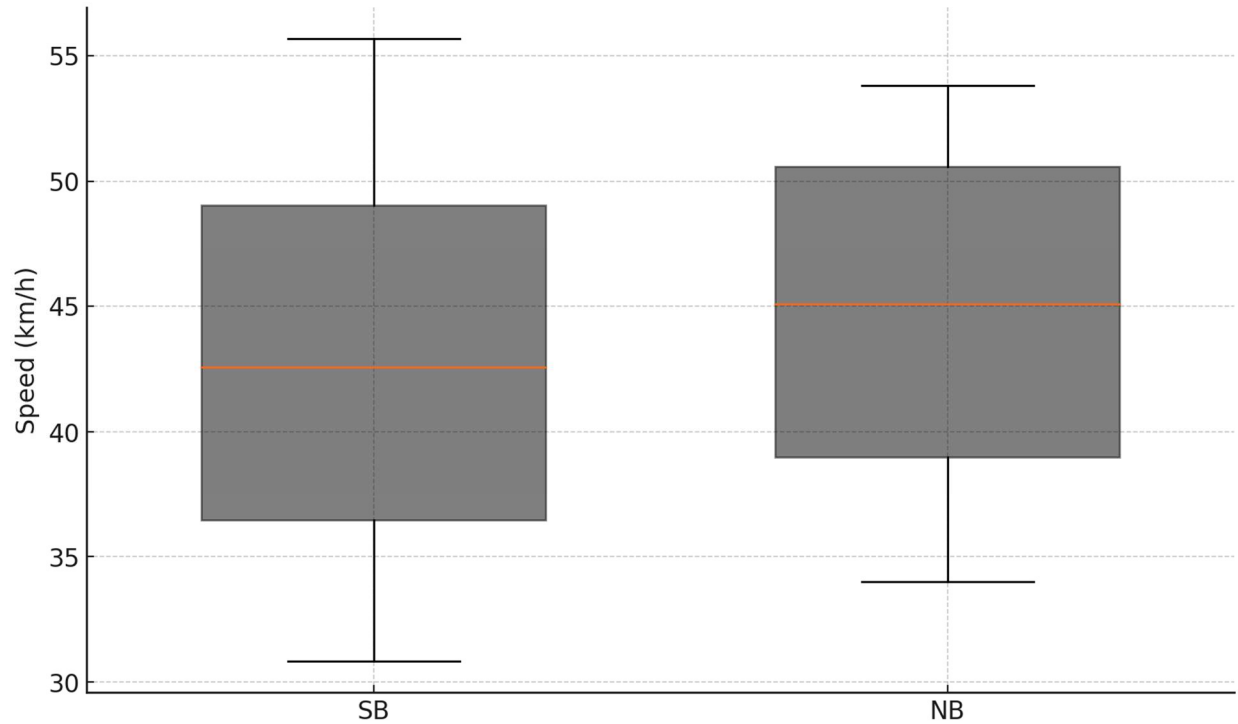


**Figure 41 LiDAR Insight Platform Speed Candlestick Chart**

#### 4.2.2.2 Desired Acceleration/Deceleration

Desired acceleration and deceleration rates reflect driver preferences and vehicle capabilities during speed changes. In VISSIM, the desired acceleration and deceleration are modeled as functions of a vehicle speed, with default values set for minimum, median, and maximum thresholds to account for various driving behaviors and vehicle characteristics. To determine the optimal input values, the thesis conducted multiple simulations testing different percentiles applied to collected data to identify those thresholds minimizing the MANE. The simulation runs are conducted, exploring combinations of percentiles for desired acceleration and deceleration. The percentile combination of the 5<sup>th</sup> for minimum, 50<sup>th</sup> for median, and 95<sup>th</sup> for maximum result in producing the lowest MANE for both AM and PM peak periods, as shown in Table 11. These percentiles are applied to the acceleration and deceleration data derived from LiDAR sensor observations to generate the final desired acceleration and deceleration distributions. Figure 42 to Figure 45 illustrate the results for the AM and PM peak hours, with Figure 42 and Figure 43 represent the desired acceleration and deceleration for the AM peak hour, while Figure 44 and Figure 45 for the PM peak. The undertaken approach ensures the selected values for the desired acceleration and deceleration reflect observed driving behaviours and vehicle dynamics along the corridor as closely as possible.

**Table 11 Desired Acceleration/Deceleration MANE Results at Different Percentiles**

| Simulation<br>Test # | Acceleration     |                  |                  | Deceleration     |                  |                  | MANE       |            |
|----------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------|------------|
|                      | Minimum          | Median           | Maximum          | Minimum          | Median           | Maximum          | AM<br>Peak | PM<br>Peak |
| 1                    | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> | 0.256      | 0.250      |
| 2                    | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> | 0.365      | 0.263      |
| 3                    | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> | 0.362      | 0.263      |
| 4                    | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> | 0.381      | 0.380      |
| 5                    | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> | 0.346      | 0.316      |
| 6                    | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> | 0.386      | 0.375      |
| 7                    | 5 <sup>th</sup>  | 50 <sup>th</sup> | 95 <sup>th</sup> | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> | 0.381      | 0.262      |
| 8                    | 10 <sup>th</sup> | 50 <sup>th</sup> | 90 <sup>th</sup> | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> | 0.347      | 0.313      |
| 9                    | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> | 15 <sup>th</sup> | 50 <sup>th</sup> | 85 <sup>th</sup> | 0.376      | 0.374      |

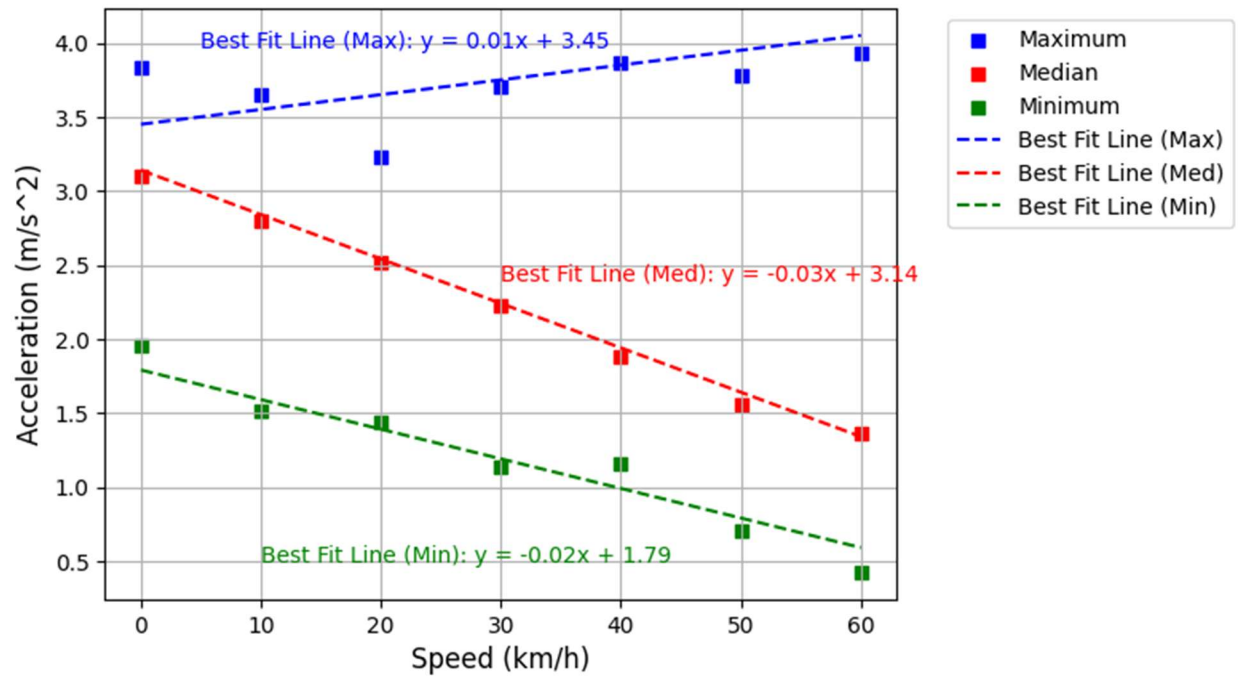


**Figure 42 Calibrated Desired Acceleration from LiDAR for AM Peak**

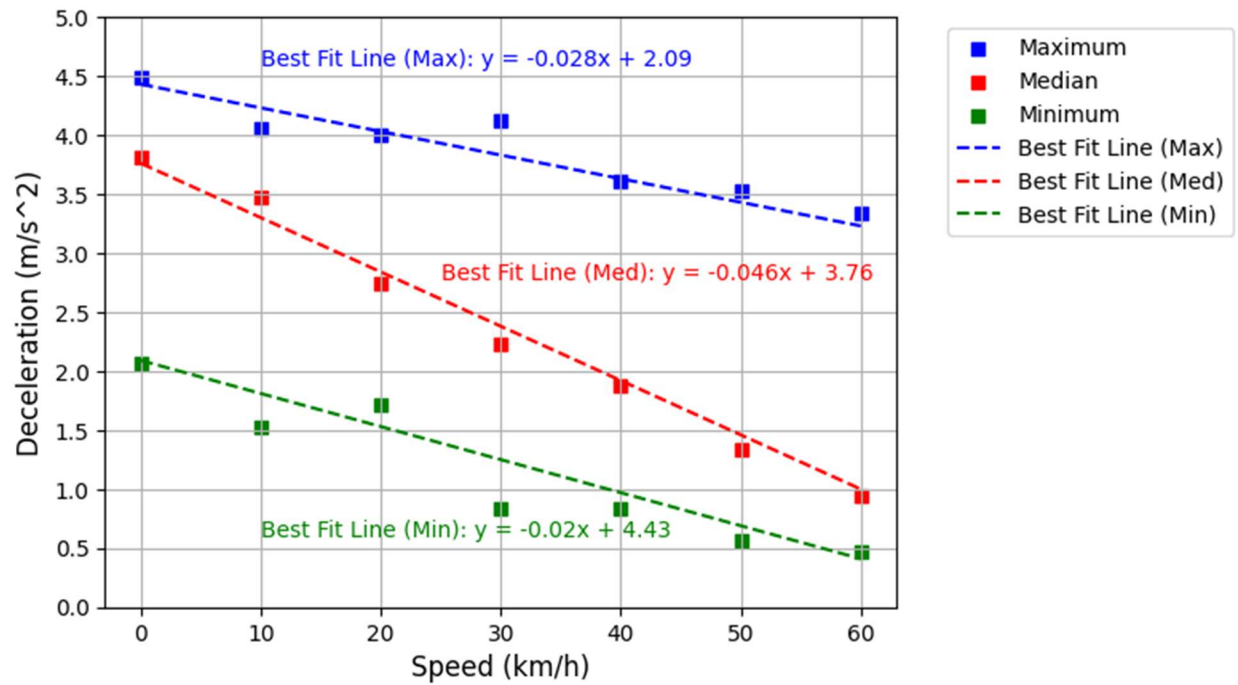


Figure 43 Calibrated Desired Deceleration from LiDAR for AM Peak

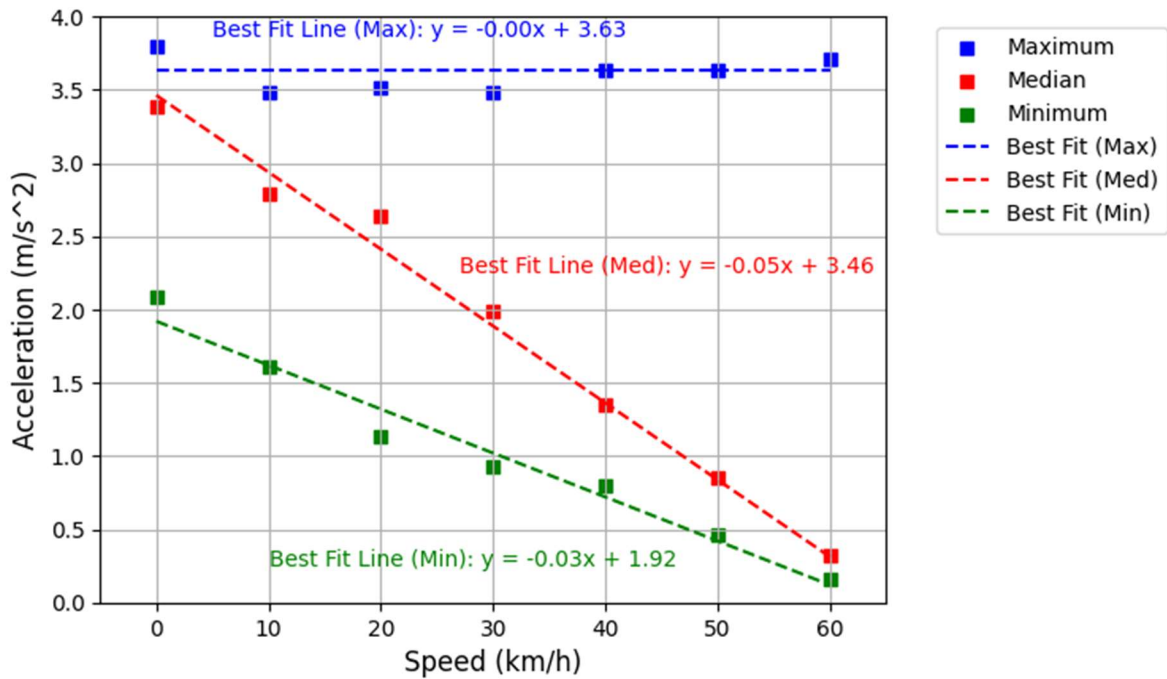


Figure 44 Calibrated Desired Acceleration from LiDAR for PM Peak

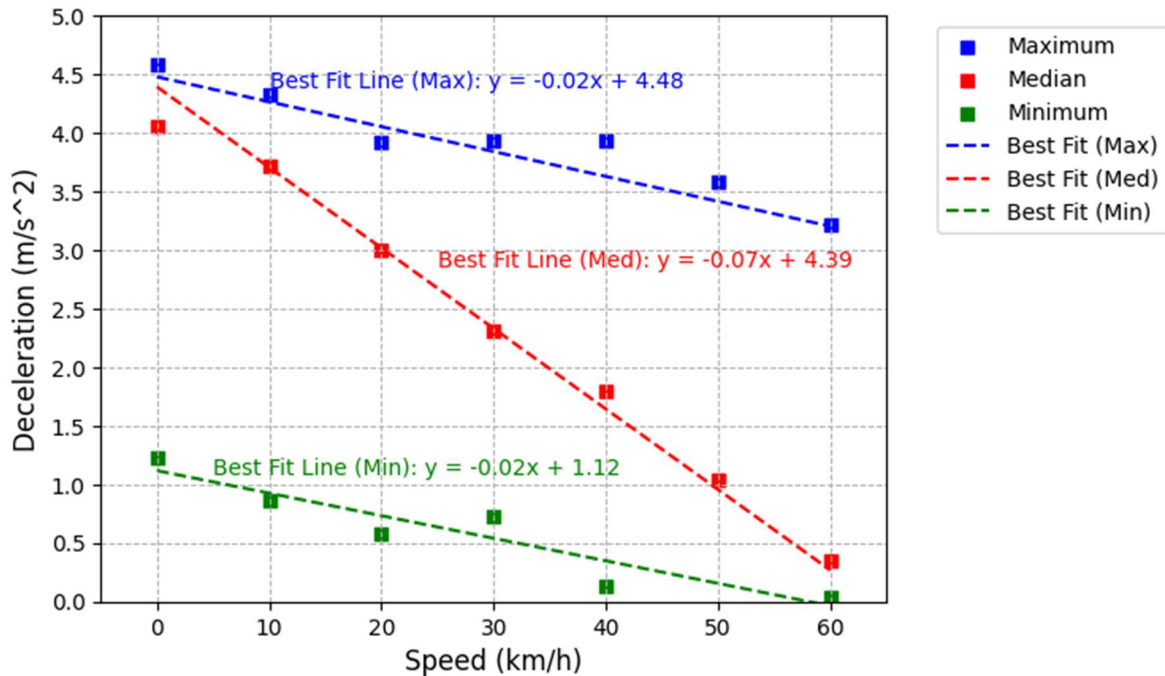


Figure 45 Calibrated Desired Deceleration from LiDAR for PM Peak

#### 4.2.2.3 Maximum Acceleration/Deceleration

Maximum acceleration and deceleration rates define the limits of vehicle performance and influence traffic dynamics. The calibrated values for maximum acceleration and deceleration function are based on mirroring the desired minimum and maximum values in Section 4.2.2.2 to represent the vehicle capabilities. Subsequently, a relationship was generated for the default median values of maximum acceleration and deceleration against the default desired median values, to be applied as the calibrated median for maximum acceleration and deceleration function. This method was applied for each of morning and afternoon peak hours.

### 4.3 Performance Model Validation

The validation of the performance model validation is critical for ensuring that the calibrated model accurately reflects real-world traffic dynamics. In this context, travel time and speed distribution figures are generated for the corridor, consisting of both northbound and southbound movements. The analysis focuses on a defined route extending from 28 metres south of the centerline intersection at Rue Laviolette to 30 metres north of the centerline intersection at Boulevard des Chenaux. It is worth noting that the speed distribution from the LiDAR API reflects compiled speeds from individual intersections along the corridor, as the sensor extracts data from one intersection at a time, as previously mentioned in Section 4.1. In contrast, the Google Distance Matrix API captures speed distributions that reflect dynamic variations across the entire corridor, accounting for the impacts of traffic signals, as the Google Matrix API uses the starting and ending points of

corridor. These locates from Google Matrix API, were also used for microsimulation data collection points to ensure a fair comparison between observed and simulated datasets.

Figure 46 to Figure 49 show the compared travel time for AM peak hour over one-week period between: LiDAR API, Google Matrix API, performance calibrated model, and default model.

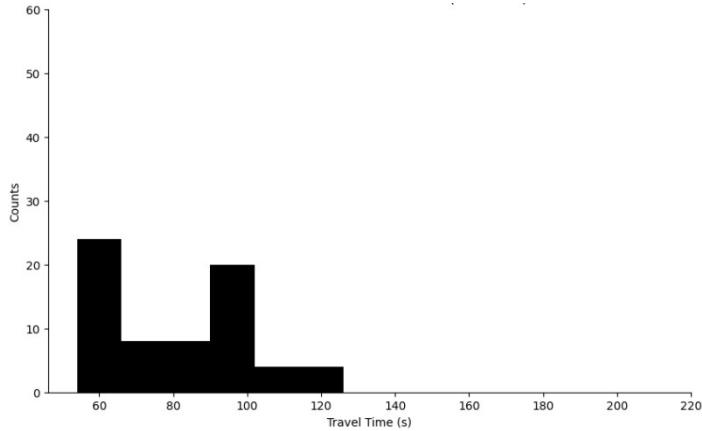


**Figure 46 AM Peak LiDAR API Observed Travel Time Distribution**

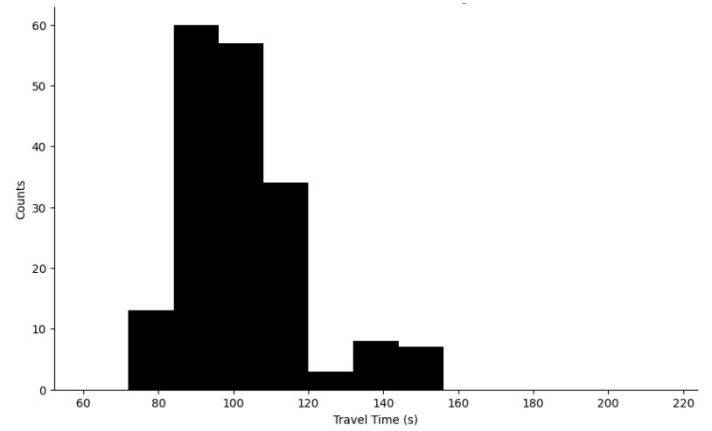


**Figure 47 AM Peak Google Matrix API Observed Travel Time Distribution**

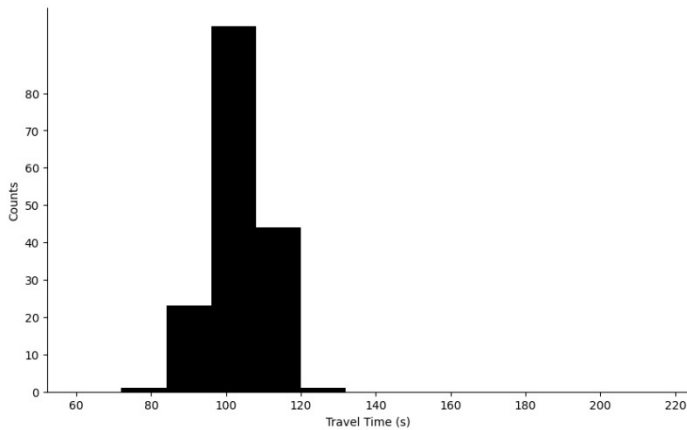


**Figure 48 AM Peak Simulated Travel Time Distribution for Performance Calibration**

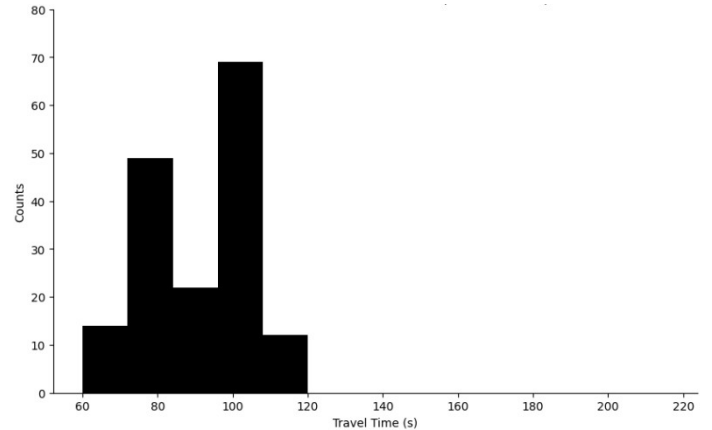


**Figure 49 AM Peak Simulated Travel Time Distribution for Default Model**

Figure 50 to Figure 53 show the compared travel time for PM peak hour over one-week period between: LiDAR API, Google Matrix API, performance calibrated model, and default model.

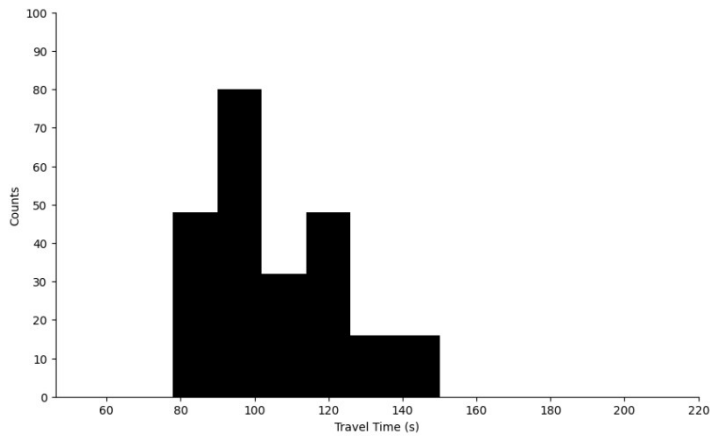


**Figure 50 PM Peak LiDAR API Observed Travel Time Distribution**

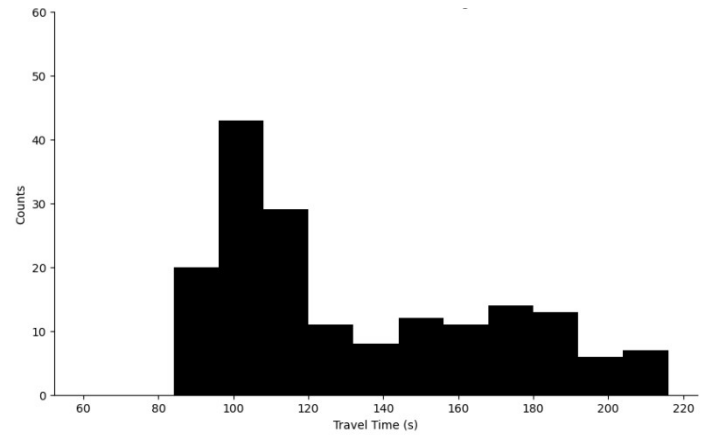


**Figure 51 PM Peak Google Matrix API Observed Travel Time Distribution**

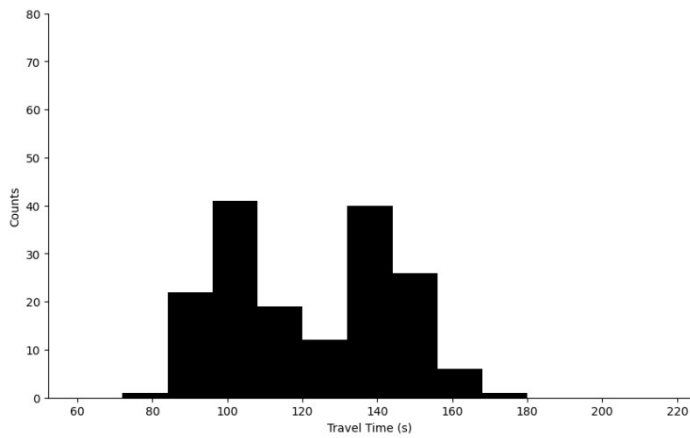


**Figure 52 PM Peak Simulated Travel Time Distribution for Performance Calibration**

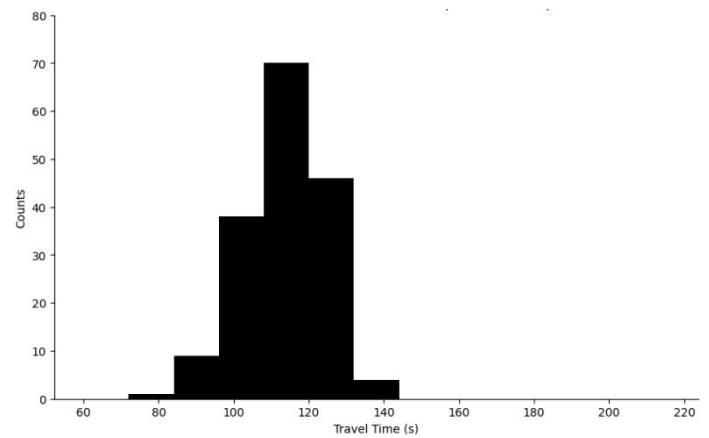


**Figure 53 PM Peak Simulated Travel Time Distribution for Default Model**

Figure 54 to Figure 57 show the compared speed for AM peak hour over one-week period between: LiDAR API, Google Matrix API, performance calibrated model, and default model.

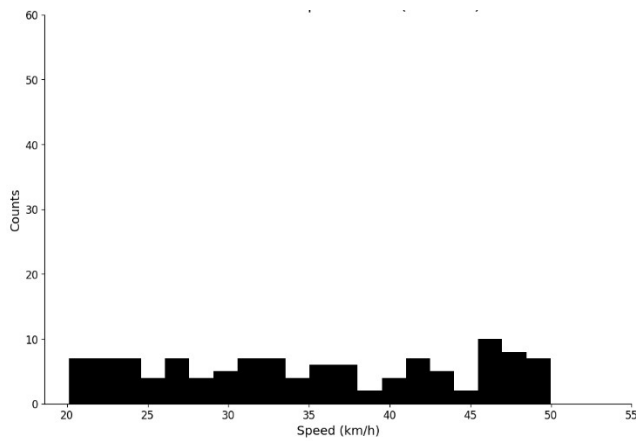


**Figure 54 AM Peak LiDAR API Observed Speed Distribution**

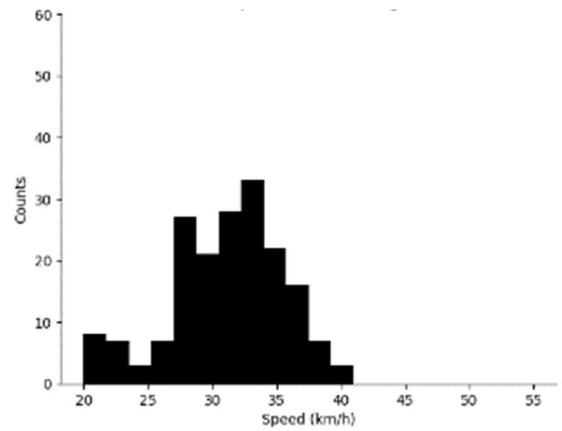


**Figure 55 AM Peak Google Matrix API Observed Speed Distribution**

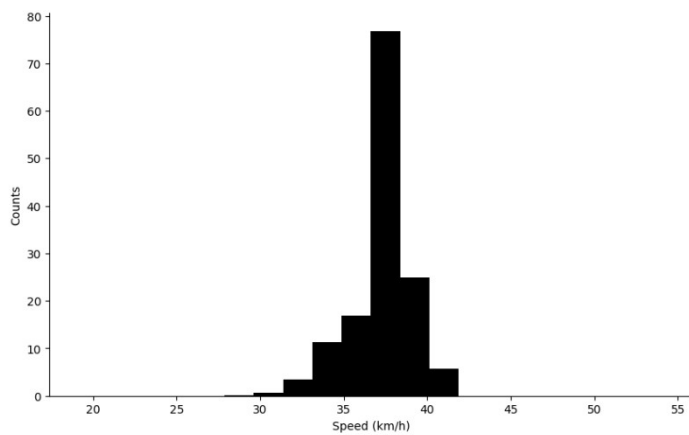


**Figure 56 AM Peak Simulated Speed Distribution for Performance Calibration**

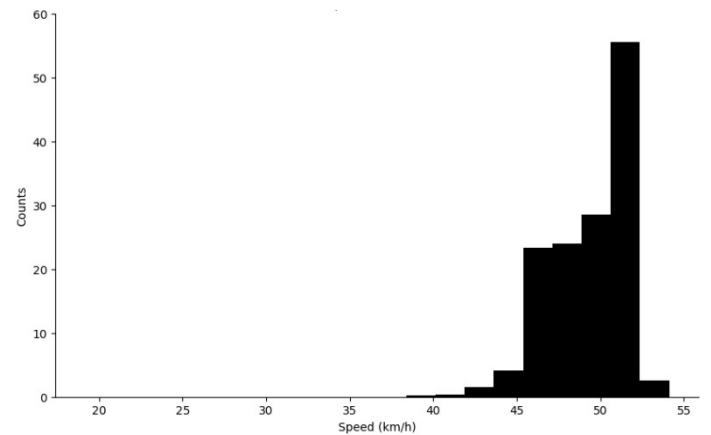


**Figure 57 AM Peak Simulated Speed Distribution for Default Model**

Figure 58 to Figure 61 show the compared speed for PM peak hour over one-week period between: LiDAR API, Google Matrix API, performance calibrated model, and default model.

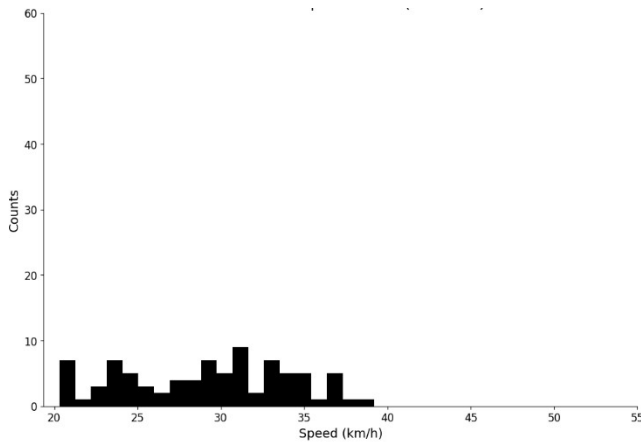


**Figure 58 PM Peak LiDAR API Observed Speed Distribution**

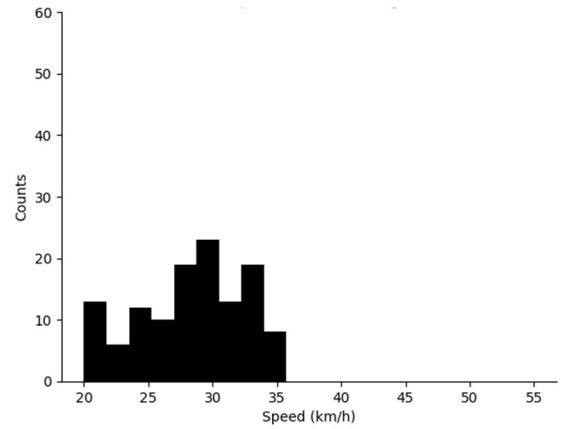


**Figure 59 PM Peak Google Matrix API Observed Speed Distribution**

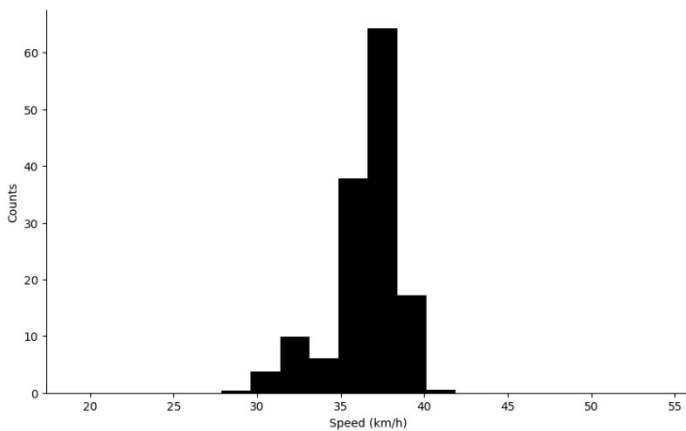


**Figure 60 PM Peak Simulated Speed Distribution for Performance Calibration**

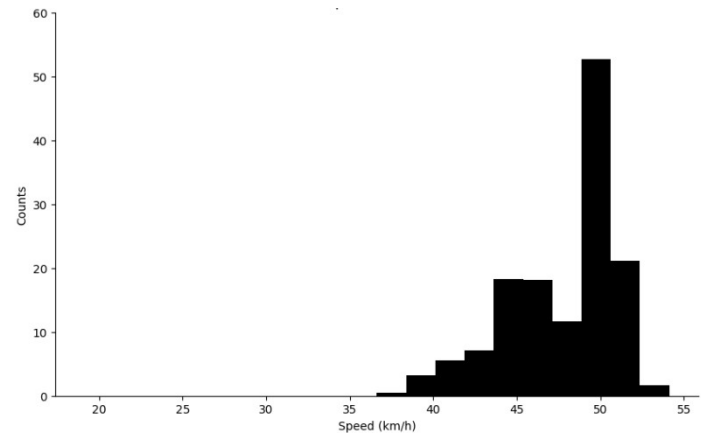


**Figure 61 PM Peak Simulated Speed Distribution for Default Model**

#### 4.4 Parameter Selection

In the parameter selection process, sensitivity analysis is performed for both the AM and PM peak hours using one-way ANOVA test to identify the influential variables affecting model performance in terms of conflicts. Each pair of VISSIM parameters is tested using their minimum and maximum values derived from the literature, with a median value incorporated when necessary. All other parameters are held constant at the default settings to isolate the effect of the paired parameters being tested. Each combination of parameter values undergo five simulation runs. Additionally, the calibrated parameters from the performance model are integrated throughout the sensitivity analysis process, apart from parameters requiring recalibration including standstill distance and minimum distance headway.

Conflicts examined during the sensitivity analysis through SSAM are based on a PET threshold of 10 seconds, which aligns with the threshold used in LiDAR Insight Platform. These conflicts are assessed across the entire corridor, including the intersecting roads at the studied intersections. The results of the ANOVA tests identifying significant parameters for both the AM and PM peak hours, are as summarized in Table 12.

**Table 12 Parameter Selection Results**

| Parameter                               | Tested Thresholds | AM Peak               |                    | PM Peak               |                    |
|-----------------------------------------|-------------------|-----------------------|--------------------|-----------------------|--------------------|
|                                         |                   | Conflict Rate p-value | Critical Parameter | Conflict Rate p-value | Critical Parameter |
| Standstill Distance                     | 0, 2.7, 10        | $8.15e^{-10}$         | Yes                | $3.25e^{-21}$         | Yes                |
| Minimum Distance Headway                | 0.1, 5, 500       | $9.72e^{-3}$          | Yes                | $3.03e^{-4}$          | Yes                |
| Safety Distance Reduction Factor        | 0.1, 2.55, 5      | $9.54e^{-1}$          | No                 | $9.29e^{-1}$          | No                 |
| Accepted Deceleration (Lane Changing)   | -0.01, -4, -10    | $8.78e^{-1}$          | No                 | $9.79e^{-1}$          | No                 |
| Accepted Deceleration (Following)       | -0.01, -4, -10    | $9.96e^{-1}$          | No                 | $8.79e^{-1}$          | No                 |
| Look-ahead distance (Maximum)           | 50, 100, 300      | $2.32e^{-1}$          | No                 | $1.76e^{-2}$          | Yes                |
| Look-ahead distance (Observed)          | 10, 30, 50        | $9.75e^{-1}$          | No                 | $9.97e^{-1}$          | No                 |
| Temporary lack of attention duration    | 0.1, 2.5, 5       | $1.97e^{-3}$          | Yes                | $8.31e^{-8}$          | Yes                |
| Temporary lack of attention probability | 5, 15, 25         | $5.49e^{-4}$          | Yes                | $4.28e^{-17}$         | Yes                |
| Desired position at free flow           | Any, Left, Right  | $2.47e^{-5}$          | Yes                | $1.60e^{-6}$          | Yes                |
| Lateral Distance Driving                | 0.3, 0.6, 0.8     | 1                     | No                 | 1                     | No                 |
| Lateral Distance Standstill             | 0.3, 0.8, 1.0     | 1                     | No                 | 1                     | No                 |
| Reaction Time Distribution              | 0.5, 0.9, 2.0     | 1                     | No                 | 1                     | No                 |

#### 4.5 Performance and Safety Model Calibration

Conflict-sensitive parameters were further refined through a trial-and-error process aimed at accurately estimating values based on observed conflicts. This method involved systematically adjusting the parameters within predefined ranges and closely analyzing the resulting simulated conflicts to ensure proximity with real-world data. Each modification was tested through multiple simulation runs by utilizing VISSIM COM, allowing for efficient automation. After calibrating the parameters, the validation process included comparing the calibrated parametric values against literature. The calibrated performance and safety outcomes, along with comparison to literature, are presented in Table 13.

**Table 13 Performance and Safety Calibrated Model Parametric Values**

| Parameter                                      | Tested Range     | Increment | AM Peak          | PM Peak | Literature <sup>1</sup> |
|------------------------------------------------|------------------|-----------|------------------|---------|-------------------------|
|                                                |                  |           | Calibrated Value |         |                         |
| Standstill Distance (m)                        | 0 to 2.7         | 0.18      | 2.70             | 2.34    | 2.68                    |
| Minimum Distance Headway (m)                   | 0.1 to 5         | 0.32      | 4.58             | 4.58    | 1.60                    |
| Look-ahead distance (Maximum) (m)              | 50 to 300        | 17.0      | Default          | 203     | 156                     |
| Temporary lack of attention duration (seconds) | 0.1 to 2.5       | 0.16      | 0.42             | 1.22    | 1.36 (1)                |
| Temporary lack of attention probability (%)    | 5 to 15          | 0.5       | 5                | 11.30   | 11.86 (5)               |
| Desired position at free flow                  | Any, Left, Right | N/A       | Left             | Left    | Any or Right            |

**Notes:** 1. Parametric values based on literature are based on averaged value of more than two sources (Abuamer et al., 2019; Brill & Brink, 2016; CDM Smith, 2014; Deluka Tibljaš et al., 2018; Essa & Sayed, 2020b; Liang et al., 2020; Lokesh et al., 2020; Ulak et al., 2019). Where the range is wide, most common value is included between brackets.

#### 4.5.1 Standstill Distance

The standstill distance parameter, which represents the distance a vehicle maintains while stationary, was tested within a range of 0 to 2.70 metres, with increments of 0.18 metres. Through this iterative process, the calibrated standstill distance value for the AM peak period was determined to be 2.70 metres, while for the PM peak period resulted to be 2.34 metres. Notably, the safety-calibrated standstill distance is larger than the performance-calibrated value, reflecting less severe conflicts and prioritizing safety over operational efficiency.

#### 4.5.2 Minimum Distance Headway

The minimum distance headway parameter, which defines the smallest allowable space gap between two vehicles, was tested in increments of 0.32 metres across a range of 0.1 to 5 metres. The calibrated value is determined to be 4.58 metres for both AM and PM peak periods, a value deviating from typical values found in literature as well as the operational-calibrated values. This deviation arises from the focus on safety calibration rather than operational efficiency. Although shorter time headways are common in operational models, they reduce the safety buffer between vehicles, increasing the risk of conflicts during lane changes (Vogel, 2003). In contrast, the safety-calibrated minimum distance headway of 4.58 metres aligns more closely with the observed longitudinal conflicts and PET values, which are encroached into higher PET values, hence indicating less severe conflicts. Due to the tendency of microsimulation to generate higher number of longitudinal conflicts than observed, creating an overestimate of potential safety risks, the increase of minimum distance headway to 4.58 metres introduces greater spacing between vehicles to reduce this overestimation error.

#### 4.5.3 Look-Ahead Distance (Maximum)

The maximum look-ahead distance, indicating the furthest distance a driver considers when making decisions, was tested within a range of 50 to 300 metres, with increments of 17 metres. The calibrated value for the PM peak period was 203 metres, whereas the default value of 250 metres was retained for the AM peak period. This suggests that during the PM peak, drivers tend to look further ahead, due to higher traffic volumes or different driving conditions.

#### 4.5.4 Temporary Lack of Attention (Duration/Probability)

The duration for which a driver temporarily lacks attention was tested within a range of 0.1 to 2.5 seconds, with increments of 0.16 seconds. The calibrated values were 0.42 seconds for the AM peak period and 1.22 seconds for the PM peak period. The default value for temporary lack of attention is 0 seconds, indicating no temporary lack of attention in the default model. The calibrated parameter however indicates that drivers are more likely to experience longer lapses in attention during the PM peak period compared to the AM peak.

The probability that a driver experiences a temporary lack of attention was tested within a range of 5 percent to 15 percent, with increments of 0.5 percent. The calibrated value for the AM peak period was 5 percent, while for the PM peak period resulted to be 11.30 percent. The default probability of 0 percent reflects no such lapses in attention. The calibrated probability for suggests that while temporary lapses in attention occur yet infrequent.

#### 4.5.5 Desired Position at Free Flow

The desired position of a vehicle at free flow, which can be to the left, right, or any position within the lane, was tested without a specific range or increment. The calibrated value for both the AM and PM peak periods was the left position. The desired position at free flow is set at middle position for the default model. While the calibrated parameter indicates a consistent preference among drivers to position their vehicles towards the left side of the lane during free-flow conditions.

### 4.6 Conflicts Validation

#### 4.6.1 Distribution of Post-Encroachment Time (PET)

##### 4.6.1.1 Combined Conflict Types

The evaluation of PET distribution figures at each intersection, comparing observed and simulated surrogate measure of combined conflict types, is presented in Figure 62. This figure reflects data from two peak hours over the entire study month, resulting in a total of 60 peak-hour observations, and with PET threshold of 10 seconds to meet the settings of LiDAR Insight Platform.

As shown in Figure 62, a consistent discrepancy exists between observed and modeled conflicts. The observed distributions are more complex and often bimodal, while

the default and calibrated models produce simpler, skewed distributions towards lower PET values. This discrepancy highlights the limitations of the models to intricate vehicle interactions at intersections.

The discrepancy may also be linked to the intersection geometry and traffic volumes. At high-volume intersections like Laviolette, Francois Xavier, and Chenaux – where two major roads intersect, or minor roads are two-way – conflicts tend to be overestimated. Conversely, at lower-volume intersections like Angele, Paul, and Nicolas Perrot, involving commercial driveways or one-way roads, conflicts are underestimated. Nicolas Perrot intersection, however, added complexity due to its small commercial site and multiple access points to the south.

Each calibration stage from the default model to the only safety model (Calibration 3), demonstrates a gradual shift in the distribution. Although the distributions remain skewed towards lower PET values, later calibrations exhibit a slightly broader spread, indicating better capture of the presented variability in the observed data. Notably, in the only safety model (Calibration 3), there is an evident increase in the variability of PET distribution, particularly in intersections such as Chenaux. Although the calibrated models do not fully replicate the bimodal nature of the observed data, there is refinement in the peaks of the distributions. The later calibrations tend to have slightly more defined peaks compared to the smoother distributions seen in the default model, indicating progress towards recognizing the multiplicity of interaction patterns.

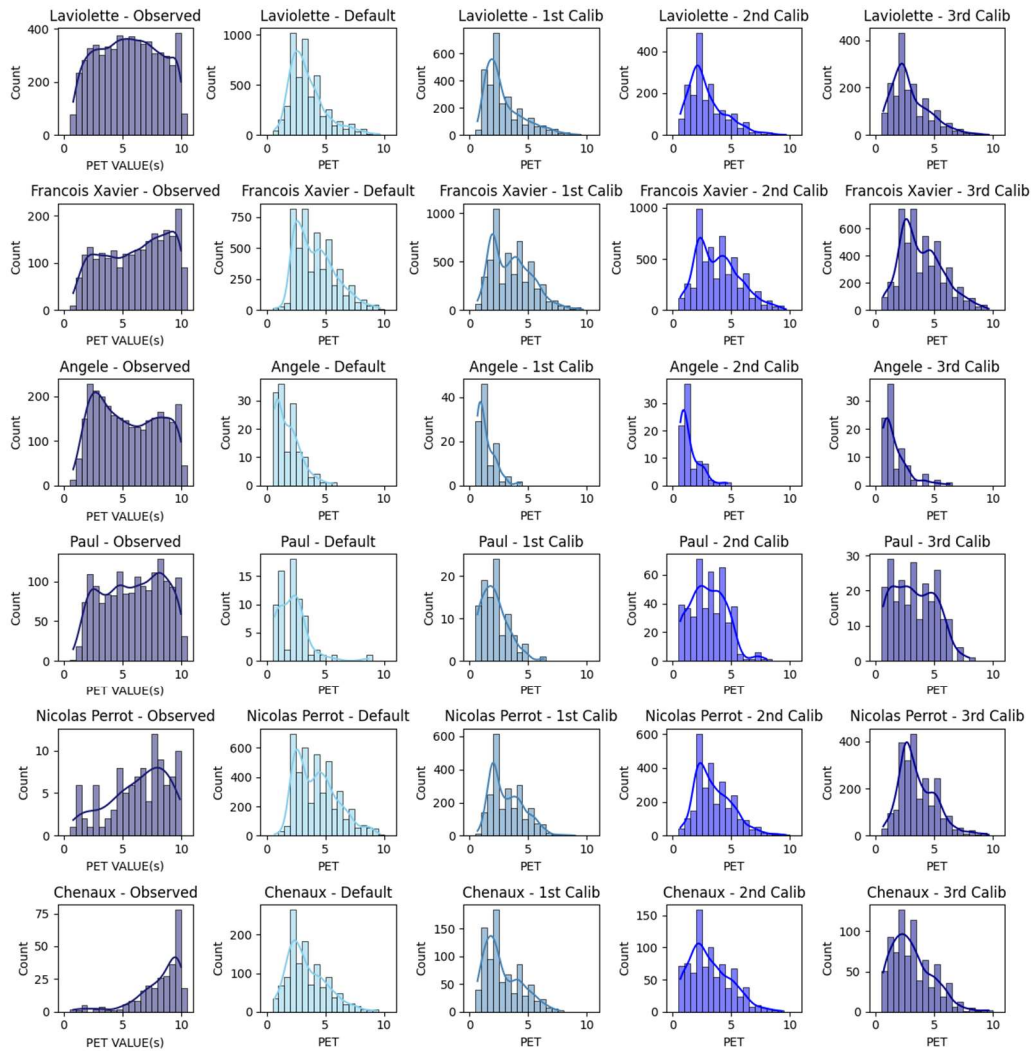


**Figure 62 Combined Conflicts PET Distribution per Intersection**

#### 4.6.1.2 Conflict Type Specific

The assessment of simulated conflicts at each intersection, categorized by conflict type, is conducted over the study month, and encompassing two peak hours, resulting in a total of 60 peak hour observations. Both crossing and longitudinal conflicts are analyzed using a PET threshold of 10 seconds to align with the LiDAR Insight Platform. The total number of observed conflicts reflects those recorded during the combined peak periods throughout the study month.

Each intersection along the corridor presented challenges in replicating observed PET distributions. As shown in Figure 63, default models tended to concentrate conflicts in the lower PET ranges, hence underestimating the variability present in real-world conditions. Meanwhile, the transition from performance (Calibration 1) to only safety model (Calibration 3) improved the representation of PET distribution, yet difficulties in

capturing multiple peaks and the full spread of observed data persisted. Progress across intersections varied; Laviolette, Francois Xavier, Nicolas Perrot and Chenaux displayed more relatively balanced distributions, whereas Angele and Paul intersections struggled to reproduce the peaks diverse spread of the observed data.

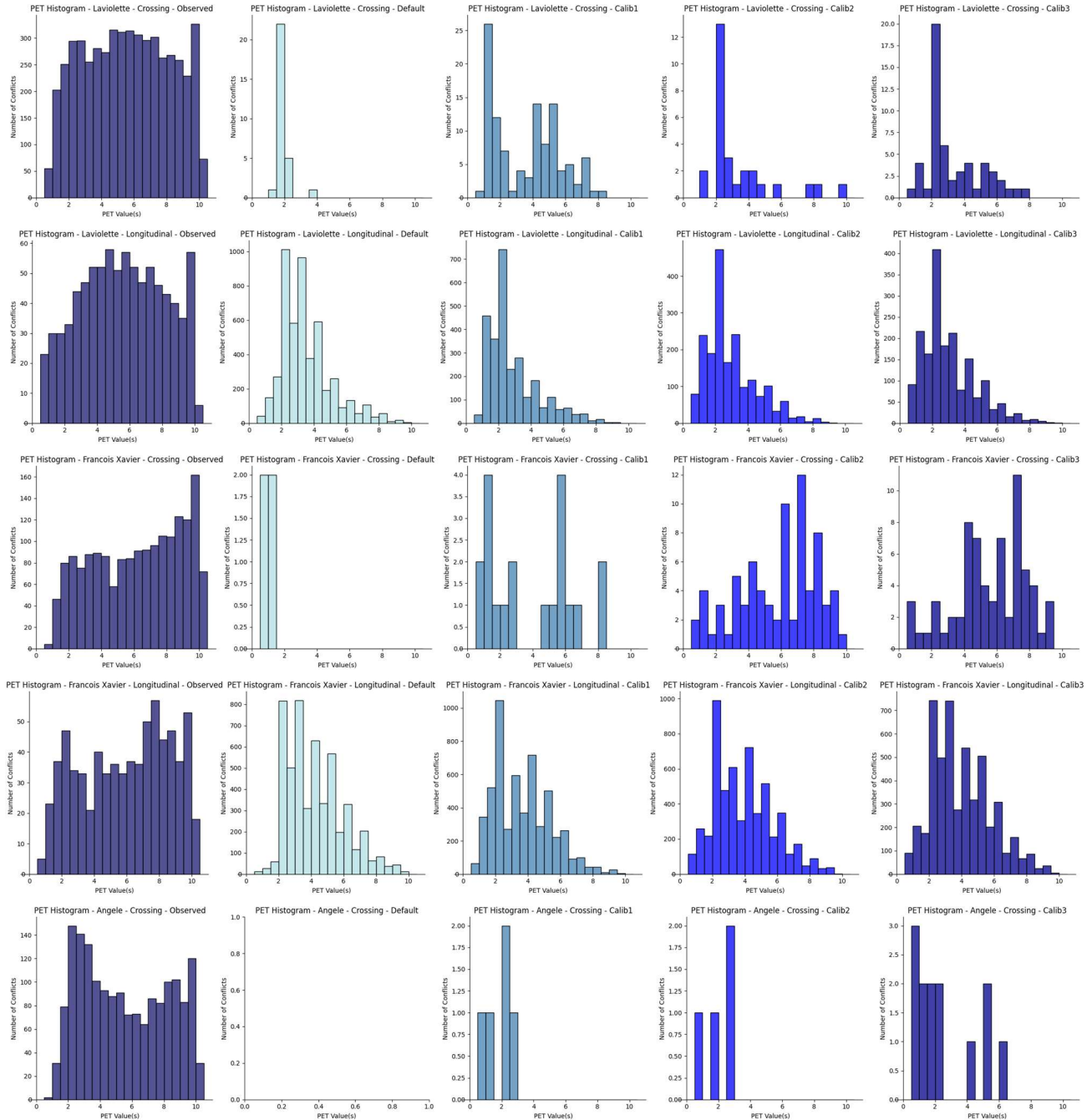


Figure 63 Conflict Type PET Distribution per Intersection

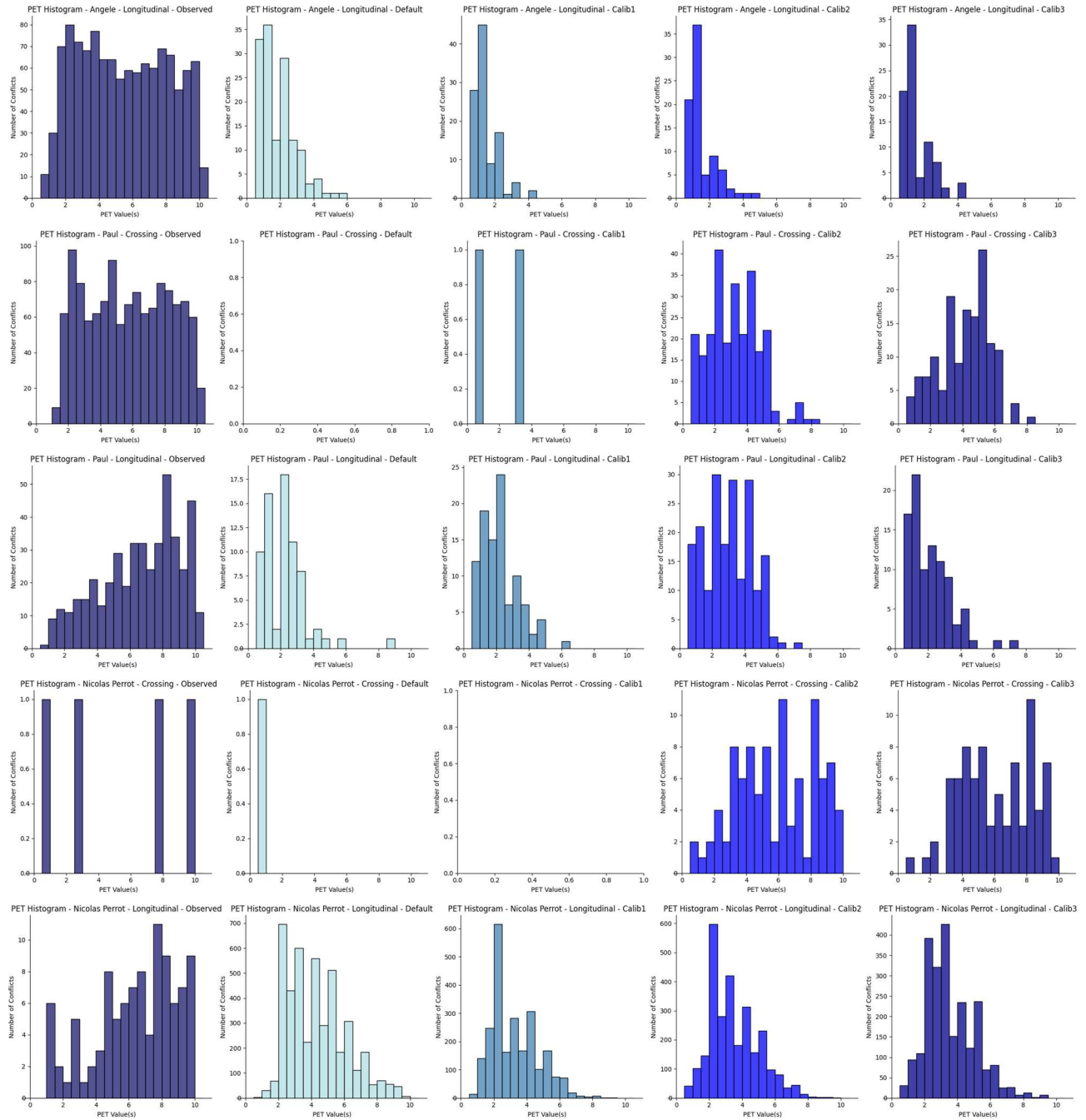


**Figure 63 Continued Conflict Type PET Distribution per Intersection**

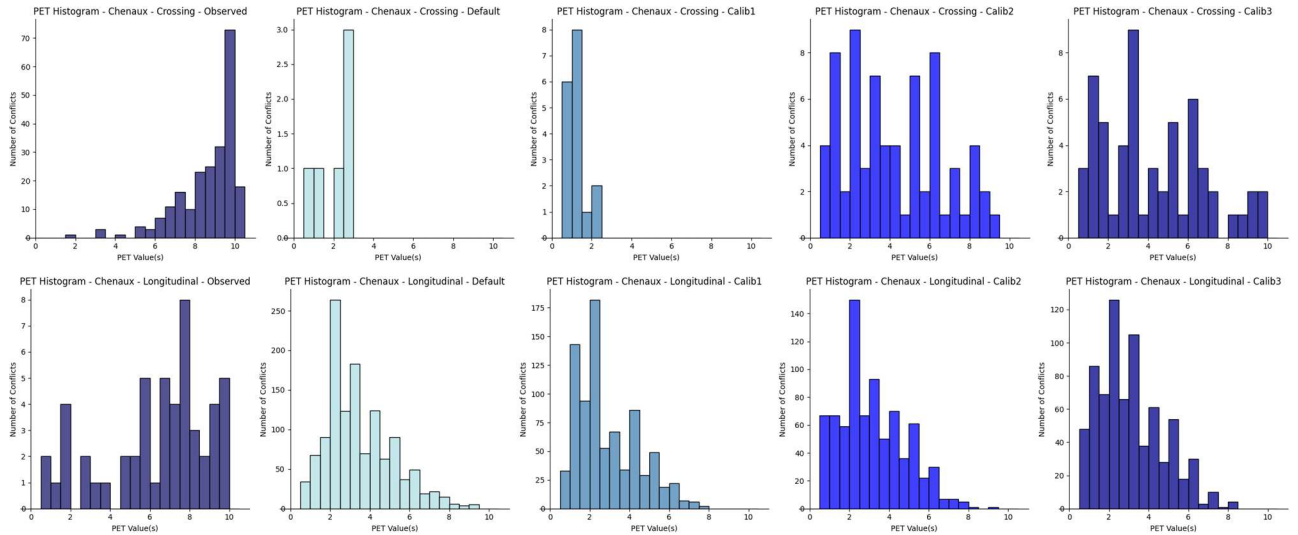


**Figure 63 Continued Conflict Type PET Distribution per Intersection**

#### 4.6.2 Distribution of Conflict Types

The analysis of all intersections indicates a consistent trend where longitudinal conflicts are consistently overestimated, while crossing conflicts are consistently underestimated in all simulated models. This discrepancy may be due to complex behaviours at signalized intersections, such as illegal turns, turning on red lights, and drivers not respecting right-of-way rules – behaviours that are challenging to accurately simulate with typical VISSIM parameters. Although the models struggle to accurately predict the observed crossing conflicts, the performance and safety model (Calibration 2) provides the closest approximation to the observed number of crossing conflicts. Meanwhile, the only safety model (Calibration 3) yields the lowest number of longitudinal conflicts among the simulated models, making it the most suitable for comparison with observed data. However, despite this improvement, the number of longitudinal conflicts in the only safety model (Calibration 3) remains higher than the observed values.

The small number of observed crossing conflicts at the Nicolas Perrot intersection – primarily functioning with north-south manoeuvring due to a residential roadway on the west leg and a driveway at the east leg – aligns with the expected traffic dynamics at this location, with only four observed crossing conflicts during the combined peak hours.

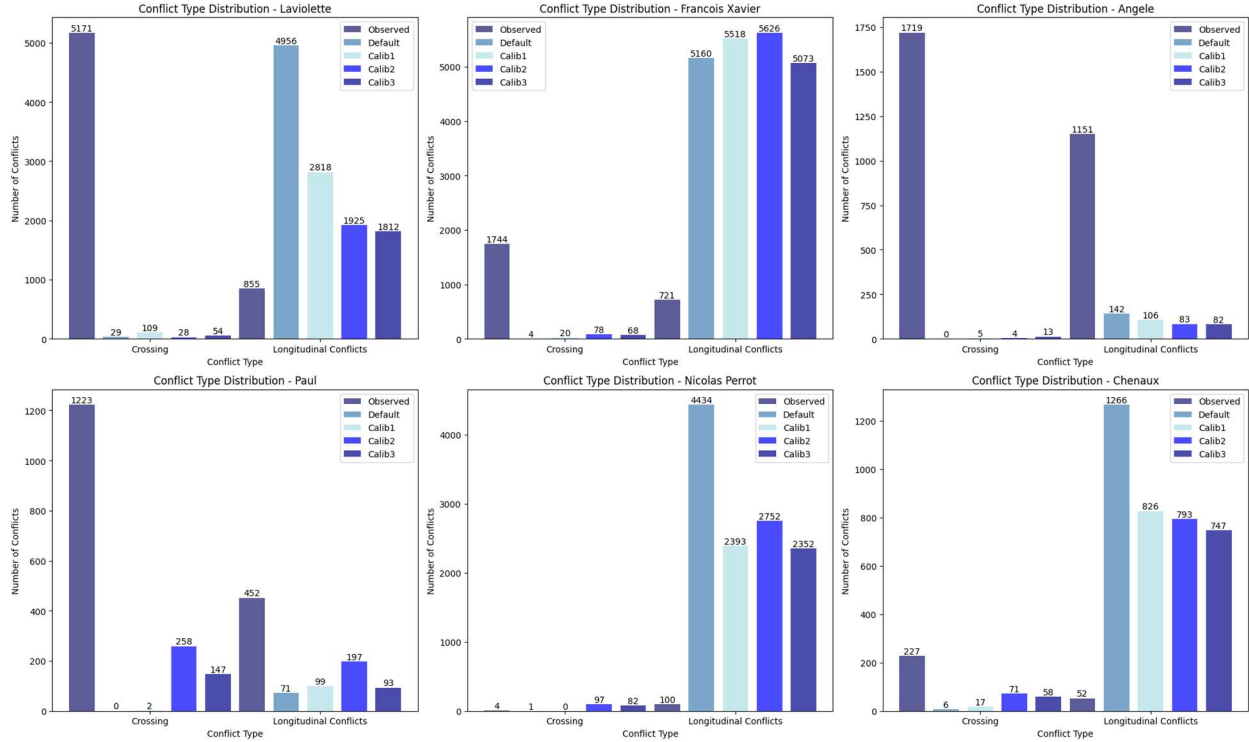


**Figure 64 Conflict Type Distribution per Intersection**

#### 4.6.3 Spatial Distribution of Conflicts

The heat map in Figure 65 illustrates observed conflicts pattern of conflicts along the corridor, compiled over the study period of one month during the AM and PM peak hours, totaling 60 peak-hour observations. The default model, shown in Figure 66, exhibits significantly fewer crossing conflicts with a sparse distribution, failing to capture the concentrated patterns evident in the observed heat map. This indicates limitations of default model to predict actual traffic interactions.

Upon applying the performance model (Calibration 1), shown in Figure 67, there is an increase in the number of conflicts compared to the default model; however, the distribution remains inadequate, particularly at critical locations like crossings. The performance and safety model (Calibration 2), shown in Figure 68, presents a significant enhancement, with an increased number of conflicts and a distribution defining more crossing conflicts as closely aligned with the observed data. The only safety model (Calibration 3), shown in Figure 69, closely resembles the performance and safety model (Calibration 2) with only minor improvements in the representation of crossing conflicts at Angele intersection.

The progression from the performance (Calibration 1) to only safety (Calibration 3) model, illustrate a clear trajectory of increased realism in representing conflicts.



**Figure 65 Observed Conflicts Heat Map**

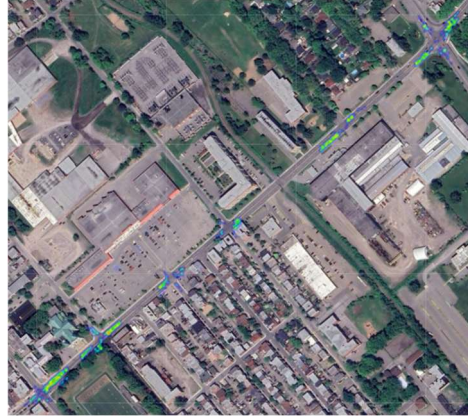


**Figure 66 Default Conflicts Heat Map**

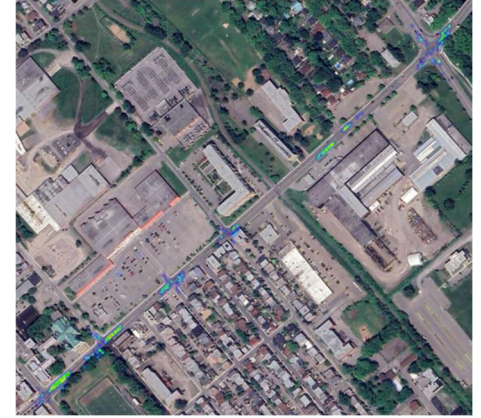


**Figure 67 Performance Calibrated Conflicts Heat Map**



**Figure 68 Performance and Safety Calibrated Conflicts Heat Map**

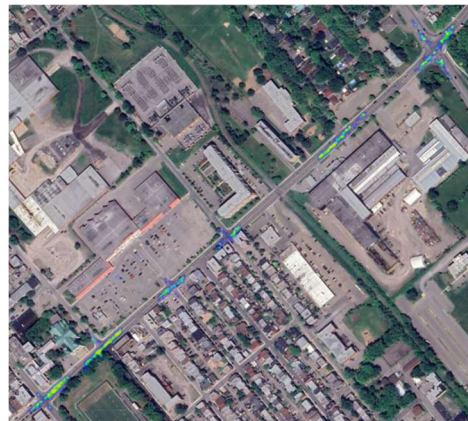


**Figure 69 Only Safety Calibrated Conflicts Heat Map**

## 4.7 Effectiveness of Calibration

### 4.7.1 Error Degree

The analysis of simulation models demonstrates that calibration significantly enhances accuracy, as evidenced by the reductions in MANE values across peak hours and directions. During the AM peak for northbound traffic, the performance model (Calibration 1) shows a notable improvement over the default model, reducing the MANE from 0.469 to 0.239. The performance and safety model (Calibration 2) achieves an improved MANE of 0.229, resulting to be the most accurate model. Remarkably, the only safety model (Calibration 3) performs well in predicting operational outcomes, with a MANE of 0.251. In the PM peak for northbound direction, the performance and safety model (Calibration 2) leads slightly with a MANE of 0.244, followed closely by the only safety model (Calibration 3) at 0.228. Such results confirm that the only safety model (Calibration 3) can effectively predict traffic operations. The performance model (Calibration 1) also

shows improvement with a MANE of 0.236, though it is not as strong as the other two calibrations.

For the southbound direction during the AM peak, all the three calibrated models perform similarly with a MANE ranging from 0.264 to 0.266. Moreover, the only safety calibrated model (Calibration 3) proves that safety-focused model does not compromise performance accuracy. In the PM peak, although the performance and safety model (Calibration 2) shows a slight increase in MANE compared to performance model (Calibration 1), all the calibrated models still perform much better than the default model.

Overall, the performance and safety model (Calibration 2) provides the closest accuracy. However, the only safety model (Calibration 3) demonstrates an ability to effectively predict operational outcomes, although the model primarily focuses on safety.

**Table 14 Operational MANE Results**

| Approach   | Peak Hour | MANE        | Simulated Model |               |               |               |
|------------|-----------|-------------|-----------------|---------------|---------------|---------------|
|            |           |             | Default         | Calibration 1 | Calibration 2 | Calibration 3 |
| Northbound | AM        | Speed       | 0.275           | 0.131         | 0.103         | 0.106         |
|            |           | Travel Time | 0.194           | 0.108         | 0.126         | 0.145         |
|            |           | Total       | 0.469           | 0.239         | 0.229         | 0.251         |
|            | PM        | Speed       | 0.223           | 0.123         | 0.120         | 0.116         |
|            |           | Travel Time | 0.114           | 0.113         | 0.124         | 0.112         |
|            |           | Total       | 0.337           | 0.236         | 0.244         | 0.228         |
| Southbound | AM        | Speed       | 0.241           | 0.105         | 0.104         | 0.106         |
|            |           | Travel Time | 0.144           | 0.161         | 0.160         | 0.158         |
|            |           | Total       | 0.385           | 0.266         | 0.264         | 0.264         |
|            | PM        | Speed       | 0.221           | 0.139         | 0.122         | 0.119         |
|            |           | Travel Time | 0.287           | 0.209         | 0.228         | 0.230         |
|            |           | Total       | 0.508           | 0.348         | 0.350         | 0.349         |

#### 4.7.2 Effectiveness Rate

The AM peak period, Table 15, necessitates significant parameter adjustments in comparing the calibrated models against the default settings. These adjustments have substantial impacts on both operational effectiveness and safety. Notably, the standstill distance in the calibrated models exhibits considerable discrepancies, with a 20 percent change in performance model and a 35 percent in the safety model. While this adjustment has a marginal effect on performance by 0.8 percent, the safety calibrated value influence safety by 5.7 percent. Additionally, the minimum distance headway demonstrates a 26 percent disparity in the operational model. Conversely, the safety model displays a notable difference compared to the default model. These adjustments underscore the importance of this parameter in influencing operational performance, although with a minor effect on safety by 1.4 percent. Other parameters, such as the additive and multiplicative components of the safety distance and the desired position at free flow, also manifest notable variances between the default and calibrated models, with percentage differences ranging from of 50

percent for the additive part and 43 percent for the multiplicative part. Although these parameters individually have minimal direct effects on operation or safety, their cumulative impact on model performance is pivotal when considered in conjunction with other parameters.

Despite certain parameters individually exhibiting a 0 percent effect rate, their combination with other parameters can contribute to total effectiveness, underscoring the importance of comprehensive sensitivity analysis that encompasses parameter combinations to capture the full scope of their impact on model performance.

**Table 15 AM Peak Parameter Effectiveness Rate**

| Parameter                               | Default | Calibrated<br>(Operational, A) | Calibrated<br>(Safety, B) | Absolute<br>Difference |      | percent<br>Difference |     | percent<br>Effect on<br>Operation |      | percent<br>Effect on<br>Safety |      |
|-----------------------------------------|---------|--------------------------------|---------------------------|------------------------|------|-----------------------|-----|-----------------------------------|------|--------------------------------|------|
|                                         |         |                                |                           | A                      | B    | A                     | B   | A                                 | B    | A                              | B    |
| Standstill Distance                     | 2       | 1.6                            | 2.7                       | 0.4                    | 0.7  | 20%                   | 35% | 0.8%                              | 1%   | 5.6%                           | 5.7% |
| Minimum Distance Headway                | 0.5     | 0.63                           | 4.58                      | 0.13                   | 4.08 | 26%                   | HI  | 0%                                | 0.2% | 0%                             | 1.4% |
| Additive parts of safety distance       | 2       | 1                              | -                         | 1                      | -    | 50%                   | -   | 0.1%                              | -    | 0%                             | -    |
| Multiplicative parts of safety distance | 3       | 1.7                            | -                         | 1.3                    | -    | 43%                   | -   | 0.3%                              | -    | 0.4%                           | -    |
| Temporary lack of attention duration    | 0       | -                              | 0.42                      | -                      | 0.42 | -                     | N/A | -                                 | 0%   | -                              | 0%   |
| Temporary lack of attention probability | 0       | -                              | 5                         | -                      | 5    | -                     | N/A | -                                 | 0%   | -                              | 0%   |
| Desired Position at free flow           | Middle  | -                              | Left                      | -                      | N/A  | -                     | N/A | -                                 | 0.1% | -                              | 89%  |
| Safety Distance Reduction Factor        | 0.6     | 0.92                           | -                         | 0.32                   | -    | 53%                   | -   | 0%                                | -    | 0.1%                           | -    |
| Accepted Deceleration (Lane Changing)   | -1      | -0.93                          | -                         | 0.07                   | -    | 7%                    | -   | 0%                                | -    | 0%                             | -    |
| Accepted Deceleration (Following)       | -1      | -0.92                          | -                         | 0.08                   | -    | 8%                    | -   | 0%                                | -    | 0%                             | -    |
| Maximum Deceleration (Leading)          | -4      | -3.54                          | -                         | 0.46                   | -    | 12%                   | -   | 0%                                | -    | 0%                             | -    |
| Maximum Deceleration (Following)        | -3      | -3.52                          | -                         | 0.52                   | -    | 17%                   | -   | 0.1%                              | -    | 0%                             | -    |

During the PM peak, Table 16, parameter calibration similarly results in significant deviations from the default model, impacting operational and safety aspects. The standstill distance undergoes adjustments with smaller absolute differences compared to the morning peak but still presents with 5 percent and 17 percent for the operational and safety models, respectively. This results in a modest impact on both operation by 1.9 percent and safety

by 1 percent. Like the AM peak, the minimum headway parameter displays a 22 percent difference for the operational calibrated model, while it is a significant difference for the safety calibrated model. The look-ahead distance shows a 19 percent difference comparing default and safety model, with the parameter contributes to safety by 1.1 percent. Parameters such as the desired position at free flow and the safety distance reduction factor maintain their significance. Notably, the positioning of the traveling vehicle to the left of lane during free flow influences the safety model by 98 percent.

**Table 16 PM Peak Parameter Effectiveness Rate**

| Parameter                               | Default | Calibrated<br>(Operational. A) | Calibrated<br>(Safety, B) | Absolute<br>Difference |      | percent<br>Difference |     | percent<br>Effect on<br>Operation |      | percent<br>Effect on<br>Safety |      |
|-----------------------------------------|---------|--------------------------------|---------------------------|------------------------|------|-----------------------|-----|-----------------------------------|------|--------------------------------|------|
|                                         |         |                                |                           | A                      | B    | A                     | B   | A                                 | B    | A                              | B    |
| Standstill Distance                     | 2       | 1.9                            | 2.34                      | 0.1                    | 0.34 | 5%                    | 17% | 1.9%                              | 1.5% | 1%                             | 2.1% |
| Minimum Distance Headway                | 0.5     | 0.61                           | 4.58                      | 0.11                   | 4.08 | 22%                   | HI  | 0%                                | 1.7% | 0%                             | 0.8% |
| Additive parts of safety distance       | 2       | 1                              | -                         | 1                      | -    | 50%                   | -   | 0.5%                              | -    | 1.4%                           | -    |
| Multiplicative parts of safety distance | 3       | 1.3                            | -                         | 1.7                    | -    | 57%                   | -   | 1.2%                              | -    | 0.8%                           | -    |
| Look-ahead distance (Maximum)           | 250     | -                              | 203                       | -                      | 47   | 19%                   | -   | 2%                                | -    | 1.1%                           | -    |
| Temporary lack of attention duration    | 0       | -                              | 1.22                      | -                      | 1.22 | -                     | N/A | -                                 | 0%   | -                              | 0%   |
| Temporary lack of attention probability | 0       | -                              | 11.3                      | -                      | 11.3 | -                     | N/A | -                                 | 0%   | -                              | 0%   |
| Desired Position at free flow           | Middle  | -                              | Left                      | -                      | N/A  | -                     | N/A | -                                 | 5%   | -                              | 98%  |
| Safety Distance Reduction Factor        | 0.6     | 0.93                           | -                         | 0.33                   | -    | 55%                   | -   | 0.2%                              | -    | 0.1%                           | -    |
| Accepted Deceleration (Lane Changing)   | -1      | -0.92                          | -                         | 0.08                   | -    | 8%                    | -   | 0%                                | -    | 0%                             | -    |
| Accepted Deceleration (Following)       | -1      | -0.89                          | -                         | 0.11                   | -    | 11%                   | -   | 0%                                | -    | 0.3%                           | -    |
| Maximum Deceleration (Leading)          | -4      | -3.51                          | -                         | 0.49                   | -    | 12%                   | -   | 0%                                | -    | 0%                             | -    |
| Maximum Deceleration (Following)        | -3      | -3.18                          | -                         | 0.18                   | -    | 6%                    | -   | 0%                                | -    | 0%                             | -    |

The model effectiveness was evaluated with the recognition that certain parameters, as shown in Table 15 and Table 16, demonstrated no measurable effect on operational performance or safety when assessed individually. This observation emphasizes the necessity of calibrating parameters in combination, as some parameters exhibit their influence only when partnered with complementary parameters. Thus, the model

effectiveness in Table 17 and Table 18 aimed to explore the interplay between the collective impact of calibrated parameters on operation and safety.

The analysis of operational and safety effectiveness across different calibrations reveals clear trends during the AM and PM peak periods. The effectiveness rates quantify the model improvements in both operational performance – measured through travel time and speed – or safety which is assessed through the average PET.

During the AM peak, as shown in Table 17, the performance model (Calibration 1), which focuses on operational factors, demonstrated a 36 percent improvement in operational performance compared to default model. This result is notably higher compared to 19 percent operational improvement achieved by performance and safety model (Calibration 2) and 15 percent improvement by the only safety model (Calibration 3). These findings suggest that performance model (Calibration 1) is the most effective when operational performance is the primary objective. In terms of safety, the only safety model (Calibration 3) yields the highest effectiveness, with a 49 percent improvement in safety outcomes, slightly exceeding the 48 percent improvement obtained from the performance and safety model (Calibration 2). Both safety-concentrated models outperform the performance model (Calibration 1), which provides a comparatively modest 18 percent improvement in safety. Although the only safety model (Calibration 3) enhances safety, it does so at the cost of operational performance. The performance and safety model (Calibration 2) offers a more balanced approach.

During the PM peak, Table 18, the results reveal a slightly different pattern. The performance model (Calibration 1) achieves 31 percent operational effectiveness, marginally surpassing by the performance and safety model (Calibration 2) at 29 percent, and further improved by the only safety model (Calibration 3) at 27 percent. Although these differences are minor, all calibrated models perform better compared to the default model. In terms of safety outcomes, the only safety model (Calibration 3) demonstrates highest improvement over the default model, with a 54 percent increase in safety effectiveness. This performance slightly exceeds the 53 percent improvement achieved by the performance and safety model (Calibration 2) and significantly exceeds the 16 percent improvement by the performance model (Calibration 1).

Overall, the results indicate that the performance model (Calibration 1) is most effective for operational efficiency, while the only safety model (Calibration 3) excels in safety outcomes. The performance and safety model (Calibration 2), with its balanced approach, offers a practical compromise by providing satisfactory performance in both operational and safety metrics. This makes it a versatile choice for scenarios where both aspects need to be considered.

**Table 17 AM Peak Model Effectiveness Rate**

| <b>Operational Effectiveness</b> |         |               |               |               |
|----------------------------------|---------|---------------|---------------|---------------|
|                                  | Default | Calibration 1 | Calibration 2 | Calibration 3 |
| Default                          | -       | 36%           | 19%           | 15%           |
| Calibration 1                    | -       | -             | 20%           | 24%           |
| Calibration 2                    | -       | -             | -             | 4%            |
| <b>Safety Effectiveness</b>      |         |               |               |               |
|                                  | Default | Calibration 1 | Calibration 2 | Calibration 3 |
| Default                          | -       | 18%           | 48%           | 49%           |
| Calibration 1                    | -       | -             | 37%           | 38%           |
| Calibration 2                    | -       | -             | -             | 2%            |

**Table 18 PM Peak Model Effectiveness Rate**

| <b>Operational Effectiveness</b> |         |               |               |               |
|----------------------------------|---------|---------------|---------------|---------------|
|                                  | Default | Calibration 1 | Calibration 2 | Calibration 3 |
| Default                          | -       | 31%           | 29%           | 27%           |
| Calibration 1                    | -       | -             | 2%            | 4%            |
| Calibration 2                    | -       | -             | -             | 3%            |
| <b>Safety Effectiveness</b>      |         |               |               |               |
|                                  | Default | Calibration 1 | Calibration 2 | Calibration 3 |
| Default                          | -       | 16%           | 53%           | 54%           |
| Calibration 1                    | -       | -             | 45%           | 45%           |
| Calibration 2                    | -       | -             | -             | 0.5%          |

#### 4.7.3 Overall Assessment

As shown in Table 19, the MANE for speed decreased significantly following the performance model (Calibration 1), reducing from 0.24 in the default model to 0.12, with further slight improvement to 0.11 in both performance and safety (Calibration 2) and only safety (Calibration 3) models. This highlights the effectiveness of performance model in enhancing speed prediction accuracy. In contrast, the travel time MANE showed modest reductions, decreasing from 0.18 to 0.15 after the performance model (Calibration 1), but regressing slightly to 0.16 in the subsequent calibrated models.

For crossing conflicts, the predicted number of conflicts increased, rising from 40 in the default model to 153 in the performance model (Calibration 1), then peaking at 536 in the performance and safety model (Calibration 2), yet slightly declining to 422 in the only safety model (Calibration 3). This significant rise reflects the improved sensitivity of the calibrated models in detecting crossing conflicts, which were underrepresented in the default model. The 85<sup>th</sup> percentile PET for crossing conflicts also increased 2.00 seconds in the default models to 5.67 seconds after the performance model (Calibration 1), and further to 7.00 seconds in both performance and safety (Calibration 2) and only safety (Calibration 3) models. This progression suggests a closer alignment with real-world

conflict severity, capturing more low risk interactions. Additionally, PET variance increased from 0.32 in default model to 4.51 after the performance model (Calibration 1), peaking at 5.36 in the performance and safety model (Calibration 2) then slightly decreasing to 4.85 in the only safety model (Calibration 3), indicating that the calibrated models captures a wider range of conflict severity. Furthermore, the PET MANE consistently improved across calibrations, decreasing from 0.70 in the default model to 0.21 by the only safety model (Calibration 3), further demonstrating enhanced accuracy of PET predictions for crossing conflicts.

For longitudinal conflicts, the total number steadily decreased throughout the calibration process, dropping from 16,029 in the default model to 10,159 in the only safety model (Calibration 3). This reduction suggests that the calibrations refined the ability to predict longitudinal conflicts, mitigating the tendency of simulated models to overestimate conflicts in linear traffic movements. The 85<sup>th</sup> percentile PET for longitudinal conflicts remained stable, fluctuating slightly around 5.33 seconds, indicating that predictions were already closely aligned with observed values. Similarly, PET variance showed minimal changes, reflecting consistent conflict severity across models. However, PET MANE increased from 0.34 to 0.44 after the performance model (Calibration 1), prior to stabilizing at 0.39 in both performance and safety (Calibration 2) and only safety (Calibration 3) models.

**Table 19 Modelled Corridor Results Summary**

|                                      | <b>Observed</b> | <b>Default</b> | <b>Calibration 1</b> | <b>Calibration 2</b> | <b>Calibration 3</b> |
|--------------------------------------|-----------------|----------------|----------------------|----------------------|----------------------|
| <b>Operation</b>                     |                 |                |                      |                      |                      |
| Speed MANE                           | -               | 0.24           | 0.12                 | 0.11                 | 0.11                 |
| Travel Time MANE                     | -               | 0.18           | 0.15                 | 0.16                 | 0.16                 |
| <b>Safety</b>                        |                 |                |                      |                      |                      |
| <b><u>Crossing Conflicts</u></b>     |                 |                |                      |                      |                      |
| Total Number                         | 10,088          | 40             | 153                  | 536                  | 422                  |
| 85 <sup>th</sup> Percentile PET      | 8.90            | 2.00           | 5.67                 | 7.00                 | 7.00                 |
| PET Variance                         | -               | 0.32           | 4.51                 | 5.36                 | 4.85                 |
| PET MANE                             | -               | 0.70           | 0.44                 | 0.28                 | 0.21                 |
| <b><u>Longitudinal Conflicts</u></b> |                 |                |                      |                      |                      |
| Total Number                         | 3,331           | 16,029         | 11,760               | 11,376               | 10,159               |
| 85 <sup>th</sup> Percentile PET      | 8.80            | 5.67           | 5.00                 | 5.33                 | 5.33                 |
| PET Variance                         | -               | 2.98           | 2.75                 | 3.01                 | 3.02                 |
| PET MANE                             | -               | 0.34           | 0.44                 | 0.39                 | 0.39                 |

## 4.8 Discussion

In the thesis, it was essential to compare the ratio of simulated conflicts with findings from existing literature to provide a broader contextual understanding of discrepancies between simulated and observed conflicts. The conflict ratios were derived using the operation and safety model (Calibration 2), which was determined to offer a balanced assessment of both operational performance and safety outcomes. For the comparison, literature involving simulated conflicts generated from VISSIM trajectory files was standardized by converting conflict data into conflicts per hour. Although many studies rely on TTC as a surrogate safety measure, only studies that applied the highest acceptable TTC threshold recognized by SSAM were included, ensuring a fair comparison of total conflicts between the thesis and the literature. In studies utilizing PET as the surrogate measure, the corresponding PET threshold was applied to the thesis' simulated conflicts to maintain consistency in the evaluation.

Igene et al. (2024) analyzed six intersections using TTC without a defined threshold, finding a range of 15 to 30 rear-end conflicts per hour, with an average of 22 rear-end conflicts per hour. By contrast, the thesis examined longitudinal conflicts across a corridor which ranged from 1 to 94 simulated conflicts per hour, with an average of 32 longitudinal conflicts per hour. The observed conflicts in the thesis ranged from 1 to 19 longitudinal conflicts per hour, averaging 9 longitudinal conflicts per hour, a lower rate when compared to the simulated conflicts.

Ansariyar & Taherpour (2023) applied a PET threshold of 5 seconds to assess conflicts at a single intersection over a five-month period. To align with the one-month study period of the thesis, the conflict ratio from their study was divided by five, yielding an average of 97 conflicts per hour. The total number of simulated conflicts in the thesis, using a PET threshold of 5 seconds, was divided by six to calculate an average for each intersection, and then further divided by 30 to account for the number of simulations runs over the one-month period. This resulted in an average of 27 simulated conflicts per hour, a value lower than both 97 conflicts per hour extract from literature, but higher than the LiDAR observed conflict rate of 16 conflicts per hour.

Rajeswaran et al. (2023) reported an average of 179 conflicts per hour across 91 intersections in the City of Toronto, using a PET threshold of 5 seconds. For a comparable analysis, the simulated conflicts across six intersections consisted of dividing the total conflicts by 60 runs—30 runs for each peak hour—yielding an average of 163 conflicts per hour. The observed conflict rate from the thesis using LiDAR is 94 conflicts per hour, showing a notable difference when compared to the simulated conflicts and Rajeswaran's 179 conflicts per hour.

Table 20 summarizes the variation in conflict outcomes across literature, observed through LiDAR system, and simulated conflicts. The simulated conflicts align with the literature ranges.

**Table 20 Comparison of Conflict Ratios Summary**

| Literature                    | Conflict Measure      | Conflicts per Hour |               |                    |
|-------------------------------|-----------------------|--------------------|---------------|--------------------|
|                               |                       | Literature         | Simulated     | Observed via LiDAR |
| (Ansariyar & Taherpour, 2023) | PET $\leq$ 5 seconds  | 97                 | 27            | 16                 |
| (Igene et al., 2024)          | TTC $\leq$ 10 seconds | 15 – 30            | 1 – 94        | 1 – 19             |
|                               | PET $\leq$ 10 seconds | (Average: 22)      | (Average: 32) | (Average: 9)       |
| (Rajeswaran et al., 2023)     | PET $\leq$ 5 seconds  | 179                | 163           | 94                 |

## 4.9 Limitations

### 4.9.1 Constraints of LiDAR Data

The extraction of observed conflicts from the CSV files by LiDAR Insight Platform introduced several challenges that may have compromised the accuracy of the dataset. A selective review of vehicle interaction videos, which is limited to crossing conflicts with PET threshold of 2 seconds, revealed instances of concurrent conflicts, particularly rear-end interactions, which were not captured in the CSV. Many of these unrecorded conflicts were situated along the corridor or near the center of intersections, suggesting gaps in the source of data collection. A beneficial approach that can be undertaken by the LiDAR system used for this thesis is the extension of PET threshold up to 10 seconds to yield more comprehensive data check.

Furthermore, the CSV file exclusively categorized conflicts as crossing, despite evidence from heatmaps and spatial analyses that indicated longitudinal conflicts occurring along the lanes. This highlights a potential limitation in the conflict classification of LiDAR system. Although a Python code was developed to manually classify conflicts based on their geocoordinates, the introduction of advanced automated conflict identification would enhance reliability.

Another constraint was the absence of conflict angle data, which would have facilitated the calibration process in SSAM. As a result, conflict angles must be adjusted manually through iterative trials until simulated crossing conflicts were not as underestimated as per default angles. Future LiDAR configurations that include conflict angles would rationalize calibration and improve model accuracy.

### 4.9.2 Microsimulation Limitations

Microsimulation models typically assume idealized driver behavior, where the drivers respond promptly and consistently to the predefined parameters. However, the real-world behavior is more variable, influenced by factors such as traffic violations and

distractions, often resulting in delayed or inconsistent reactions. These behavioral discrepancies can cause the underestimation of conflicts.

Additionally, calibration tends to focus on replicating more critical conflicts, such as those with PET values below 5 seconds (Killi & Vedagiri, 2014). Consequently, interactions with longer PET values may not be accurately represented, leading to biased conflict estimations.

## 5 Conclusion

Microscopic traffic simulation has become a valuable tool for understanding and improving roadway safety. Contrary to the reactive approach taken by the traditional road safety techniques such as the Roadside Safety Program (RSM), microscopic traffic simulation offers proactivity by assessing road safety through replicating the intricate movements and interactions of individual vehicles. By offering real-time insights into traffic behaviour, microscopic traffic simulation allows researchers and planners to explore safety measures and contributed parameters in detail. However, the utilization of microscopic traffic simulation often oversimplify vehicle dynamics leading to failure in capturing the full complexity of traffic conflicts. Additionally, when paired with computer vision technique for gathering road user data, the risk of misclassification increases. The misclassification of road users can introduce inaccurate calibrated parameters, undermining the reliability of the simulation outcomes.

To address limitations associated with camera vision, recent technological advancements particularly LiDAR sensors, offer a propitious data collection method including precise trajectory information and detailed dimensions of road users. The incorporation of LiDAR data into the calibration process of microsimulation model holds the potential to enhance the accuracy of parametric values, thereafter, allowing for a better representation of real-world driving behaviours. Accurate representation of driving behaviours requires a crucial evaluation of calibrated parameters to obtain the optimal potential in predicting traffic conflicts. Historically, calibration has focused primarily on operational metrics, such as travel time and speed, rather than on safety-related targets. Hence, a key concern arises regarding whether parameters adjusted for performance metrics are equally suitable for simulating traffic conflicts accurately. This thesis research attempted to bridge this gap by introducing a methodology that considers both operational and safety metrics during the calibration process, while utilizing collected traffic data from LiDAR sensors. Additionally, the thesis expanded on assessing the effectiveness of each calibrated parameter to provide a comprehensive understanding of both operational performance and safety implications.

The findings of this thesis contributed to the growing knowledge on microscopic traffic simulation and roadway safety assessment by demonstrating the potential of LiDAR data, as well as, balancing operational and safety metrics during the calibration process.

### 5.1 Contributions and Key Findings

The thesis introduced a calibration methodology that employs a stepwise approach, addressing the need to consider both operational and safety performance measures. The method showed some promising results in terms of reduction in the Mean Absolute Normalized Error (MANE), demonstrating marked improvements compared to default microscopic simulation models. The calibration process progresses through three phases: an initial focus on operational accuracy, followed by a combined emphasis on both

operational and safety factors, and culminating in a final stage centered exclusively on safety. These phases revealed how prioritizing different calibration targets could influence the accuracy of speed, travel time, and traffic conflicts predictions. In evaluating the effectiveness of calibration, the thesis conducts an analysis of parameters affecting both operational and safety outcomes. Key factors such as standstill distance and minimum distance headway proved to be crucial in shaping model performance, while desired position at flow was shown to impact safety. A comprehensive sensitivity analysis revealed that some model parameters, although having limited influence individually, have substantial impact in combination. This underscores the importance of selecting appropriate parameter combinations to enhance the accuracy and reliability of microscopic simulation models.

Furthermore, this research also made progress in estimating traffic conflicts, particularly crossing conflicts. Improvements observed in the second and only safety calibrated models were more noticeable as compared to real-world conditions. The performance and safety calibrated model, which balances both operational and safety accuracy, proves to be more effective. Quantifying the trade-offs between operational and safety improvements constitutes a significant contribution. The performance calibrated model excel in optimizing operational metrics, delivering improved performance in travel time and speed accuracy. Conversely, the only safety calibrated model prioritizes safety outcomes, enhancing the capability of traffic microscopic simulation to predict and capture conflicts. The performance and safety calibrated model provides a balanced solution, allowing for flexibility in prioritizing model outcomes based on operational and safety needs.

## References

- Abay, K. A., Paleti, R., & Bhat, C. R. (2013). The joint analysis of injury severity of drivers in two-vehicle crashes accommodating seat belt use endogeneity. *Transportation Research Part B: Methodological*, 50, 74–89. <https://doi.org/10.1016/j.trb.2013.01.007>
- Abdel-Aty, M., & Pande, A. (2007). Crash data analysis: Collective vs. individual crash level approach. *Journal of Safety Research*, 38(5), 581–587. <https://doi.org/10.1016/j.jsr.2007.04.007>
- Abdel-Aty, M., Wang, Z., Zheng, O., & Abdelraouf, A. (2023). Advances and applications of computer vision techniques in vehicle trajectory generation and surrogate traffic safety indicators. *Accident Analysis & Prevention*, 191, 107191. <https://doi.org/10.1016/j.aap.2023.107191>
- Abdelraouf, A., Abdel-Aty, M., Wang, Z., & Zheng, O. (2022). *Trajectory Prediction for Vehicle Conflict Identification at Intersections Using Sequence-to-Sequence Recurrent Neural Networks* (arXiv:2210.08009). arXiv. <https://doi.org/10.48550/arXiv.2210.08009>
- Abdulhafedh, A. (2017). Road Traffic Crash Data: An Overview on Sources, Problems, and Collection Methods. *Journal of Transportation Technologies*, 7(2), Article 2. <https://doi.org/10.4236/jtts.2017.72015>
- Abuamer, I. M., Himpe, W., Raza, S., & Tampere, C. M. J. (2019). *Systematic Identification of Key Parameters and their Influence Range for Calibration of VISSIM Microscopic Simulator*. <https://lirias.kuleuven.be/retrieve/540163>
- Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 22(5), 717–727. [https://doi.org/10.1016/S0731-7085\(99\)00272-1](https://doi.org/10.1016/S0731-7085(99)00272-1)

- Alfonsi, R., Persia, L., Tripodi, A., & Usami, D. (2016). Advancements in Road Safety Management Analysis. *Transportation Research Procedia*, 14, 2064–2073. <https://doi.org/10.1016/j.trpro.2016.05.105>
- Al-Ghandour, M. N., Schroeder, B. J., Williams, B. M., & Rasdorf, W. J. (2011). Conflict Models for Single-Lane Roundabout Slip Lanes from Microsimulation: Development and Validation. *Transportation Research Record*, 2236(1), 92–101. <https://doi.org/10.3141/2236-11>
- Allen, B. L., Shin, B. T., & Cooper, P. J. (1978). ANALYSIS OF TRAFFIC CONFLICTS AND COLLISIONS. *Transportation Research Record*, 667, Article HS-025 846. <https://trid.trb.org/view/85806>
- Almqvist, S., Hyden, C., & Risser, R. (1991). USE OF SPEED LIMITERS IN CARS FOR INCREASED SAFETY AND A BETTER ENVIRONMENT. *Transportation Research Record*, 1318. <https://trid.trb.org/view/365564>
- Amundsen, F. H., & Hyden, C. (1977). *Proceedings: First Workshop on Traffic Conflicts*. Norwegian Council for Scientific and Industrial Research.
- Ansariyar, A. (2023). *Analyzing Vehicle-Vehicle and Vehicle-Pedestrian Conflicts at Signalized Intersections: Exploring The Influence of Vehicle Angle on Conflict Severity Through LiDAR Sensor During Traffic Signal Failures* (SSRN Scholarly Paper 4631890). <https://doi.org/10.2139/ssrn.4631890>
- Ansariyar, A., & Taherpour, A. (2023). Investigating the accuracy rate of vehicle-vehicle conflicts by LIDAR technology and microsimulation in VISSIM and AIMSUN. *Advances in Transportation Studies*, 61, 37–52.

- Archer, J. (2005). *Indicators for traffic safety assessment and prediction and their application in micro-simulation modelling*:
- Autey, J., Sayed, T., & Zaki, M. H. (2012). Safety evaluation of right-turn smart channels using automated traffic conflict analysis. *Accident Analysis & Prevention*, 45, 120–130. <https://doi.org/10.1016/j.aap.2011.11.015>
- Bachmann, C., Roorda, M. J., & Abdulhai, B. (2011). Improved Time-to-Collision Definition for Simulating Traffic Conflicts on Truck-Only Infrastructure. *Transportation Research Record*, 2237(1), 31–40. <https://doi.org/10.3141/2237-04>
- Ballard, D. H., & Brown, C. M. (1982). *Computer Vision*. Prentice-Hall.
- Barnett, A. G., van der Pols, J. C., & Dobson, A. J. (2005). Regression to the mean: What it is and how to deal with it. *International Journal of Epidemiology*, 34(1), 215–220. <https://doi.org/10.1093/ije/dyh299>
- Berntman, M. (1994). *Metoder för insamling av uppgifter om svårt trafikskadade: Några källor och tekniker: Vol. Bulletin vägbyggnad, 6*. Lunds tekniska högskola, institutionen för teknik och samhälle, trafik och väg.
- Bhattarai, N., Zhang, Y., Liu, H., Pakzad, Y., & Xu, H. (2023). Proactive Safety Analysis Using Roadside LiDAR Based Vehicle Trajectory Data: A Study of Rear-End Crashes. *Transportation Research Record*, 03611981231182704. <https://doi.org/10.1177/03611981231182704>
- Bliss, T., & Breen, J. (2009). *Country Guidelines for the Conduct of Road Safety Management Capacity Reviews and the Specification of Lead Agency Reforms, Investment Strategies and Safe System Projects*.

- Boora, A., Ghosh, I., & Chandra, S. (2017). Identification of Free Flowing Vehicles on Two Lane Intercity Highways under Heterogeneous Traffic condition. *Transportation Research Procedia*, 21, 130–140. <https://doi.org/10.1016/j.trpro.2017.03.083>
- Brill, M., & Brink, H. P. (2016). *Micro simulation of roundabouts: An analysis of simulation parameters in Vissim*. <https://odr.chalmers.se/server/api/core/bitstreams/f6b5c15a-1f97-40bb-ae81-c3ef6928b060/content>
- Cafiso, S., D’Agostino, C., Bąk, R., & Kieć, M. (2018). Traffic conflicts analyses for 2+1 road sections. *MATEC Web of Conferences*, 231, 01006. <https://doi.org/10.1051/matecconf/201823101006>
- Canada, T. (2023, January 9). *2020 statistics on the social costs of collisions in Canada*. Transport Canada. <https://tc.canada.ca/en/road-transportation/statistics-data/statistics-data-road-safety/2020-statistics-social-costs-collisions-canada>
- Canada, T. (2024, May 2). *Canadian Motor Vehicle Traffic Collision Statistics: 2022*. Transport Canada. <https://tc.canada.ca/en/road-transportation/statistics-data/canadian-motor-vehicle-traffic-collision-statistics-2022>
- CDM Smith. (2014). *VISSIM Development and Calibration Report: US 231 Scottsville Road Scoping and Traffic Operations Study From I-65 to Lovers Lane*. <https://transportation.ky.gov/Planning/Planning%20Studies%20and%20Reports/Appendix%20B%20-%20VISSIM%20Development%20and%20Calibration%20Report.pdf>
- Chin, H.-C., & Quek, S.-T. (1997). Measurement of traffic conflicts. *Safety Science*, 26(3), 169–185. [https://doi.org/10.1016/S0925-7535\(97\)00041-6](https://doi.org/10.1016/S0925-7535(97)00041-6)

- Cui, H., Dong, J., Zhu, M., Li, X., & Wang, Q. (2022). Identifying accident black spots based on the accident spacing distribution. *Journal of Traffic and Transportation Engineering (English Edition)*, 9(6), 1017–1026. <https://doi.org/10.1016/j.jtte.2021.02.006>
- Cunto, F. (2008). *Assessing Safety Performance of Transportation Systems using Microscopic Simulation* [Doctoral Thesis, University of Waterloo]. <https://uwspace.uwaterloo.ca/handle/10012/4111>
- Da Costa, S., Qu, X., & Parajuli, P. M. (2015). A Crash Severity-Based Black Spot Identification Model. *Journal of Transportation Safety & Security*, 7(3), 268–277. <https://doi.org/10.1080/19439962.2014.911230>
- Dadashzadeh, N., Ergun, M., Kesten, S., & Žura, M. (2019a). An Automatic Calibration Procedure of Driving Behaviour Parameters in the Presence of High Bus Volume. *PROMET - Traffic&Transportation*, 31(5), 491–502. <https://doi.org/10.7307/ptt.v31i5.3100>
- Dadashzadeh, N., Ergun, M., Kesten, S., & Žura, M. (2019b). An Automatic Calibration Procedure of Driving Behaviour Parameters in the Presence of High Bus Volume. *PROMET - Traffic&Transportation*, 31(5), 491–502. <https://doi.org/10.7307/ptt.v31i5.3100>
- Davis, G. A., Hourdos, J., Xiong, H., & Chatterjee, I. (2011). Outline for a causal model of traffic conflicts and crashes. *Accident Analysis & Prevention*, 43(6), 1907–1919. <https://doi.org/10.1016/j.aap.2011.05.001>
- Deluka Tibljaš, A., Giuffrè, T., Surdonja, S., & Trubia, S. (2018). Introduction of Autonomous Vehicles: Roundabouts Design and Safety Performance Evaluation. *Sustainability*, 10(4), Article 4. <https://doi.org/10.3390/su10041060>

- Ding, S., Abdel-Aty, M., Wang, Z., & Wang, D. (2024). Insights into vehicle conflicts based on traffic flow dynamics. *Scientific Reports*, 14(1), 1536. <https://doi.org/10.1038/s41598-023-50017-3>
- Dupont, E., Martensen, H., Papadimitriou, E., Yannis, G., MUHLRAD, N., JAHI, H., Vallet, G., Giustiniani, G., PERSIA, L., BAX, C., Wijnen, W., Machata, K., BUTTLER, I., ZYSINSKA, M., Barnes, J., Gitelman, V., & Hakkert, A. (2011). *Deliverable 1.2 Road safety management investigation model and questionnaire*.
- El-Tantawy, S., Djavadian, S., Roorda, M. J., & Abdulhai, B. (2009). Safety Evaluation of Truck Lane Restriction Strategies Using Microsimulation Modeling. *Transportation Research Record*, 2099(1), 123–131. <https://doi.org/10.3141/2099-14>
- Elvik, R. (2002). The importance of confounding in observational before-and-after studies of road safety measures. *Accident Analysis & Prevention*, 34(5), 631–635. [https://doi.org/10.1016/S0001-4575\(01\)00062-8](https://doi.org/10.1016/S0001-4575(01)00062-8)
- Essa, M., & Sayed, T. (2015a). Simulated Traffic Conflicts: Do They Accurately Represent Field-Measured Conflicts? *Transportation Research Record: Journal of the Transportation Research Board*, 2514(1), 48–57. <https://doi.org/10.3141/2514-06>
- Essa, M., & Sayed, T. (2015b). Simulated Traffic Conflicts: Do They Accurately Represent Field-Measured Conflicts? *Transportation Research Record: Journal of the Transportation Research Board*, 2514(1), 48–57. <https://doi.org/10.3141/2514-06>
- Essa, M., & Sayed, T. (2020a). Comparison between Surrogate Safety Assessment Model and Real-Time Safety Models in Predicting Field-Measured Conflicts at Signalized Intersections. *Transportation Research Record: Journal of the Transportation Research Board*, 2674(3), 100–112. <https://doi.org/10.1177/0361198120907874>

- Essa, M., & Sayed, T. (2020b). Comparison between Surrogate Safety Assessment Model and Real-Time Safety Models in Predicting Field-Measured Conflicts at Signalized Intersections. *Transportation Research Record: Journal of the Transportation Research Board*, 2674(3), 100–112. <https://doi.org/10.1177/0361198120907874>
- Fan, R., Yu, H., Liu, P., & Wang, W. (2013a). Using VISSIM simulation model and Surrogate Safety Assessment Model for estimating field measured traffic conflicts at freeway merge areas. *IET Intelligent Transport Systems*, 7(1), 68–77. <https://doi.org/10.1049/iet-its.2011.0232>
- Fan, R., Yu, H., Liu, P., & Wang, W. (2013b). Using VISSIM simulation model and Surrogate Safety Assessment Model for estimating field measured traffic conflicts at freeway merge areas. *IET Intelligent Transport Systems*, 7(1), 68–77. <https://doi.org/10.1049/iet-its.2011.0232>
- FDOT. (2021). *Traffic Analysis Handbook*.
- Foomani, M. G. (2022). *Assessing the Impact of Active Signage Systems on Driving Behavior*.
- Gettman, D., & Head, L. (2003). *Surrogate safety measures from traffic simulation models* (FHWA-RD-03-050). <https://rosap.ntl.bts.gov/view/dot/1005>
- Gettman, D., Pu, L., Sayed, T., & Shelby, S. (2008). *Surrogate Safety Assessment Model and Validation: Final Report, June 2008—FHWA-HRT-08-051*. <https://www.fhwa.dot.gov/publications/research/safety/08051/08051.pdf>
- Glauz, W. D., & Migletz, D. J. (1980). APPLICATION OF TRAFFIC CONFLICT ANALYSIS AT INTERSECTIONS. *NCHRP Report*, 219, Article HS-028 882. <https://trid.trb.org/view/153539>

- Griffith, L., & Flowers, R. (1996). *A Discussion of Six Procedures for Evaluating Highway Safety Projects*.
- Guettinger, V. A. (1982). *From Accidents to Conflicts: Alternative Safety Measurement*.
- Guido, G., Saccomanno, F., Vitale, A., Astarita, V., & Festa, D. (2011). Comparing Safety Performance Measures Obtained from Video Capture Data. *Journal of Transportation Engineering*, 137(7), 481–491. [https://doi.org/10.1061/\(ASCE\)TE.1943-5436.0000230](https://doi.org/10.1061/(ASCE)TE.1943-5436.0000230)
- Hauer, E. (2004). Statistical Road Safety Modeling. *Transportation Research Record*, 1897, 81–87. <https://doi.org/10.3141/1897-11>
- Hawas, Y. E., Khan, M. B., & Maraqa, M. A. (2012). Analysis of the Effectiveness of the Road-Crash Database in the United Arab Emirates. *Journal of Transportation Safety & Security*, 4(3), 225.
- Heydari, S., Hickford, A., McIlroy, R., Turner, J., & Bachani, A. M. (2019). Road Safety in Low-Income Countries: State of Knowledge and Future Directions. *Sustainability*, 11(22), Article 22. <https://doi.org/10.3390/su11226249>
- Hidas, P., & Wagner, P. (2004). Review of Data Collection Methods for Microscopic Traffic Simulation. *Proceedings of WCTR*. World Conference on Transport Research (WCTR), Istanbul (Turkey). <https://elib.dlr.de/21358/>
- Highway Safety Manual: User Guide*. (2022). <https://nap.nationalacademies.org/catalog/26552/highway-safety-manual-user-guide>
- Highway Safety Manual—1st Edition*. (2009, April). <https://www.fpp.uni-lj.si/mma/HSM.pdf/2019060611143076/?m=1559812470>

- Hou, J., List, G. F., & Guo, X. (2014). New Algorithms for Computing the Time-to-Collision in Freeway Traffic Simulation Models. *Computational Intelligence and Neuroscience*, 2014, 761047. <https://doi.org/10.1155/2014/761047>
- Huang, F., Liu, P., Yu, H., & Wang, W. (2013). Identifying if VISSIM simulation model and SSAM provide reasonable estimates for field measured traffic conflicts at signalized intersections. *Accident Analysis & Prevention*, 50, 1014–1024. <https://doi.org/10.1016/j.aap.2012.08.018>
- Huang, H., Chin, H. C., & Haque, Md. M. (2008). Severity of driver injury and vehicle damage in traffic crashes at intersections: A Bayesian hierarchical analysis. *Accident Analysis & Prevention*, 40(1), 45–54. <https://doi.org/10.1016/j.aap.2007.04.002>
- Hyden, C. (1987). THE DEVELOPMENT OF A METHOD FOR TRAFFIC SAFETY EVALUATION: THE SWEDISH TRAFFIC CONFLICTS TECHNIQUE. *BULLETIN LUND INSTITUTE OF TECHNOLOGY, DEPARTMENT*, 70. <https://trid.trb.org/view/239059>
- Igene, M., Luo, Q., Jimenez, K., Soltanirad, M., Bataineh, T., & Liu, H. (2024). Integrating LiDAR Sensor Data into Microsimulation Model Calibration for Proactive Safety Analysis. *Sensors*, 24(13), Article 13. <https://doi.org/10.3390/s24134393>
- Imprialou, M., & Quddus, M. (2019). Crash data quality for road safety research: Current state and future directions. *Accident Analysis & Prevention*, 130, 84–90. <https://doi.org/10.1016/j.aap.2017.02.022>
- Ismail, K. A. (2010). *Application of computer vision techniques for automated road safety analysis and traffic data collection* [University of British Columbia]. <https://doi.org/10.14288/1.0062871>

- Ismail, K., Sayed, T., Saunier, N., & Lim, C. (2009). Automated Analysis of Pedestrian–Vehicle Conflicts Using Video Data. *Transportation Research Record: Journal of the Transportation Research Board*, 2140(1), 44–54. <https://doi.org/10.3141/2140-05>
- Ištoka Otković, I., Tollazzi, T., Šraml, M., & Varevac, D. (2023). Calibration of the Microsimulation Traffic Model Using Different Neural Network Applications. *Future Transportation*, 3(1), Article 1. <https://doi.org/10.3390/futuretransp3010010>
- James, H. F. (1991). UNDER-REPORTING OF ROAD TRAFFIC ACCIDENTS. *Traffic Engineering & Control*, 32(12). <https://trid.trb.org/view/365831>
- Jiang, R., Zhu, S., Chang, H., Wu, J., Ding, N., Liu, B., & Qiu, J. (2021). Determining an Improved Traffic Conflict Indicator for Highway Safety Estimation Based on Vehicle Trajectory Data. *Sustainability*, 13(16), Article 16. <https://doi.org/10.3390/su13169278>
- Khavas, R. G., Hellinga, B., & Zarinbal Masouleh, A. (2017). Identifying Parameters for Microsimulation Modeling of Traffic in Inclement Weather. *Transportation Research Record*, 2613(1), 52–60. <https://doi.org/10.3141/2613-07>
- Killi, D. V., & Vedagiri, P. (2014). Proactive Evaluation of Traffic Safety at An Unsignalized Intersection Using Micro- Simulation. *Journal of Traffic and Logistics Engineering*, 2(2), 140–145. <https://doi.org/10.12720/jtle.2.2.140-145>
- Kronprasert, N., Sutheerakul, C., Satiennam, T., & Luathep, P. (2021). Intersection Safety Assessment Using Video-Based Traffic Conflict Analysis: The Case Study of Thailand. *Sustainability*, 13(22), 12722. <https://doi.org/10.3390/su132212722>
- Kruysse, H. W. (1991). The subjective evaluation of traffic conflicts based on an internal concept of dangerousness. *Accident Analysis & Prevention*, 23(1), 53–65. [https://doi.org/10.1016/0001-4575\(91\)90035-4](https://doi.org/10.1016/0001-4575(91)90035-4)

- Laureshyn, A. (2010). *Application of automated video analysis to road user behaviour*.
- Laureshyn, A., & Várhelyi, A. (n.d.). *The Swedish Traffic Conflict Technique – Observer’s manual*.  
1.
- Leal, S. S., Maciel de Almeida, P. E., & Ribeiro, R. G. (2020). Calibrating Traffic Microscopic Simulation Model Parameters Using an Evolutionary Approach. *Transportation Research Procedia*, 48, 1038–1045. <https://doi.org/10.1016/j.trpro.2020.08.131>
- Lee, C., Hellinga, B., & Ozbay, K. (2006). Quantifying effects of ramp metering on freeway safety. *Accident Analysis & Prevention*, 38(2), 279–288.  
<https://doi.org/10.1016/j.aap.2005.09.011>
- Lee, S. E., Olsen, E. C. B., & Wierwille, W. W. (2004). *A Comprehensive Examination of Naturalistic Lane-Changes* [Dataset]. <https://doi.org/10.1037/e733232011-001>
- Lei, Q. (2021). *Semi-Automated Microscopic Traffic Flow Simulation Development Using Smart City Data* [Master Thesis, University of Waterloo].  
<https://uwspace.uwaterloo.ca/handle/10012/17234>
- Li, S., Xiang, Q., Ma, Y., Gu, X., & Li, H. (2016). Crash Risk Prediction Modeling Based on the Traffic Conflict Technique and a Microscopic Simulation for Freeway Interchange Merging Areas. *International Journal of Environmental Research and Public Health*, 13(11). <https://doi.org/10.3390/ijerph13111157>
- Liang, Q., Wan, Q., Bai, L., Yu, H., Lv, L., & Li, D. (2020). *Sensitivity of Simulated Conflicts to VISSIM Driver Behavior Parameter Modification*. Springerprofessional.De.  
<https://www.springerprofessional.de/en/sensitivity-of-simulated-conflicts-to-vissim-driver-behavior-par/17824592>

- Lokesh, Y., N S, C., & Tr, N. (2020). *Calibration and Validation of VISSIM Driving Behavior Parameters for Signalized and UN-Signalized Intersections in Bangalore City*.
- Long, K., Gao, Z., Yuan, Q., Xiang, W., & Hao, W. (2018). Safety evaluation for roadside crashes by vehicle–object collision simulation. *Advances in Mechanical Engineering*, 10(10), 1687814018805581. <https://doi.org/10.1177/1687814018805581>
- Lord, D., & Persaud, B. N. (2000). Accident Prediction Models With and Without Trend: Application of the Generalized Estimating Equations Procedure. *Transportation Research Record*, 1717(1), 102–108. <https://doi.org/10.3141/1717-13>
- Lu, C., He, X., van Lint, H., Tu, H., Happee, R., & Wang, M. (2021). Performance evaluation of surrogate measures of safety with naturalistic driving data. *Accident Analysis & Prevention*, 162, 106403. <https://doi.org/10.1016/j.aap.2021.106403>
- Lu, M. (2007). *Modelling and evaluation of the effects of traffic safety measures*.
- Mahmud, S. M., Ferreira, L., & Hojati, A. (2016). *Traditional Approaches to Traffic Safety Evaluation (TSE): Application Challenges and Future Directions*. 242–262. <https://doi.org/10.1061/9780784479810.028>
- Mahmud, S. M. S., Ferreira, L., Hoque, Md. S., & Tavassoli, A. (2017). Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs. *IATSS Research*, 41(4), 153–163. <https://doi.org/10.1016/j.iatssr.2017.02.001>
- Mallory, A., Kender, A., Watkins, L., Hutter, E., Moorhouse, K., & Transportation Research Center Inc. (2021). *Methods Used to Develop a Model for Crash and Injury Projections for 2020–2030* (DOT HS 813 147). <https://rosap.ntl.bts.gov/view/dot/58185>

- Mannering, F., Bhat, C. R., Shankar, V., & Abdel-Aty, M. (2020). Big data, traditional data and the tradeoffs between prediction and causality in highway-safety analysis. *Analytic Methods in Accident Research*, 25, 100113. <https://doi.org/10.1016/j.amar.2020.100113>
- Mannering, F., Shankar, V., & Bhat, C. R. (2016). Unobserved heterogeneity and the statistical analysis of highway accident data. *Analytic Methods in Accident Research*, 11, 1–16. <https://doi.org/10.1016/j.amar.2016.04.001>
- McGehee, D. V., Mazzae, E. N., & Baldwin, G. H. S. (2000). Driver Reaction Time in Crash Avoidance Research: Validation of a Driving Simulator Study on a Test Track. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 44(20), 3-320-3–323. <https://doi.org/10.1177/154193120004402026>
- Mehdizadeh, A., Cai, M., Hu, Q., Alamdar Yazdi, M. A., Mohabbati-Kalejahi, N., Vinel, A., Rigdon, S. E., Davis, K. C., & Megahed, F. M. (2020). A Review of Data Analytic Applications in Road Traffic Safety. Part 1: Descriptive and Predictive Modeling. *Sensors (Basel, Switzerland)*, 20(4), 1107. <https://doi.org/10.3390/s20041107>
- Mehendale, N., & Neoge, S. (2020). *Review on Lidar Technology* (SSRN Scholarly Paper 3604309). <https://doi.org/10.2139/ssrn.3604309>
- Meng, Q., & Weng, J. (2011). Evaluation of rear-end crash risk at work zone using work zone traffic data. *Accident Analysis & Prevention*, 43(4), 1291–1300. <https://doi.org/10.1016/j.aap.2011.01.011>
- Mimbela, L. E., & Klein, L. (2007, August 31). *A Summary of Vehicle Detection and Surveillance Technologies use in Intelligent Transportation Systems*. <https://www.fhwa.dot.gov/policyinformation/pubs/vdstits2007/05.cfm>

- Ministry of Transportation in Ontario. (2012, March). *Ontario Traffic Manual (OTM)—Book 12—Traffic Signals*. Sydney PROD.  
<https://www.library.mto.gov.on.ca/SydneyPLUS/Sydney/Portal/default.aspx?component=AAAAIY&record=59cabe78-8aaf-4347-95ab-d6c066099015>
- Mohamed, H. A., Puan, O. C., Hassan, S. A., & Azhari, S. F. (2021). Calibration and Validation of Lane Changing Following Gap Distance in VISSIM. *IOP Conference Series: Materials Science and Engineering*, 1153(1), 012016. <https://doi.org/10.1088/1757-899X/1153/1/012016>
- Montoya-Alcaraz, M., Mungaray-Moctezuma, A., Calderón-Ramírez, J., García, L., & Martínez-Lazcano, C. (2020). Road Safety Analysis of High-Risk Roads: Case Study in Baja California, México. *Safety*, 6(4), Article 4. <https://doi.org/10.3390/safety6040045>
- Muhlrad, N. (with Unité mixte de recherche épidémiologique et de surveillance transport, travail, environnement). (2009). *Road safety management systems: A comprehensive diagnosis method adaptable to low and middle income countries*. INRETS.
- Mullakkal-Babu, F. A., Wang, M., Farah, H., van Arem, B., & Happee, R. (2017). Comparative Assessment of Safety Indicators for Vehicle Trajectories on Highways. *Transportation Research Record*, 2659(1), 127–136. <https://doi.org/10.3141/2659-14>
- Ngeni, F., Mwakalonge, J., & Siuhi, S. (2024). Solving traffic data occlusion problems in computer vision algorithms using DeepSORT and quantum computing. *Journal of Traffic and Transportation Engineering (English Edition)*, 11(1), 1–15. <https://doi.org/10.1016/j.jtte.2023.05.006>

- Nguyen, H., Taneerananon, P., & Luatthep, P. (2016). Approach to Identifying Black Spots Based on Potential Saving in Accident Costs. *Engineering Journal*, 20, 109–122. <https://doi.org/10.4186/ej.2016.20.2.109>
- Oketch, T., & Carrick, M. (2005). *Calibration and Validation of a Micro-Simulation Model in Network Analysis*.
- Opromolla, R., & Fasano, G. (2021). Visual-based obstacle detection and tracking, and conflict detection for small UAS sense and avoid. *Aerospace Science and Technology*, 119, 107167. <https://doi.org/10.1016/j.ast.2021.107167>
- Orsini, F., Gecchele, G., Rossi, R., & Gastaldi, M. (2021). A conflict-based approach for real-time road safety analysis: Comparative evaluation with crash-based models. *Accident Analysis & Prevention*, 161, 106382. <https://doi.org/10.1016/j.aap.2021.106382>
- Ozbay, K., Yang, H., Bartin, B., & Mudigonda, S. (2008). Derivation and Validation of New Simulation-Based Surrogate Safety Measure. *Transportation Research Record*, 2083(1), 105–113. <https://doi.org/10.3141/2083-12>
- Paliotto, A., Meocci, M., Terrosi, A., & La Torre, F. (2024). Systematic review, evaluation and comparison of different approaches for the implementation of road network safety analysis. *Heliyon*, 10(7), e28391. <https://doi.org/10.1016/j.heliyon.2024.e28391>
- Papadimitriou, E., Yannis, G., Muhlrads, N., Vallet, G., Butler, I., Gitelman, V., Doveh, E., Dupont, E., Thomas, P., Giustiniani, G., Talbot, R., Machata, K., & Bax, C. (2012). *Analysis of road safety management in the European countries. Deliverable 1.5 Vol.II of the EC FP7 project DaCoTA*. <https://www.semanticscholar.org/paper/Analysis-of-road-safety-management-in-the-European-Papadimitriou-Yannis/8873498ab7d4e749de018d8ce3b9755197582657>

- Park, B. (Brian), & Qi, H. (Maggie). (2005). Development and Evaluation of a Procedure for the Calibration of Simulation Models. *Transportation Research Record*, 1934(1), 208–217. <https://doi.org/10.1177/0361198105193400122>
- Park, B., & Qi, M. (2005). Development and Evaluation of a Procedure for the Calibration of Simulation Models. *Transportation Research Record*, 1934(1), 208–217. <https://doi.org/10.1177/0361198105193400122>
- Park, C., Jeong, N.-T., Yu, D., & Hwang, S.-H. (2019). Path Generation Algorithm Based on Crash Point Prediction for Lane Changing of Autonomous Vehicles. *International Journal of Automotive Technology*, 20(3), 507–519. <https://doi.org/10.1007/s12239-019-0048-1>
- Parker Jr., M. R., & Zegeer, C. V. (1989, January). *Traffic Conflict Techniques for Safety and Operations: Observers Manual*. U.S. Department of Transportation Federal Highway Administration.
- Peesapati, L. N., Hunter, M. P., & Rodgers, M. O. (2018). Can post encroachment time substitute intersection characteristics in crash prediction models? *Journal of Safety Research*, 66, 205–211. <https://doi.org/10.1016/j.jsr.2018.05.002>
- Periyasamy, N. 1, Lynch, C. A. 2, Dharmaratne, S. D. 3, Nugegoda, D. B. 4, Østbye, T. 5 1 P. V. S., Department of Global Health, S. of P. H., & Evaluation, U. of W. (2013). *Under reporting of road traffic injuries in the district of Kandy, Sri Lanka*. e003640. <https://doi.org/10.1136/bmjopen-2013-003640>
- Perkins, S. R., & Harris, J. I. (1968). *Traffic conflict characteristics-accident potential at intersections*.

- Persaud, B. N., Griffith, M. S., Hayden, C. M., Kononov, J., Lewis II, C. R., Pain, R. F., Salzberg, P. M., Scopatz, R. A., Stewart, D. E., & Washington, S. P. (2001). *NCHRP Synthesis 295—Statistical Methods in Highway Safety Analysis*.
- Rajeswaran, T., Persaud, B., & Anarkooli, A. J. (2023). *Exploration of Some Key Issues in Developing and Applying Crash–Conflict Models for Signalized Intersections*. <https://journals-sagepub-com.proxy.lib.uwaterloo.ca/doi/full/10.1177/03611981221148483>
- Sacchi, E., Sayed, T., & deLeur, P. (2013). A comparison of collision-based and conflict-based safety evaluations: The case of right-turn smart channels. *Accident Analysis & Prevention*, 59, 260–266. <https://doi.org/10.1016/j.aap.2013.06.002>
- Salifu, M., & Ackaah, W. (2012). Under-reporting of road traffic crash data in Ghana. *International Journal of Injury Control and Safety Promotion*, 19(4), 331–339. <https://doi.org/10.1080/17457300.2011.628752>
- Salvucci, D. D., & Liu, A. (2002). The time course of a lane change: Driver control and eye-movement behavior. *Transportation Research Part F: Traffic Psychology and Behaviour*, 5(2), 123–132. [https://doi.org/10.1016/S1369-8478\(02\)00011-6](https://doi.org/10.1016/S1369-8478(02)00011-6)
- Sánchez, D. E., & Pany, T. (2019). *Analysis and Calibration of the VLP-16 LiDAR for Automotive Applications*. 2611–2621. <https://doi.org/10.33012/2019.17093>
- Saunier, N., & Sayed, T. (2007). Automated Analysis of Road Safety with Video Data. *Transportation Research Record*, 2019(1), 57–64. <https://doi.org/10.3141/2019-08>
- Saunier, N., Sayed, T., & Ismail, K. (2010). Large-Scale Automated Analysis of Vehicle Interactions and Collisions. *Transportation Research Record*, 2147(1), 42–50. <https://doi.org/10.3141/2147-06>

- Savolainen, P. T., Mannering, F. L., Lord, D., & Quddus, M. A. (2011). The statistical analysis of highway crash-injury severities: A review and assessment of methodological alternatives. *Accident Analysis & Prevention*, 43(5), 1666–1676. <https://doi.org/10.1016/j.aap.2011.03.025>
- Sayed, T. (2021, May 4). *Are all traffic conflict studies equal? – How to ensure reliability*. Advanced Mobility Analytics. <https://amagroup.io/are-all-traffic-conflict-studies-equal/>
- Sayed, T., Brown, G., & Navin, F. (1994). Simulation of traffic conflicts at unsignalized intersections with TSC-Sim. *Accident Analysis & Prevention*, 26(5), 593–607. [https://doi.org/10.1016/0001-4575\(94\)90021-3](https://doi.org/10.1016/0001-4575(94)90021-3)
- Sayed, T., Zaki, M. H., & Autey, J. (2013a). *A NOVEL APPROACH FOR DIAGNOSING ROAD SAFETY ISSUES USING AUTOMATED COMPUTER VISION TECHNIQUES*.
- Sayed, T., Zaki, M. H., & Autey, J. (2013b). Automated safety diagnosis of vehicle–bicycle interactions using computer vision analysis. *Safety Science*, 59, 163–172. <https://doi.org/10.1016/j.ssci.2013.05.009>
- Shahdah, U., Saccomanno, F., & Persaud, B. (2014). Integrated traffic conflict model for estimating crash modification factors. *Accident Analysis & Prevention*, 71, 228–235. <https://doi.org/10.1016/j.aap.2014.05.019>
- Shinar, D. (1984). The traffic conflict technique: A subjective vs. objective approach. *Journal of Safety Research*, 15(4), 153–157. [https://doi.org/10.1016/0022-4375\(84\)90046-X](https://doi.org/10.1016/0022-4375(84)90046-X)
- St-Aubin, P. G. (2011). *Traffic Safety Analysis for Urban Highway Ramps and Lane-Change Bans Using Accident Data and Video-Based Surrogate Safety Measures*.

- Sun, Y., Xu, H., Wu, J., Zheng, J., & Dietrich, K. M. (2018). 3-D Data Processing to Extract Vehicle Trajectories from Roadside LiDAR Data. *Transportation Research Record*, 2672(45), 14–22. <https://doi.org/10.1177/0361198118775839>
- Svensson, Å., & Hydén, C. (2006). Estimating the severity of safety related behaviour. *Accident Analysis & Prevention*, 38(2), 379–385. <https://doi.org/10.1016/j.aap.2005.10.009>
- The Swedish Traffic-Conflicts Technique.pdf*. (n.d.).
- Ulak, M. B., Ozguven, E. E., Moses, R., Sando, T., Boot, W., AbdelRazig, Y., & Sobanjo, J. O. (2019). Assessment of traffic performance measures and safety based on driver age and experience: A microsimulation based analysis for an unsignalized T-intersection. *Journal of Traffic and Transportation Engineering (English Edition)*, 6(5), 455–469. <https://doi.org/10.1016/j.jtte.2018.05.004>
- van der Horst, A. R. A., de Goede, M., de Hair-Buijssen, S., & Methorst, R. (2014). Traffic conflicts on bicycle paths: A systematic observation of behaviour from video. *Accident Analysis & Prevention*, 62, 358–368. <https://doi.org/10.1016/j.aap.2013.04.005>
- Varhelyi, A. (2016). Road Safety Management – The Need for a Systematic Approach. *The Open Transportation Journal*, 10(1), 137–155. <https://doi.org/10.2174/1874447801610010137>
- Vasconcelos, L., Neto, L., Seco, Á. M., & Silva, A. B. (2014). Validation of the Surrogate Safety Assessment Model for Assessment of Intersection Safety. *Transportation Research Record*, 2432(1), 1–9. <https://doi.org/10.3141/2432-01>
- Vishnuvardhan, K., Muthukeerthana, S., Muthuiswarya, S., Ganesh, V. N., & Rajkumar, R. (2023). Analysis of road traffic fatalities and injuries using artificial neural network: A case study on NH-544. *AIP Conference Proceedings*, 2690(1), 020013. <https://doi.org/10.1063/5.0119699>

- Vogel, K. (2003). A comparison of headway and time to collision as safety indicators. *Accident Analysis & Prevention*, 35(3), 427–433. [https://doi.org/10.1016/S0001-4575\(02\)00022-2](https://doi.org/10.1016/S0001-4575(02)00022-2)
- Wang, C., & Stamatiadis, N. (2014). Evaluation of a simulation-based surrogate safety metric. *Accident Analysis & Prevention*, 71, 82–92. <https://doi.org/10.1016/j.aap.2014.05.004>
- Wang, J. (1994). *Application of an Improved Accident Analysis Method for Highway Safety Evaluations* (FHWA-RD-94-082). <https://rosap.nhtl.bts.gov/view/dot/38350>
- Ward, J. R., Agamennoni, G., Worrall, S., Bender, A., & Nebot, E. (2015). Extending Time to Collision for probabilistic reasoning in general traffic scenarios. *Transportation Research Part C: Emerging Technologies*, 51, 66–82. <https://doi.org/10.1016/j.trc.2014.11.002>
- Winston, C., Maheshri, V., & Mannering, F. (2006). An exploration of the offset hypothesis using disaggregate data: The case of airbags and antilock brakes. *Journal of Risk and Uncertainty*, 32(2), 83–99. <https://doi.org/10.1007/s11166-006-8288-7>
- Yang, H. (2012a). *Simulation-based evaluation of traffic safety performance using surrogate safety measures* [Rutgers University - Graduate School - New Brunswick]. <https://doi.org/10.7282/T3FX78GP>
- Yang, H. (2012b). *Simulation-based evaluation of traffic safety performance using surrogate safety measures*. <https://doi.org/10.7282/T3FX78GP>
- Yasmin, S., Eluru, N., & Haque, Md. M. (2022). Addressing endogeneity in modeling speed enforcement, crash risk and crash severity simultaneously. *Analytic Methods in Accident Research*, 36, 100242. <https://doi.org/10.1016/j.amar.2022.100242>
- Zegeer, C. V., & Deen, R. C. (1978). *Traffic Conflicts as a Diagnostic Tool in Highway Safety*.
- Zhang, L., Zhang, M., Yang, W., & Dong, D. (2015). Golden Ratio Genetic Algorithm Based Approach for Modelling and Analysis of the Capacity Expansion of Urban Road Traffic

- Network. *Computational Intelligence and Neuroscience*, 2015, 512715.  
<https://doi.org/10.1155/2015/512715>
- Zheng, L., Fan, S., Ma, S., & Jiao, H. (2024). Multi-Type Traffic Conflict Identification at Signalized Intersections Based on LiDAR Point Cloud. *Transportation Research Record*, 2678(10), 916–925. <https://doi.org/10.1177/03611981241235178>
- Zheng, L., Ismail, K., & Meng, X. (2014). *Traffic conflict techniques for road safety analysis: Open questions and some insights*.  
[https://www.researchgate.net/publication/264829445\\_Traffic\\_conflict\\_techniques\\_for\\_road\\_safety\\_analysis\\_Open\\_questions\\_and\\_some\\_insights](https://www.researchgate.net/publication/264829445_Traffic_conflict_techniques_for_road_safety_analysis_Open_questions_and_some_insights)
- Zheng, L., & Sayed, T. (2019). Comparison of Traffic Conflict Indicators for Crash Estimation using Peak Over Threshold Approach. *Transportation Research Record*, 2673(5), 493–502.  
<https://doi.org/10.1177/0361198119841556>

## Appendix A – LiDAR Data Pipeline Python Code

```
import numpy as np
import pandas as pd
import glob
import os
import datetime
import pytz
from google.colab import drive
import math
import matplotlib.pyplot as plt
import seaborn as sns
import multiprocessing

# IMPORTING DATA
drive.mount('/content/drive')

gdrive = '/content/drive/My Drive/7. Corridor/AM Peak/Raw'

csv_files = glob.glob(gdrive + "/*.csv")

# Debugging: Print the list of CSV files found
print("CSV files found in the folder:")
for csv_file in csv_files:
    print(csv_file)

# Check if any CSV files were found
if not csv_files:
    print("No CSV files found in the specified folder.")
else:
    # Read each CSV file into a list of DataFrames
    df_list = [pd.read_csv(file) for file in csv_files]

    # Concatenate all DataFrames
    clean_df = pd.concat(df_list, ignore_index=True)

    first_column = clean_df.columns[0]
    clean_df = clean_df.drop([first_column], axis=1)
    clean_df['timestamp'] = pd.to_datetime(clean_df['timestamp'], errors
= 'coerce')

# CONVERTER CODE
# Update these values based on LiDAR sensor rotation
SENSOR_POSITION_LAT = 46.35075
SENSOR_POSITION_LON = -72.548111
SENSOR_ROTATION = -132
```

```

def convertPixelToGeo(x, y):
    lat = SENSOR_POSITION_LAT
    lon = SENSOR_POSITION_LON
    cx = -x
    cy = -y

    dn = cx*math.cos(SENSOR_ROTATION/180*math.pi)-cy *
math.sin(SENSOR_ROTATION/180*math.pi)
    de = cy*math.cos(SENSOR_ROTATION/180*math.pi)+cx *
math.sin(SENSOR_ROTATION/180*math.pi)

    # Earth's radius, sphere
    R = 6378137

    # Coordinate offsets in radians
    dLat = dn/R
    dLon = de/(R*math.cos(math.pi*lat/180))

    # OffsetPosition, decimal degrees
    lat = lat + dLat * 180/math.pi
    lon = lon + dLon * 180/math.pi
    return lat, lon

cX = clean_df['centerX'].apply(lambda x: float(x))
cY = clean_df['centerY'].apply(lambda y: float(y))

result = convertPixelToGeo(cX, cY)

# Ignore Warning
clean_df['lat'] = result[0]
clean_df['lng'] = result[1]

# COLUMN PROCESSING
cars = clean_df.drop(['centerX', 'centerY', 'width', 'length',
'rotation'], axis=1)
cars.columns = ['timestamp', 'objectId', 'classType', 'lat', 'lng']
# Fill NAs
cars.loc[cars['objectId'] == 'N/A', 'objectId'] = 0
cars.loc[cars['classType'] == 'N/A', 'classType'] = 0

# Filter to obtain only passenger vehicles
cars = cars[cars['classType'] == 2]

```

```

cars = cars.groupby(['objectId',
'timestamp']).first().drop(['classType'], axis=1).astype('float')
# DATA NORMALIZATION AND CLEANING
# In preparation for clustering by lane and then driving headway
position = cars[['lat', 'lng']]

object_displacement = {}

# Cluster by start position and unit vectors
for index, row in position.groupby(level = 0):
    latlng_displacement = {}
    middle_index = math.floor((len(row) - 1)/2)
    latlng_displacement['d_lat_2'] = (row.iloc[-1]['lat'] -
row.iloc[middle_index]['lat']) / math.sqrt(row.iloc[-1]['lat']**2 +
row.iloc[middle_index]['lat']**2)
    latlng_displacement['d_lng_2'] = (row.iloc[-1]['lng'] -
row.iloc[middle_index]['lng']) / math.sqrt(row.iloc[-1]['lng']**2 +
row.iloc[middle_index]['lng']**2)
    latlng_displacement['d_lat_1'] = (row.iloc[middle_index]['lat']
- row.iloc[0]['lat']) / math.sqrt(row.iloc[middle_index]['lat']**2 +
row.iloc[0]['lat']**2)
    latlng_displacement['d_lng_1'] = (row.iloc[middle_index]['lng']
- row.iloc[0]['lng']) / math.sqrt(row.iloc[middle_index]['lng']**2 +
row.iloc[0]['lng']**2)
    latlng_displacement['s_lat'] = row.iloc[0]['lat']
    latlng_displacement['s_lng'] = row.iloc[0]['lng']
    latlng_displacement['e_lat'] = row.iloc[-1]['lat']
    latlng_displacement['e_lng'] = row.iloc[-1]['lng']
    latlng_displacement['distance'] = (row.iloc[-1]['lat'] -
row.iloc[0]['lat'])**2 + (row.iloc[-1]['lng'] - row.iloc[0]['lng'])**2
    object_displacement[index] = latlng_displacement

def find_cluster_params_for_obj(x):
    x['d_lat_1'] = object_displacement[x.name[0]]['d_lat_1']
    x['d_lng_1'] = object_displacement[x.name[0]]['d_lng_1']
    x['d_lat_2'] = object_displacement[x.name[0]]['d_lat_2']
    x['d_lng_2'] = object_displacement[x.name[0]]['d_lng_2']
    x['s_lat'] = object_displacement[x.name[0]]['s_lat']
    x['s_lng'] = object_displacement[x.name[0]]['s_lng']
    x['e_lat'] = object_displacement[x.name[0]]['e_lat']
    x['e_lng'] = object_displacement[x.name[0]]['e_lng']
    x['distance'] = object_displacement[x.name[0]]['distance']
    return x

position = position.apply(find_cluster_params_for_obj, axis=1)

```

```

# Scale the d_lat and d_lng before clustering
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Now that we are using different units, normalize them all.

position['nd_lat_1'] =
scaler.fit_transform(np.array(position['d_lat_1'].values).reshape(-1,1))
position['nd_lng_1'] =
scaler.fit_transform(np.array(position['d_lng_1'].values).reshape(-1,1))
position['nd_lat_2'] =
scaler.fit_transform(np.array(position['d_lat_2'].values).reshape(-1,1))
position['nd_lng_2'] =
scaler.fit_transform(np.array(position['d_lng_2'].values).reshape(-1,1))
position['ns_lat'] =
scaler.fit_transform(np.array(position['s_lat'].values).reshape(-1,1))
position['ns_lng'] =
scaler.fit_transform(np.array(position['s_lng'].values).reshape(-1,1))
position['distance'] =
scaler.fit_transform(np.array(position['distance'].values).reshape(-1,1))

position.to_csv(path_or_buf='/content/drive/My Drive/1.
Laviolette/position-result.csv')

data_points = position.groupby('objectId').count()['lat']
dps = pd.DataFrame(data_points)
dps.columns = ['count']

# Clean / filter out any trajectories with less than four recorded
data points
filtered_dps = dps[dps['count'] > 4]

filtered_pos =
position[position.index.get_level_values(0).isin(list(filtered_dps.index))
]

# CLUSTERING CARS INTO LANES
from sklearn.cluster import KMeans

# 4 directions (N,E,S,W) * 3 actions (straight, left, right) = 12
clusters

```

```

# Trial and error informs us that [X] is better, based on how many
clusters have a single vehicle trajectory in "validation"
NUM_CLUSTERS = 25

kmeans = KMeans(n_clusters=NUM_CLUSTERS, random_state=101)
kmeans.fit(filtered_pos[['nd_lat_1', 'nd_lng_1', 'nd_lat_2',
'nd_lng_2', 'ns_lat', 'ns_lng', 'distance']])

filtered_pos['cluster'] = kmeans.labels_

plt.figure(figsize=(12,10))
scatterplot = sns.scatterplot(x='lng', y='lat', data=filtered_pos,
hue='cluster', legend='full', palette='tab20')
fig = scatterplot.get_figure()
fig.savefig("/content/drive/My Drive/7.
Corridor/Corridor_clusters_AM.png")
plt.show()

# Plotting all points by cluster
num_rows = 5
num_cols = 5

fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols,
figsize=(12,20))
fig.tight_layout()

for i in range(num_rows):
    for j in range(num_cols):
        cluster_index = i*num_cols+j
        cluster_pos = filtered_pos[filtered_pos['cluster'] ==
cluster_index]

        axes[i,j].scatter(cluster_pos['lng'], cluster_pos['lat'])
        axes[i,j].set_title('Cluster {}'.format(cluster_index))
        axes[i,j].set_xlim([filtered_pos['lng'].min() - 0.0001,
filtered_pos['lng'].max() + 0.0001])
        axes[i,j].set_ylim([filtered_pos['lat'].min() - 0.0001,
filtered_pos['lat'].max() + 0.0001])

        num_objects = len(cluster_pos.groupby('objectId').count())
        # print('For cluster {} - {}'.format(cluster_index,
num_objects))

plt.show()

```

```

# VALIDATION, VISUALIZATION, AND CHECKPOINTING
# List of cars in chronological order of first timestamp
sorted_cars = filtered_pos.sort_index(axis=0, level=1)

# Checkpoint
sorted_cars.to_csv('checkpoint-processed_data.csv')
demo = sorted_cars[['s_lng', 's_lat', 'e_lng', 'e_lat', 'cluster']]

# Plotting all points by cluster
num_rows = 5
num_cols = 5

fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols,
figsize=(12,20))
fig.tight_layout()

for i in range(num_rows):
    for j in range(num_cols):
        cluster_index = i*num_cols+j
        cluster_pos = demo[demo['cluster'] == cluster_index]

        axes[i,j].scatter(cluster_pos['s_lng'],
cluster_pos['s_lat'], c='r')
        axes[i,j].scatter(cluster_pos['e_lng'],
cluster_pos['e_lat'], c='b')
        axes[i,j].set_title('Cluster {}'.format(cluster_index))
        axes[i,j].set_xlim([demo['s_lng'].min() - 0.001,
demo['s_lng'].max() + 0.001])
        axes[i,j].set_ylim([demo['s_lat'].min() - 0.001,
demo['s_lat'].max() + 0.001])

        # num_objects = len(cluster_pos.groupby('objectId').count())
        # print('For cluster {} - {}'.format(cluster_index,
num_objects))

plt.show()

# TMC
# Manually identify the clusters with similar movements and add them
together
WBTH = len(demo[demo['cluster'] == 0])
+ len(demo[demo['cluster'] == 1])
+ len(demo[demo['cluster'] == 4])
+ len(demo[demo['cluster'] == 5])
+ len(demo[demo['cluster'] == 10])

```

```

+ len(demo[demo['cluster'] == 12])
+ len(demo[demo['cluster'] == 15])
+ len(demo[demo['cluster'] == 17])
+ len(demo[demo['cluster'] == 23])
+ len(demo[demo['cluster'] == 24])

WBRT = len(demo[demo['cluster'] == 2])

EBTH = len(demo[demo['cluster'] == 11])
+ len(demo[demo['cluster'] == 16])
+ len(demo[demo['cluster'] == 19])
+ len(demo[demo['cluster'] == 20])

EBRT = len(demo[demo['cluster'] == 8])

EBLT = len(demo[demo['cluster'] == 21])

NBTH = len(demo[demo['cluster'] == 9])
+ len(demo[demo['cluster'] == 13])

NBRT = len(demo[demo['cluster'] == 22])

NBLT = len(demo[demo['cluster'] == 6])

SBTH = len(demo[demo['cluster'] == 3])

SBRT = len(demo[demo['cluster'] == 7])

SBLT = len(demo[demo['cluster'] == 14])
+ len(demo[demo['cluster'] == 18])

print(f'WBTH = {WBTH}')
print(f'WBRT = {WBRT}')
print(f'EBTH = {EBTH}')
print(f'EBRT = {EBRT}')
print(f'EBLT = {EBLT}')
print(f'NBTH = {NBTH}')
print(f'NBRT = {NBRT}')
print(f'NBLT = {NBLT}')
print(f'SBTH = {SBTH}')
print(f'SBRT = {SBRT}')
print(f'SBLT = {SBLT}')

# DERIVING HEADWAY (IN SECONDS) PER LANE
# functions for obtaining distance.

```

```

def lat_to_m(lat):
    return lat * 111111

def lng_to_m(lng):
    return lng * 111111 * math.cos(math.radians(49.9))

# Euclidian Distance
def get_dist(length, width):
    return math.sqrt(length**2 + width**2)

# Finding nearest time between cars with nearest point approx.
def approximate_headway(df, latlng, timestamp):
    min_dist = -1
    best_approx_headway = -1
    for index, row in df.iterrows():
        dt_index = pd.to_datetime(index[1])
        dt_timestamp = pd.to_datetime(timestamp[1])

        curr_latlng = (row.lat, row.lng)

        length = lat_to_m(curr_latlng[0] - latlng[0])
        width = lng_to_m(curr_latlng[1] - latlng[1])

        dist_cars = get_dist(length, width)

        #         print('dist_cars = {}, curr_latlng = {}, latlng =
        {}.format(dist_cars, curr_latlng, latlng))

        if min_dist == -1:
            min_dist = dist_cars
            best_approx_headway = abs(dt_timestamp - dt_index)
        elif dist_cars < min_dist:
            min_dist = dist_cars
            best_approx_headway = abs(dt_timestamp - dt_index)

    best_approx_headway_s = (best_approx_headway.microseconds +
best_approx_headway.seconds * 1000000)/1000000
    return best_approx_headway_s

# PARALLEL-PROCESSING OF BULK DATA
manager = multiprocessing.Manager()
shared_dict = manager.dict()

def process_cluster(arg_data):
    curr_cluster = arg_data[0]

```

```

    i = arg_data[1]
    all_cluster_results = arg_data[2]

    prev_car = pd.DataFrame()
    cluster_result = []

    print(f'Processing Cluster {i}')

    for objectId, objectDf in curr_cluster.groupby(level = 0):
        if not prev_car.empty:
            lead_car = objectDf
            for index, row in lead_car.iterrows():
                result = approximate_headway(prev_car, (row.lat,
row.lng), index)

                try:
                    all_cluster_results[index] = result
                except Exception:
                    print(f"process failed for: {index}")
            prev_car = objectDf

    print(f'Finished Cluster {i}')

# Analyze clusters
args = []
for i in range(NUM_CLUSTERS):
    curr_cluster = sorted_cars[sorted_cars['cluster'] == i]
    args.append([curr_cluster, i, shared_dict])

pool = multiprocessing.Pool(processes=12)
pool.map(process_cluster, args)
pool.close()
pool.join()

print("Program Completed")

# Create the result dataframe
multiIndex = pd.MultiIndex.from_tuples(shared_dict.keys(),
names=('objectId', 'timestamp'))

headway_df = pd.DataFrame(data=shared_dict.values(),
                           index=multiIndex,
                           columns=['headway'])

headway_df

```

```

    headway_df.to_csv(path_or_buf='/content/drive/My Drive/7.
Corridor/Corridor_headway_AM_peak.csv')

# MERGE RESULTS
result_df = headway_df.join(sorted_cars)
result_df

# PLOT TIME HEADWAY
plt.figure(figsize=(12,12))
# plt.hist(result_df['headway'])
plt.hist(result_df[result_df['headway'] < 60]['headway'])
# plt.hist(result_df[result_df['headway'] < 4]['headway'])
plt.title('Histogram of Time Headway at AM peak hour')
plt.savefig('/content/drive/My Drive/7. Corridor/Corridor_time-
headway_AM_peak.png')
plt.show()

# SAVE RESULTS
result_df.to_csv(path_or_buf='/content/drive/My Drive/7.
Corridor/checkpoint-result.csv')

```

## Appendix B – LiDAR Checkpoint Data Distribution Python Code

```
!pip install pandas
!pip install seaborn matplotlib
import numpy as np
import pandas as pd
import os
import datetime
import pytz
import math
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import drive

drive.mount('/content/drive')

Update the path so that it matches your Google Drive path
gdrive = '/content/drive/My Drive/7. Corridor/PM Peak/Checkpoint'
path = '/checkpoint_PM Peak_combined result.csv'

df = pd.read_csv(gdrive + path)

# Convert latitude and longitude to meters
def lat_to_m(lat):
    return lat * 111111

def lng_to_m(lng, lat):
    return lng * 111111 * math.cos(math.radians(lat))

# Calculate distance between two points
def calculate_distance(row):
    start_lat, start_lng = row['s_lat'], row['s_lng']
    end_lat, end_lng = row['e_lat'], row['e_lng']

    start_lat_m, start_lng_m = lat_to_m(start_lat),
lng_to_m(start_lng, start_lat)
    end_lat_m, end_lng_m = lat_to_m(end_lat), lng_to_m(end_lng,
end_lat)

    delta_lat = end_lat_m - start_lat_m
    delta_lng = end_lng_m - start_lng_m

    distance_m = math.sqrt(delta_lat**2 + delta_lng**2)
    return distance_m
```

```

    # Calculate speed in km/h using 'travel_time (s)' instead of
timestamp conversion
    def calculate_speed(row):
        distance_m = row['distance']
        travel_time_seconds = row['travel_time (s)']

        if pd.isna(travel_time_seconds) or travel_time_seconds == 0:
            return 0 # To avoid division by zero or handle NaT

        speed_kmh = (distance_m / 1000) / (travel_time_seconds / 3600)
        return speed_kmh

    # Calculate travel time for each objectId
    travel_time = df.groupby('objectId')['time (s)'].agg(lambda x:
x.max() - x.min()).reset_index()
    travel_time.rename(columns={'time (s)': 'travel_time (s)'},
inplace=True)

    # Merge the travel time back to the original DataFrame
    df = pd.merge(df, travel_time, on='objectId')

    # Apply the distance and speed calculations
    df['distance'] = df.apply(calculate_distance, axis=1)
    df['speed_kmh'] = df.apply(calculate_speed, axis=1)

    result_df = df[['objectId', 'length', 'travel_time (s)', 'distance',
'speed_kmh', 'acc/dec (m/s2)', 'headway']]

    # Plot speed distribution
    plt.figure(figsize=(10, 6))
    sns.histplot(result_df['speed_kmh'], bins=20, kde=True,
color='blue')
    plt.title('Speed Distribution')
    plt.xlabel('Speed (km/h)')
    plt.ylabel('Frequency')
    plt.show()

    # Convert 'acc/dec (m/s2)' column to numeric
    result_df['acc/dec (m/s2)'] = pd.to_numeric(result_df['acc/dec
(m/s2)'], errors='coerce')

    # Filter entries with acc/dec values outside the range [-4.0, 4.0]
    result_df = result_df[(result_df['acc/dec (m/s2)'] >= -4.0) &
(result_df['acc/dec (m/s2)'] <= 4.0)]

```

```

# Filter the result_df where headway is between 6 and 160 seconds
filtered_result_df = result_df[(result_df['headway'] >= 6) &
(result_df['headway'] <= 160)]

# Create a boxplot to visualize the distribution
plt.figure(figsize=(8, 6))
sns.boxplot(x=result_df['speed_kmh'])

# Calculate the whiskers' range
Q1 = result_df['speed_kmh'].quantile(0.25)
Q3 = result_df['speed_kmh'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter entries within the whiskers
filtered_df = result_df[(result_df['speed_kmh'] >= Q1) &
(result_df['speed_kmh'] <= upper_bound)]

plt.title('Boxplot for Speed (km/h) with Whiskers')
plt.xlabel('Speed (km/h)')
plt.show()

# Display the cleaned result
filtered_df

# Filter the result_df where headway is between 6 and 160 seconds
filtered_result_df = result_df[(result_df['headway'] >= 6) &
(result_df['headway'] <= 160)]

# Calculate Q1 and Q3 for the filtered data (speed_kmh column)
Q1 = filtered_result_df['speed_kmh'].quantile(0.25)
Q3 = filtered_result_df['speed_kmh'].quantile(0.75)
IQR = Q3 - Q1

# Calculate the lower and upper bounds using 1.5 * IQR
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Print Q1, Q3, and the bounds
print(f"Q1 (25th percentile): {Q1}")
print(f"Q3 (75th percentile): {Q3}")
print(f"Lower bound for speed (based on 1.5 * IQR): {lower_bound}")
print(f"Upper bound for speed (based on 1.5 * IQR): {upper_bound}")

```

```

# Optionally display the filtered data for visual confirmation
filtered_result_df
fig, axs = plt.subplots(2, 1, figsize=(12, 10))

# Histogram for Speed (km/h)
axs[0].hist(filtered_df['speed_kmh'], bins=50, color='blue',
alpha=0.7)
axs[0].set_title('Distribution of Observed Speed (LiDAR API) - PM
Peak')
axs[0].set_xlabel('Speed (km/h)')
axs[0].set_ylabel('Frequency')

# Histogram for Acceleration/Deceleration (m/s2)
axs[1].hist(filtered_df['acc/dec (m/s2)'], bins=50, color='orange',
alpha=0.7)
axs[1].set_title('Distribution of Acceleration/Deceleration (m/s2)')
axs[1].set_xlabel('Acceleration/Deceleration (m/s2)')
axs[1].set_ylabel('Frequency')

plt.tight_layout()
plt.show()

# Calculate the average speed_kmh
average_speed_kmh = filtered_df['speed_kmh'].mean()
print("\nAverage Speed (km/h):", average_speed_kmh)

# Group by objectId and sum the travel time for each objectId
total_travel_time = filtered_df.groupby('objectId')['travel_time
(s)'].sum()

# Plot histogram for total travel time
plt.hist(total_travel_time, bins=20, color='skyblue',
edgecolor='black')

# Add labels and title
plt.xlabel('Total Travel Time (s)')
plt.ylabel('Frequency')
plt.title('Histogram of Total Travel Time for Each objectId')

# Show plot
plt.show()

# Group by objectId and sum the distance for each objectId
total_distance = filtered_df.groupby('objectId')['distance'].sum()

```

```

    # Plot histogram for total distance
    plt.hist(total_distance, bins=20, color='skyblue',
edgecolor='black')

    # Add labels and title
    plt.xlabel('Total Distance')
    plt.ylabel('Frequency')
    plt.title('Histogram of Total Distance for Each objectId')

    # Show plot
    plt.show()

    # Group by objectId and sum the travel_time for each objectId
    total_travel_time = filtered_df.groupby('objectId')['travel_time
(s)'].sum()

    # Group by objectId and sum the distance for each objectId
    total_distance = filtered_df.groupby('objectId')['distance'].sum()

    # Create a DataFrame with objectId, total_travel_time, and
total_distance
    total_travel_df = pd.DataFrame({'objectId': total_travel_time.index,
                                'total_travel_time':
total_travel_time.values})

    # Merge total_travel_df with total_distance based on objectId
    total_travel_distance_df =
total_travel_df.merge(pd.DataFrame({'objectId': total_distance.index,
'total_distance': total_distance.values}), on='objectId', how='inner')

    # Display the DataFrame
    total_travel_distance_df

def haversine(lat1, lon1, lat2, lon2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # Convert decimal degrees to radians
    lat1, lon1, lat2, lon2 = map(math.radians, [lat1, lon1, lat2,
lon2])

    # Haversine formula
    dlon = lon2 - lon1

```

```

        dlat = lat2 - lat1
        a = math.sin(dlat/2)**2 + math.cos(lat1) * math.cos(lat2) *
math.sin(dlon/2)**2
        c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
        r = 6371 # Radius of earth in kilometers
        return c * r * 1000

# Coordinates
lat1, lon1 = 46.356364, -72.539311
lat2, lon2 = 46.351149, -72.547623

# Calculate distance
distance = haversine(lat1, lon1, lat2, lon2)
print("Distance between the coordinates:", distance, "meters")

# Filter the DataFrame for total_distance
cleaned_total_travel_distance_df =
total_travel_distance_df[(total_travel_distance_df['total_distance'] >=
700) & (total_travel_distance_df['total_distance'] <= 1000)]

cleaned_total_travel_distance_df

# Define the bin edges
bin_edges = np.arange(60, 222, 12)

counts, bins =
np.histogram(cleaned_total_travel_distance_df['total_travel_time'],
bins=bin_edges)

# Plot histogram with multiplied frequencies
plt.figure(figsize=(10, 6))
plt.bar(bins[:-1], counts, width=(bins[1] - bins[0]), color='black',
alpha=1.0)
plt.xlabel('Travel Time (s)')
plt.ylabel('Counts')
plt.title('Distribution of Observed Travel Time Counts (LiDAR API) -
PM Peak')

# Set x-axis ticks
plt.xticks(np.arange(60, 222, 20))

# Set y-axis ticks
plt.yticks(np.arange(0, 101, 10))

# Hide the grid

```

```

plt.grid(False)

# Hide the top and right spines (box around the figure)
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.show()

# Calculate the total distance traveled for each objectId
total_distances = filtered_df.groupby('objectId')['distance'].sum()

# Filter objectIds with total distance traveled between 0 and 1000m
filtered_distances = total_distances[(total_distances >= 0) &
(total_distances <= 1000)]

# Filter the original DataFrame to include only these objectIds
filtered_df =
filtered_df[filtered_df['objectId'].isin(filtered_distances.index)]

# Find the maximum speed for each objectId
max_speeds = filtered_df.groupby('objectId')['speed_kmh'].max()

# Filter speeds between 0 km/h and 70 km/h
filtered_speeds = max_speeds[(max_speeds > 0) & (max_speeds <= 70)]

# Create a histogram and cumulative distribution plot
fig, ax1 = plt.subplots(figsize=(12, 8))

# Define bins
bins = [0, 10, 20, 30, 40, 50, 60, 70]

# Plot histogram
counts, bins, patches = ax1.hist(filtered_speeds, bins=bins,
color='black', alpha=1.0, edgecolor='black')

# Set x-axis and y-axis labels
ax1.set_xlabel('Speed (km/h)')
ax1.set_ylabel('Observations')

# Add title
ax1.set_title('Vehicle Speed Distribution - PM Peak')

# Adjust the y-axis limits for better clarity
ax1.set_ylim(0, max(counts) * 1.1)

```

```

# Create a second y-axis for the cumulative distribution
ax2 = ax1.twinx()

# Calculate cumulative counts as percentages
cumulative_counts = np.cumsum(counts)
cumulative_percentages = 100 * cumulative_counts /
cumulative_counts[-1]

# Plot cumulative distribution on the second y-axis
# Calculate bin centers for plotting
bin_centers = 0.5 * (bins[1:] + bins[:-1])
ax2.plot(bin_centers, cumulative_percentages, color='orange',
marker='o', linestyle='-', linewidth=2)

# Set second y-axis label
ax2.set_ylabel('Cumulative Observations ( percent)')

# Set y-axis limits
ax2.set_ylim(0, 105)

# Adjust x-axis ticks to be centered on each bin
ax1.set_xticks(bin_centers)

# Custom x-axis tick labels
ax1.set_xticklabels(['0-10', '10-20', '20-30', '30-40', '40-50',
'50-60', '60-70'])

# Show plot spines to make axis lines visible
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)
ax2.spines['top'].set_visible(False)
ax2.spines['right'].set_visible(True)
ax1.spines['left'].set_visible(True)
ax1.spines['bottom'].set_visible(True)

# Re-enable ticks on the axes, while keeping the spines visible
ax1.tick_params(axis='x', bottom=True)
ax1.tick_params(axis='y', left=True)
ax2.tick_params(axis='y', right=True)

# Set the figure background to be transparent or white (depending on
preference)
fig.patch.set_alpha(0) # Set transparency to 0 for no background
ax1.patch.set_alpha(0)

```

```

# Adjust the margins to make sure the entire plot fits
plt.subplots_adjust(left=0.15, right=0.85, top=0.85, bottom=0.15)

# Adjust the layout to avoid cutting off the cumulative line
plt.tight_layout(pad=1.5)

# Show plot
plt.show()

# Filter rows where 'speed_kmh' is between 0 and 70
speed_filtered_df = filtered_df[(filtered_df['speed_kmh'] >= 0) &
(filtered_df['speed_kmh'] <= 70)]

# Extract 'acc/dec (m/s²)' data
acc_dec_data = speed_filtered_df['acc/dec (m/s2)']

# Define bin edges from -4 to 4 with a bin size of 1
bin_edges = np.arange(-4, 5, 1)

# Create a histogram and cumulative distribution plot
fig, ax1 = plt.subplots(figsize=(12, 8))

# Plot histogram with default count settings
counts, bins, patches = ax1.hist(acc_dec_data, bins=bin_edges,
edgecolor='black', color='black')

# Add labels and title
ax1.set_xlabel('Acceleration/Deceleration (m/s²)')
ax1.set_ylabel('Observations')
ax1.set_title('Acceleration/Deceleration Distribution - PM Peak')

# Adjust the y-axis limits for better clarity
ax1.set_ylim(0, max(counts) * 1.1)

# Create a second y-axis for the cumulative distribution
ax2 = ax1.twinx()

# Calculate cumulative counts as percentages
cumulative_counts = np.cumsum(counts)
cumulative_percentages = 100 * cumulative_counts /
cumulative_counts[-1]

```

```

# Plot cumulative distribution on the second y-axis
bin_centers = 0.5 * (bins[1:] + bins[:-1])
ax2.plot(bin_centers, cumulative_percentages, color='orange',
marker='o', linestyle='-', linewidth=2)

# Set second y-axis label
ax2.set_ylabel('Cumulative Observations ( percent)')

# Set y-axis limits for cumulative distribution
ax2.set_ylim(0, 105)

# Set x-axis ticks to match bin centers
ax1.set_xticks(bin_centers)
# Custom x-axis tick labels for the specified ranges
ax1.set_xticklabels(['-4< $\alpha$  $\leq$ -3', '-3< $\alpha$  $\leq$ -2', '-2< $\alpha$  $\leq$ -1', '-1< $\alpha$  $\leq$ 0',
'0< $\alpha$  $\leq$ 1', '1< $\alpha$  $\leq$ 2', '2< $\alpha$  $\leq$ 3', '3< $\alpha$  $\leq$ 4'])

# Set y-axis ticks for the primary y-axis (Observations)
ax1.yaxis.set_major_locator(plt.MaxNLocator(integer=True))

# Set y-axis ticks for the secondary y-axis (Cumulative
Observations) to range from 0 to 105 with intervals of 10
ax2.set_yticks(np.arange(0, 105, 10))

# Remove grid lines
ax1.grid(False)

# Show plot spines to make axis lines visible
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)
ax2.spines['top'].set_visible(False)
ax2.spines['right'].set_visible(True)
ax1.spines['left'].set_visible(True)
ax1.spines['bottom'].set_visible(True)

# Re-enable ticks on the axes, while keeping the spines visible
ax1.tick_params(axis='x', bottom=True)
ax1.tick_params(axis='y', left=True)
ax2.tick_params(axis='y', right=True)

# Set the figure background to be transparent or white (depending on
preference)
fig.patch.set_alpha(0)
ax1.patch.set_alpha(0)

```

```

# Adjust the margins to make sure the entire plot fits
plt.subplots_adjust(left=0.15, right=0.85, top=0.85, bottom=0.15)

# Adjust the layout to avoid cutting off the cumulative line
plt.tight_layout(pad=1.5)

# Show plot
plt.show()

# Calculate the total distance traveled for each objectId
total_distances = filtered_df.groupby('objectId')['distance'].sum()

# Filter objectIds with total distance traveled between 700m and
1000m
filtered_distances = total_distances[(total_distances >= 700) &
(total_distances <= 1000)]

# Filter the original DataFrame to include only these objectIds
filtered_df =
filtered_df[filtered_df['objectId'].isin(filtered_distances.index)]

# Find the maximum speed for each objectId
max_speeds = filtered_df.groupby('objectId')['speed_kmh'].max()

# Filter speeds above 20 km/h and up to 70 km/h
filtered_speeds = max_speeds[(max_speeds > 20) & (max_speeds <= 70)]

# Plot histogram
plt.figure(figsize=(12, 8))
plt.hist(filtered_speeds, bins=20, color='black', alpha=1.0)

# Add labels and title with increased font size
plt.xlabel('Speed (km/h)', fontsize=14)
plt.ylabel('Counts', fontsize=14)
plt.title('Distribution of Observed Speed Counts (LiDAR API) - PM
Peak', fontsize=16)

# Set x-axis ticks every 5 km/h up to 71 km/h
plt.xticks(np.arange(20, 71, 5), fontsize=12)

# Set y-axis ticks to range from 0 to 70 with intervals of 10
plt.yticks(np.arange(0, 71, 10), fontsize=12)

# Remove grid lines
plt.grid(False)

```

```
# Hide the top and right spines (box around the figure)
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Show plot
plt.show()
```

## Appendix C – Minimum Desired Speed Python Code

```
import pandas as pd
import os
import re
import numpy as np
    percentmatplotlib inline
import matplotlib.pyplot as plt
from google.colab import drive
from datetime import datetime
from sklearn.metrics import mean_absolute_error
from scipy.interpolate import make_interp_spline

# Mount Google Drive
drive.mount('/content/drive', force_remount=True)

# AM PEAK OBSERVED SPEED
# File paths for observed LiDAR speed data
lidarspeed_files = {
    'nb_am': '/content/drive/MyDrive/Test/St-Maurice Corridor_AM
Peak_LIDAR Speeds NB.csv',
    'sb_am': '/content/drive/MyDrive/Test/St-Maurice Corridor_AM
Peak_LIDAR Speeds SB.csv'
}

# Function to read and calculate average observed speed
def calculate_average_observed_speed(file_paths):
    observed_speeds = {}
    for key, file_path in file_paths.items():
        df = pd.read_csv(file_path)
        observed_speeds[key] = df['LiDAR Speed'].mean()
    return observed_speeds

# Calculate average observed speed
observed_speed = calculate_average_observed_speed(lidarspeed_files)

print("Observed Speed:", observed_speed)

# AM PEAK RELABELLING SIMULATED SPEED
# Function to label directions based on DataCollectionMeasurement
def label_direction(data_collection_measurement):
    # Handle cases where the value is not a valid string
    if pd.isna(data_collection_measurement) or not
isinstance(data_collection_measurement, str):
        return None # Return None for invalid or missing values
```

```

# Split the DataCollectionMeasurement at the ":"
split_value = data_collection_measurement.split(":")[0]

# Check the numeric part before the ":" and label accordingly
if split_value in ['25', '26', '27', '28', '29']:
    return '66: St Maurice NB (Lav - Chenaux)'
elif split_value in ['30', '31', '32', '33', '34']:
    return '67: St Maurice SB (Chenaux - Lav)'
else:
    return None # Return None if the value doesn't match the
specified ranges

# Function to rename directions based on DataCollectionMeasurement
def rename_directions(directory):
    for file_name in os.listdir(directory):
        if file_name.endswith('.csv'):
            file_path = os.path.join(directory, file_name)
            try:
                df = pd.read_csv(file_path,
dtype={'DataCollectionMeasurement': str, 'SimRun': str}, low_memory=False)
            except ValueError as e:
                print(f"Error reading {file_name}: {e}")
                continue

            print(f"Reading {file_name}, shape: {df.shape}")

            # Apply the direction labeling function
            df['Direction'] =
df['DataCollectionMeasurement'].apply(label_direction)

            # Filter out rows with invalid or unmatched directions
            df = df[df['Direction'].notnull()]

            print(f"After filtering DataCollectionMeasurement,
shape: {df.shape}")

            # Replace 'DataCollectionMeasurement' with 'Direction'
            df['DataCollectionMeasurement'] = df['Direction']
            df.drop(columns=['Direction'], inplace=True)

            print(f"Modified DataFrame for {file_name}:")
            print(df[['DataCollectionMeasurement']].drop_duplicates(
))

```

```

        # Save the modified file
        output_file_path = os.path.join(directory, 'Modified_' +
file_name)

        df.to_csv(output_file_path, index=False)

        print(f"Modified file saved to {output_file_path}")

    # Directory containing speed files
    speed_dir = '/content/drive/MyDrive/Test/AM/Speed'

    # Rename directions
    rename_directions(speed_dir)

    # Function to calculate average simulated speed for each direction
    def calculate_average_speed(directory):
        results = []
        for file_name in os.listdir(directory):
            if file_name.startswith('Modified_') and
file_name.endswith('.csv'):
                file_path = os.path.join(directory, file_name)
                try:
                    df = pd.read_csv(file_path)
                except ValueError as e:
                    print(f"Error reading {file_name}: {e}")
                    continue

                if 'SpeedAvgArith(All)' not in df.columns:
                    print(f"Column 'SpeedAvgArith(All)' not found in
{file_name}")

                    continue

                nb_avg_speed = df[df['DataCollectionMeasurement'] ==
'66: St Maurice NB (Lav - Chenaux)']['SpeedAvgArith(All)'].mean()
                sb_avg_speed = df[df['DataCollectionMeasurement'] ==
'67: St Maurice SB (Chenaux - Lav)']['SpeedAvgArith(All)'].mean()

                # Extract file number using regular expression
                file_number_match =
re.search(r'\((\d+)\s*\s*Des\s*Speed\)', file_name)
                if file_number_match:
                    file_number = file_number_match.group(1)
                else:
                    file_number = "Unknown"

                results.append({

```

```

        'FileName': file_name,
        'FileNumber': file_number,
        'NB_Avg_Speed': nb_avg_speed,
        'SB_Avg_Speed': sb_avg_speed
    })

    print(f"Processed {file_name}: FileNumber =
{file_number}, NB Avg Speed = {nb_avg_speed}, SB Avg Speed =
{sb_avg_speed}")

    return results

# Calculate average simulated speed
average_speeds = calculate_average_speed(speed_dir)

print("Average Simulated Speeds:")
for result in average_speeds:
    print(result)

# AM PEAK SPEED MANE
# Function to calculate MANE between simulated and observed speeds
def calculate_mane(average_speeds, observed_speed):
    speed_mane_results = []
    for result in average_speeds:
        nb_simulated_speed = result['NB_Avg_Speed']
        sb_simulated_speed = result['SB_Avg_Speed']

        if nb_simulated_speed is not None:
            speed_mane_nb =
mean_absolute_error([observed_speed['nb_am']], [nb_simulated_speed]) /
observed_speed['nb_am']
            speed_mane_results.append({
                'FileName': result['FileName'],
                'FileNumber': result['FileNumber'],
                'Direction': 'NB',
                'MANE': speed_mane_nb
            })

        if sb_simulated_speed is not None:
            speed_mane_sb =
mean_absolute_error([observed_speed['sb_am']], [sb_simulated_speed]) /
observed_speed['sb_am']
            speed_mane_results.append({
                'FileName': result['FileName'],
                'FileNumber': result['FileNumber'],

```

```

        'Direction': 'SB',
        'MANE': speed_mane_sb
    })

# Plot MANE values with smooth curves
plt.figure(figsize=(12, 6))

for direction in ['NB', 'SB']:
    direction_data = speed_mane_df[speed_mane_df['Direction'] ==
direction]

    # Convert 'FileNumber' to numeric
    direction_data['FileNumber'] =
pd.to_numeric(direction_data['FileNumber'], errors='coerce')

    # Remove NaN values after conversion
    direction_data = direction_data.dropna(subset=['FileNumber',
'MANE'])

    # Sort by 'FileNumber' for correct plotting order
    direction_data = direction_data.sort_values(by='FileNumber')

    x_smooth = direction_data['FileNumber']
    y_smooth = direction_data['MANE']

    # Use spline interpolation for smoother curves
    spl = make_interp_spline(x_smooth, y_smooth, k=3)
    x_smooth = np.linspace(x_smooth.min(), x_smooth.max(), 300)
    y_smooth = spl(x_smooth)

    plt.plot(x_smooth, y_smooth, label=f'MANE {direction}')

plt.xlabel('Minimum Design Speed (km/h)')
plt.ylabel('MANE')
plt.title('Speed MANE')
plt.legend()
plt.xticks(np.arange(x_smooth.min(), x_smooth.max() + 1, 1),
rotation=90)
plt.grid(True)
plt.tight_layout()
plt.show()

return speed_mane_results

```

```

# Calculate MANE
speed_mane = calculate_mane(average_speeds, observed_speed)

# Convert MANE results to DataFrame for easy plotting
speed_mane_df = pd.DataFrame(speed_mane)

# AM PEAK OBSERVED TRAVEL TIME
# File paths for observed LiDAR travel time data
lidartime_files = {
    'nb_am': '/content/drive/MyDrive/Test/St-Maurice Corridor_AM
Peak_LIDAR Travel Time NB.csv',
    'sb_am': '/content/drive/MyDrive/Test/St-Maurice Corridor_AM
Peak_LIDAR Travel Time SB.csv'
}

# Function to read and calculate average observed travel time
def calculate_average_observed_time(file_paths):
    observed_times = {}
    for key, file_path in file_paths.items():
        df = pd.read_csv(file_path)
        observed_times[key] = df['GAPI TravTm'].mean()
    return observed_times

# Calculate average observed travel time
observed_time = calculate_average_observed_time(lidartime_files)

print("Observed Time:", observed_time)

# AM PEAK SIMULATED TRAVEL TIME
# Directory where CSV files are located
time_dir = '/content/drive/MyDrive/Test/AM/Travel Time'

# Function to calculate average simulated travel time for each
direction
def calculate_average_travel_time(directory):
    results = []
    for file_name in os.listdir(directory):
        if file_name.endswith('.csv'):
            file_path = os.path.join(directory, file_name)
            try:
                # Read the CSV file
                df = pd.read_csv(file_path)

                # Filter rows based on exact matches in
VehicleTravelTimeMeasurement

```

```

        nb_avg_time =
df[df['VehicleTravelTimeMeasurement'].str.contains('66: St Maurice NB (Lav
- Chenaux)', na=False, regex=False)][['TravTm(All)']].dropna().mean()
        sb_avg_time =
df[df['VehicleTravelTimeMeasurement'].str.contains('67: St Maurice SB
(Chenaux - Lav)', na=False, regex=False)][['TravTm(All)']].dropna().mean()

        # Extract file number using regular expression
        file_number_match =
re.search(r'\((\d+)\s*Des\s*Speed\)', file_name)
        if file_number_match:
            file_number = file_number_match.group(1)
        else:
            file_number = "Unknown"

        results.append({
            'FileName': file_name,
            'FileNumber': file_number,
            'NB_Avg_Time': nb_avg_time,
            'SB_Avg_Time': sb_avg_time
        })

        print(f"Processed {file_name}: FileNumber =
{file_number}, NB Avg Time = {nb_avg_time}, SB Avg Time = {sb_avg_time}")

    except Exception as e:
        print(f"Error processing {file_name}: {e}")
        continue

    return results

# Calculate average simulated travel times
average_travel_times = calculate_average_travel_time(time_dir)

print("\nAverage Simulated Travel Times:")
for result in average_travel_times:
    print(result)

# AM PEAK TRAVEL TIME MANE
# Function to calculate MANE between simulated and observed travel
times
def calculate_mane_travel_time(average_times, observed_times):
    mane_results = []
    for result in average_times:
        nb_simulated_time = result['NB_Avg_Time']

```

```

        sb_simulated_time = result['SB_Avg_Time']

        if nb_simulated_time is not None:
            mane_nb = mean_absolute_error([observed_times['nb_am']],
[nb_simulated_time]) / observed_times['nb_am']
            mane_results.append({
                'FileName': result['FileName'],
                'FileNumber': result['FileNumber'],
                'Direction': 'NB',
                'MANE': mane_nb
            })

        if sb_simulated_time is not None:
            mane_sb = mean_absolute_error([observed_times['sb_am']],
[sb_simulated_time]) / observed_times['sb_am']
            mane_results.append({
                'FileName': result['FileName'],
                'FileNumber': result['FileNumber'],
                'Direction': 'SB',
                'MANE': mane_sb
            })

    return mane_results

# Calculate MANE for travel times
travel_time_mane = calculate_mane_travel_time(average_travel_times,
observed_time)

# Convert MANE results to DataFrame for easy plotting
travel_time_mane_df = pd.DataFrame(travel_time_mane)

# Ensure 'FileNumber' is numeric in the entire DataFrame first
travel_time_mane_df['FileNumber'] =
pd.to_numeric(travel_time_mane_df['FileNumber'], errors='coerce')

# Remove NaN values after conversion
travel_time_mane_df =
travel_time_mane_df.dropna(subset=['FileNumber', 'MANE'])

# Plot MANE values with smooth curves
plt.figure(figsize=(12, 6))

for direction in ['NB', 'SB']:
    direction_data =
travel_time_mane_df[travel_time_mane_df['Direction'] == direction]

```

```

# Sort by 'FileNumber' for correct plotting order
direction_data = direction_data.sort_values(by='FileNumber')

x_smooth = direction_data['FileNumber']
y_smooth = direction_data['MANE']

# Use spline interpolation for smoother curves
spl = make_interp_spline(x_smooth, y_smooth, k=2)
x_smooth_dense = np.linspace(x_smooth.min(), x_smooth.max(),
1000)
y_smooth_dense = spl(x_smooth_dense)

# Plot the MANE line and get the color
line, = plt.plot(x_smooth_dense, y_smooth_dense, label=f'MANE
{direction}')
line_color = line.get_color()

# Fit a polynomial trendline (4th degree for example)
poly_coeffs = np.polyfit(x_smooth_dense, y_smooth_dense, 4)
poly_trendline = np.polyval(poly_coeffs, x_smooth_dense)

# Plot the polynomial trendline with the same color as the MANE
line
plt.plot(x_smooth_dense, poly_trendline, linestyle='--',
color=line_color, label=f'Trendline {direction}')

# Set x-axis ticks to show every file number
plt.xlabel('Minimum Design Speed (km/h)')
plt.ylabel('MANE')
plt.title('Travel Time MANE')
plt.legend()
plt.xticks(np.arange(travel_time_mane_df['FileNumber'].min(),
travel_time_mane_df['FileNumber'].max() + 1), rotation=90)
plt.grid(True)
plt.tight_layout()
plt.show()

# COMBINED MANE
# Convert 'FileNumber' to numeric for both DataFrames
speed_mane_df['FileNumber'] =
pd.to_numeric(speed_mane_df['FileNumber'], errors='coerce')
travel_time_mane_df['FileNumber'] =
pd.to_numeric(travel_time_mane_df['FileNumber'], errors='coerce')

```

```

# Remove NaN values after conversion
speed_mane_df = speed_mane_df.dropna(subset=['FileNumber', 'MANE'])
travel_time_mane_df =
travel_time_mane_df.dropna(subset=['FileNumber', 'MANE'])

# Plot combined MANE values for both speed and travel time, and
combined MANE for each speed limit
for direction in ['NB', 'SB']:
    plt.figure(figsize=(12, 6))

    # Plot Speed MANE
    direction_speed_data = speed_mane_df[speed_mane_df['Direction']
== direction]
    direction_speed_data =
direction_speed_data.sort_values(by='FileNumber')

    x_smooth_speed = direction_speed_data['FileNumber']
    y_smooth_speed = direction_speed_data['MANE']

    spl_speed = make_interp_spline(x_smooth_speed, y_smooth_speed,
k=3)
    x_smooth_speed_dense = np.linspace(x_smooth_speed.min(),
x_smooth_speed.max(), 1000)
    y_smooth_speed_dense = spl_speed(x_smooth_speed_dense)

    line_speed, = plt.plot(x_smooth_speed_dense,
y_smooth_speed_dense, label=f'Speed MANE {direction}', color='blue')

    # Plot Travel Time MANE
    direction_travel_time_data =
travel_time_mane_df[travel_time_mane_df['Direction'] == direction]
    direction_travel_time_data =
direction_travel_time_data.sort_values(by='FileNumber')

    x_smooth_travel = direction_travel_time_data['FileNumber']
    y_smooth_travel = direction_travel_time_data['MANE']

    spl_travel = make_interp_spline(x_smooth_travel,
y_smooth_travel, k=2)
    x_smooth_travel_dense = np.linspace(x_smooth_travel.min(),
x_smooth_travel.max(), 1000)
    y_smooth_travel_dense = spl_travel(x_smooth_travel_dense)

    # Plot the MANE line

```

```

plt.plot(x_smooth_travel_dense, y_smooth_travel_dense,
color='green', label=f'Travel Time MANE {direction}')

# Combine MANE values for each speed limit
combined_mane = np.interp(x_smooth_speed_dense,
x_smooth_travel_dense, y_smooth_travel_dense) + y_smooth_speed_dense
combined_mane_min_idx = np.argmin(combined_mane) # Index of
minimum combined MANE
combined_mane_min_speed =
x_smooth_speed_dense[combined_mane_min_idx] # Speed limit where minimum
occurs
combined_mane_min_mane = np.min(combined_mane) # Minimum MANE
value

plt.plot(x_smooth_speed_dense, combined_mane, label=f'Combined
MANE {direction}', color='red')

# Annotate the lowest MANE speed limit and MANE value without a
leader line
plt.scatter(combined_mane_min_speed, combined_mane_min_mane,
color='black', s=50)
bbox_props = dict(boxstyle="round,pad=0.3", fc="white",
ec="black", lw=1)
plt.text(combined_mane_min_speed, combined_mane_min_mane + 0.1,
f'({combined_mane_min_speed:.1f} km/h,
{combined_mane_min_mane:.2f})', fontsize=10,
ha='center', va='bottom', bbox=bbox_props)

plt.xlabel('Minimum Desired Speed (km/h)')
plt.ylabel('MANE')
plt.title(f'AM Peak Combined MANE for Minimum Desired Speed -
{direction}')
plt.legend()
plt.xticks(np.arange(travel_time_mane_df['FileNumber'].min(),
travel_time_mane_df['FileNumber'].max() + 1), rotation=90)
plt.grid(True)
plt.tight_layout()
plt.show()

```

## Appendix D – Maximum Desired Speed and Desired Acceleration/Deceleration Python Code

```
import os
import glob
import pandas as pd
from google.colab import drive
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.metrics import r2_score
from google.colab import drive

drive.mount('/content/drive')

gdrive = '/content/drive/My Drive/7. Corridor/AM Peak/Checkpoint'
path = '/checkpoint_AM Peak_combined result.csv'
df = pd.read_csv(gdrive + path)

# DESIRED SPEED
# Assuming df is your DataFrame
df['speed (km/h)'] = pd.to_numeric(df['speed (km/h)'],
errors='coerce')
df['headway'] = pd.to_numeric(df['headway'], errors='coerce')

# Fill NaN values with a placeholder, here assuming 0, but this can
be adjusted
df['speed (km/h)'].fillna(0, inplace=True)
df['headway'].fillna(0, inplace=True)

# Filter based on conditions
time_headway_6s = df[(df['headway'] >= 6) & (df['speed (km/h)'] >=
35) & (df['speed (km/h)'] <= 51)]
speed0 = time_headway_6s.groupby('objectId', sort=False)['speed
(km/h)'].max()
speed_sorted = speed0.sort_values()
speed_number_of_rows = len(speed_sorted)
rank = list(range(1, speed_number_of_rows + 1))
speed = speed_sorted.tolist()
p = [r / (len(rank) + 1) for r in rank]

# Plot Cumulative Probability vs. Speed
fig, ax = plt.subplots()
ax.plot(speed, p)
```

```

    ax.set(xlabel='Speed (km/h)', ylabel='Cumulative Probability',
ylim=(0.0, 1.0))

    # Add grid
    ax.grid(True)
    # Fit linear regression
    speed2 = sm.add_constant(speed)
    est = sm.OLS(p, speed2)
    est2 = est.fit()

    # Plot the regression line
    z = np.polyfit(speed, p, 1)
    p_hat = np.poly1d(z)(speed)
    ax.plot(speed, p_hat, "r--")

    # Annotate the point where cumulative probability equals 1.0
    cumulative_prob_1 = 1.0
    speed_at_cumulative_prob_1 = np.interp(cumulative_prob_1, p, speed)
    ax.scatter([speed_at_cumulative_prob_1], [cumulative_prob_1],
color='red')
    ax.annotate(f'Speed at Cumulative Probability 1.0:
{speed_at_cumulative_prob_1:.2f} km/h',
                xy=(speed_at_cumulative_prob_1, cumulative_prob_1),
                xytext=(speed_at_cumulative_prob_1, cumulative_prob_1 -
0.2),
                arrowprops=dict(facecolor='black',
arrowstyle='wedge,tail_width=0.7', lw=1, alpha=0.5),
                bbox=dict(boxstyle='round,pad=0.3', edgecolor='black',
facecolor='white'))

    text = f"$y={z[0]:0.3f}\\;x{z[1]:+0.3f}\\nR^2 = {r2_score(p,
p_hat):0.3f}$"
    plt.gca().text(0.05, 0.95, text, transform=plt.gca().transAxes,
                    fontsize=14, verticalalignment='top')

    # Display regression summary
    print(est2.summary())

    # Show the plot
    plt.show()

    # DESIRED ACCELERATION
    df['acc/dec (m/s2)'] = pd.to_numeric(df['acc/dec (m/s2)'],
errors='coerce')

```

```

#speed between 0,10
df1=df[(df['speed (km/h)']>0) & (df['speed (km/h)']<10) &
(df['acc/dec (m/s2)']>=0.1) & (df['acc/dec (m/s2)']<=4)]
a=df1['acc/dec (m/s2)']
import matplotlib.pyplot as plt
plt.figure(0)
plt.xlabel('Acceleration (m/s^2)',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.ylim((0,20000))
plt.grid(True)
bin_edge=[0,0.5,1,1.5,2,2.5,3,3.5,4]
plt.hist(a,bins=bin_edge)
plt.title('Vehicle Acceleration')
min_acc1=df1['acc/dec (m/s2)'].quantile(0.05)
mean_acc1=df1['acc/dec (m/s2)'].quantile(0.5)
max_acc1=df1['acc/dec (m/s2)'].quantile(0.95)

#speed between 10,20
df1=df[(df['speed (km/h)']>=10) & (df['speed (km/h)']<20) &
(df['acc/dec (m/s2)']>=0.1) & (df['acc/dec (m/s2)']<=4)]
a=df1['acc/dec (m/s2)']
import matplotlib.pyplot as plt
plt.figure(1)
plt.xlabel('Acceleration (m/s^2)',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.ylim((0,20000))
plt.grid(True)
bin_edge=[0,0.5,1,1.5,2,2.5,3,3.5,4]
plt.hist(a,bins=bin_edge)
plt.title('Vehicle Acceleration')
min_acc2=df1['acc/dec (m/s2)'].quantile(0.05)
mean_acc2=df1['acc/dec (m/s2)'].quantile(0.5)
max_acc2=df1['acc/dec (m/s2)'].quantile(0.95)

#speed between 20,30
df1=df[(df['speed (km/h)']>=20) & (df['speed (km/h)']<30) &
(df['acc/dec (m/s2)']>=0.1) & (df['acc/dec (m/s2)']<=4)]
a=df1['acc/dec (m/s2)']
import matplotlib.pyplot as plt
plt.figure(2)
plt.xlabel('Acceleration (m/s^2)',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.ylim((0,20000))
plt.grid(True)
bin_edge=[0,0.5,1,1.5,2,2.5,3,3.5,4]

```

```

plt.hist(a,bins=bin_edge)
plt.title('Vehicle Acceleration')
min_acc3=df1['acc/dec (m/s2)'].quantile(0.05)
mean_acc3=df1['acc/dec (m/s2)'].quantile(0.5)
max_acc3=df1['acc/dec (m/s2)'].quantile(0.95)

#speed between 30,40
df1=df[(df['speed (km/h)']>=30) & (df['speed (km/h)']<40) &
(df['acc/dec (m/s2)']>=0.1) & (df['acc/dec (m/s2)']<=4)]
a=df1['acc/dec (m/s2)']
import matplotlib.pyplot as plt
plt.figure(3)
plt.xlabel('Acceleration (m/s^2)',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.ylim((0,20000))
plt.grid(True)
bin_edge=[0,0.5,1,1.5,2,2.5,3,3.5,4]
plt.hist(a,bins=bin_edge)
plt.title('Vehicle Acceleration')
min_acc4=df1['acc/dec (m/s2)'].quantile(0.05)
mean_acc4=df1['acc/dec (m/s2)'].quantile(0.5)
max_acc4=df1['acc/dec (m/s2)'].quantile(0.95)

#speed between 40,50
df1 = df[(df['speed (km/h)'] >= 40) & (df['speed (km/h)'] < 50) &
(df['acc/dec (m/s2)'] >= 0.1) & (df['acc/dec (m/s2)'] <= 4)]
a5 = df1['acc/dec (m/s2)']
plt.figure(4)
plt.xlabel('Acceleration (m/s^2)', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.ylim((0, 20000))
plt.grid(True)
bin_edge = [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4]
plt.hist(a5, bins=bin_edge)
plt.title('Vehicle Acceleration')
min_acc5 = df1['acc/dec (m/s2)'].quantile(0.05)
mean_acc5 = df1['acc/dec (m/s2)'].quantile(0.5)
max_acc5 = df1['acc/dec (m/s2)'].quantile(0.95)

max_acc=[max_acc1,max_acc2,max_acc3,max_acc4,max_acc5]
med_acc=[mean_acc1,mean_acc2,mean_acc3,mean_acc4,mean_acc5]
min_acc=[min_acc1,min_acc2,min_acc3,min_acc4,min_acc5]
x=[10,20,30,40,50]

```

```

fig=plt.figure()
ax1=fig.add_subplot(111)
ax1.scatter(x,max_acc,c='b',marker="s",label='Maximum')
ax1.scatter(x,med_acc,c='r',marker="s",label='Median')
ax1.scatter(x,min_acc,c='g',marker="s",label='Minimum')

# Generate more points for the best-fit lines
x_fit = np.linspace(min(x), max(x), 1000)

fit_max = np.polyfit(x, max_acc, 1)
fit_func_max = np.poly1d(fit_max)
fit_med = np.polyfit(x, med_acc, 1)
fit_func_med = np.poly1d(fit_med)
fit_min = np.polyfit(x, min_acc, 1)
fit_func_min = np.poly1d(fit_min)

# Plot the best-fit lines
ax1.plot(x, fit_func_max(x), 'b--', label='Best Fit (Maximum)')
ax1.plot(x, fit_func_med(x), 'r--', label='Best Fit (Median)')
ax1.plot(x, fit_func_min(x), 'g--', label='Best Fit (Minimum)')

equation_max = f'Best Fit Line (Max): y = {fit_max[0]:.2f}x + {fit_max[1]:.2f}'
equation_med = f'Best Fit Line (Med): y = {fit_med[0]:.2f}x + {fit_med[1]:.2f}'
equation_min = f'Best Fit Line (Min): y = {fit_min[0]:.2f}x + {fit_min[1]:.2f}'
ax1.text(2, 2.5, equation_max, fontsize=12, color='b')
ax1.text(2, 1.5, equation_med, fontsize=12, color='r')
ax1.text(2, 0.5, equation_min, fontsize=12, color='g')

plt.legend(bbox_to_anchor=(1.05,1),loc='best');
plt.xlabel('Speed (km/h)',fontsize=12)
plt.ylabel('Acceleration (m/s^2)',fontsize=12)
plt.title('Desired Acceleration')
plt.ylim((0.0,4.0))
plt.xlim((0,60))
ax1.yaxis.grid()

plt.show()

x2=sm.add_constant(x)
est=sm.OLS(max_acc,x2)
est2=est.fit()

```

```

    result=pd.DataFrame.transpose(pd.DataFrame([est2.params,est2.tvalues
,est2.pvalues,est2.bse]))
    result.columns=['coef','t_test','p_test','std_error']
    z=np.polyfit(x,max_acc,1)
    p_hat=np.poly1d(z)(x)
    plt.plot(x,p_hat,"r--")
    text = f"$y={z[0]:0.3f}\\;x{z[1]:+0.3f}$"
    plt.gca().text(0.03, 0.85, text,transform=plt.gca().transAxes,
                    fontsize=14, verticalalignment='top')
    print(est2.summary())

    x2=sm.add_constant(x)
    est=sm.OLS(med_acc,x2)
    est2=est.fit()
    result=pd.DataFrame.transpose(pd.DataFrame([est2.params,est2.tvalues
,est2.pvalues,est2.bse]))
    result.columns=['coef','t_test','p_test','std_error']
    z=np.polyfit(x,med_acc,1)
    p_hat=np.poly1d(z)(x)
    plt.plot(x,p_hat,"r--")
    text = f"$y={z[0]:0.3f}\\;x{z[1]:+0.3f}$"
    plt.gca().text(0.03, 0.85, text,transform=plt.gca().transAxes,
                    fontsize=14, verticalalignment='top')
    print(est2.summary())

    x2=sm.add_constant(x)
    est=sm.OLS(min_acc,x2)
    est2=est.fit()
    result=pd.DataFrame.transpose(pd.DataFrame([est2.params,est2.tvalues
,est2.pvalues,est2.bse]))
    result.columns=['coef','t_test','p_test','std_error']
    z=np.polyfit(x,min_acc,1)
    p_hat=np.poly1d(z)(x)
    plt.plot(x,p_hat,"r--")
    text = f"$y={z[0]:0.3f}\\;x{z[1]:+0.3f}$"
    plt.gca().text(0.03, 0.85, text,transform=plt.gca().transAxes,
                    fontsize=14, verticalalignment='top')
    print(est2.summary())

    # DESIRED DECELERATION
    df['acc/dec (m/s2)'] = pd.to_numeric(df['acc/dec (m/s2)'],
errors='coerce')

    #speed between 0,10

```

```

df1=df[(df['speed (km/h)']>0) & (df['speed (km/h)']<10) &
(df['acc/dec (m/s2)']<-0.5) & (df['acc/dec (m/s2)']>=-3.0)]
min_dec1=df1['acc/dec (m/s2)'].quantile(0.95)
mean_dec1=df1['acc/dec (m/s2)'].quantile(0.5)
max_dec1=df1['acc/dec (m/s2)'].quantile(0.05)

#speed between 10,20
df1=df[(df['speed (km/h)']>=10) & (df['speed (km/h)']<20) &
(df['acc/dec (m/s2)']<-0.5) & (df['acc/dec (m/s2)']>=-3.5)]
min_dec2=df1['acc/dec (m/s2)'].quantile(0.95)
mean_dec2=df1['acc/dec (m/s2)'].quantile(0.5)
max_dec2=df1['acc/dec (m/s2)'].quantile(0.05)

#speed between 20,30
df1=df[(df['speed (km/h)']>=20) & (df['speed (km/h)']<30) &
(df['acc/dec (m/s2)']<-1) & (df['acc/dec (m/s2)']>=-4)]
min_dec3=df1['acc/dec (m/s2)'].quantile(0.95)
mean_dec3=df1['acc/dec (m/s2)'].quantile(0.5)
max_dec3=df1['acc/dec (m/s2)'].quantile(0.05)

#speed between 30,40
df1=df[(df['speed (km/h)']>=30) & (df['speed (km/h)']<40) &
(df['acc/dec (m/s2)']<-1) & (df['acc/dec (m/s2)']>=-4)]
min_dec4=df1['acc/dec (m/s2)'].quantile(0.95)
mean_dec4=df1['acc/dec (m/s2)'].quantile(0.5)
max_dec4=df1['acc/dec (m/s2)'].quantile(0.05)

#speed between 40,50
df1=df[(df['speed (km/h)']>=40) & (df['speed (km/h)']<50) &
(df['acc/dec (m/s2)']<-1) & (df['acc/dec (m/s2)']>=-4)]
min_dec5=df1['acc/dec (m/s2)'].quantile(0.95)
mean_dec5=df1['acc/dec (m/s2)'].quantile(0.5)
max_dec5=df1['acc/dec (m/s2)'].quantile(0.05)
max_dec=[max_dec1,max_dec2, max_dec3, max_dec4, max_dec5]
med_dec=[mean_dec1,mean_dec2, mean_dec3, mean_dec4, mean_dec5]
min_dec=[min_dec1,min_dec2, min_dec3, min_dec4, min_dec5]

x=[10,20,30,40,50]

fig=plt.figure()
ax1=fig.add_subplot(111)
ax1.scatter(x,max_dec,c='b',marker="s",label='Maximum')
ax1.scatter(x,med_dec,c='r',marker="s",label='Median')
ax1.scatter(x,min_dec,c='g',marker="s",label='Minimum')

```

```

fit_max = np.polyfit(x, max_dec, 1)
fit_func_max = np.polyld(fit_max)
fit_med = np.polyfit(x, med_dec, 1)
fit_func_med = np.polyld(fit_med)
fit_min = np.polyfit(x, min_dec, 1)
fit_func_min = np.polyld(fit_min)
# Plot the best-fit lines
ax1.plot(x, fit_func_max(x), 'b--', label='Best Fit (Maximum)')
ax1.plot(x, fit_func_med(x), 'r--', label='Best Fit (Median)')
ax1.plot(x, fit_func_min(x), 'g--', label='Best Fit (Minimum)')

equation_max = f'Best Fit Line (Max): y = {fit_max[0]:.2f}x + {fit_max[1]:.2f}'
equation_med = f'Best Fit Line (Med): y = {fit_med[0]:.2f}x + {fit_med[1]:.2f}'
equation_min = f'Best Fit Line (Min): y = {fit_min[0]:.2f}x + {fit_min[1]:.2f}'
ax1.text(2, fit_func_max(2), equation_max, fontsize=12, color='b',
verticalalignment='bottom', horizontalalignment='left')
ax1.text(2, fit_func_med(2), equation_med, fontsize=12, color='r',
verticalalignment='bottom', horizontalalignment='left')
ax1.text(2, fit_func_min(2), equation_min, fontsize=12, color='g',
verticalalignment='bottom', horizontalalignment='left')

plt.legend(bbox_to_anchor=(1.05,1),loc='best');
plt.xlabel('Speed (km/h)',fontsize=12)
plt.ylabel('Deceleration (m/s^2)',fontsize=12)
plt.title('Desired Deceleration')
plt.ylim((0.0,-4.0))
plt.xlim((0,60))
ax1.yaxis.grid()

plt.show()

x2=sm.add_constant(x)
est=sm.OLS(max_dec,x2)
est2=est.fit()
result=pd.DataFrame.transpose(pd.DataFrame([est2.params,est2.tvalues
,est2.pvalues,est2.bse]))
result.columns=['coef','t_test','p_test','std_error']
z=np.polyfit(x,max_dec,1)
p_hat=np.polyld(z)(x)
plt.plot(x,p_hat,"r--")
text = f"$y={z[0]:0.3f}\\;x{z[1]:+0.3f}$"
plt.gca().text(0.03, 0.85, text,transform=plt.gca().transAxes,

```

```

                                fontsize=14, verticalalignment='top')
print(est2.summary())
x2=sm.add_constant(x)
est=sm.OLS(med_dec,x2)
est2=est.fit()
result=pd.DataFrame.transpose(pd.DataFrame([est2.params,est2.tvalues
,est2.pvalues,est2.bse]))
result.columns=['coef','t_test','p_test','std_error']
z=np.polyfit(x,med_dec,1)
p_hat=np.polyld(z)(x)
plt.plot(x,p_hat,"r--")
text = f"$y={z[0]:0.3f}\\;x{z[1]:+0.3f}$"
plt.gca().text(0.03, 0.85, text,transform=plt.gca().transAxes,
                                fontsize=14, verticalalignment='top')
print(est2.summary())

x2=sm.add_constant(x)
est=sm.OLS(min_dec,x2)
est2=est.fit()
result=pd.DataFrame.transpose(pd.DataFrame([est2.params,est2.tvalues
,est2.pvalues,est2.bse]))
result.columns=['coef','t_test','p_test','std_error']
z=np.polyfit(x,min_dec,1)
p_hat=np.polyld(z)(x)
plt.plot(x,p_hat,"r--")
text = f"$y={z[0]:0.3f}\\;x{z[1]:+0.3f}$"
plt.gca().text(0.03, 0.85, text,transform=plt.gca().transAxes,
                                fontsize=14, verticalalignment='top')
print(est2.summary())

```

## Appendix E – VISSIM Travel Time Compiler Python Code

```
import os
import glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

# NORTHBOUND AM PEAK
# Read the CSV file from Google Colab
file_path = '/content/drive/My Drive/7. Corridor/AM Peak/Time/St-
Maurice Corridor_AM Peak_3rd Calib Travel Time NB.csv'
df = pd.read_csv(file_path)

df['SimRun'] = pd.to_numeric(df['SimRun'],
errors='coerce').astype('Int64')
df = df.dropna(subset=['SimRun'])

df['TimeInt'] = df['TimeInt'].str.extract('(\d+)').astype(float)

# Create a function to label simulation runs as days (7 days from
2022-07-08 to 2022-07-14)
def label_simulation_day(sim_run):
    if 1 <= sim_run <= 4:
        return '2022-07-08'
    elif 5 <= sim_run <= 8:
        return '2022-07-09'
    elif 9 <= sim_run <= 12:
        return '2022-07-10'
    elif 13 <= sim_run <= 16:
        return '2022-07-11'
    elif 17 <= sim_run <= 20:
        return '2022-07-12'
    elif 21 <= sim_run <= 24:
        return '2022-07-13'
    elif 25 <= sim_run <= 30:
        return '2022-07-14'
    else:
        return None
```

```

    # Apply the label_simulation_day function to create the
'SimulationDay' column
    df['SimulationDay'] = df['SimRun'].apply(label_simulation_day)

    # Create a function to label TimeInt as hours with 5-minute
intervals (from 8:00:00 AM to 10:00:00 AM)
    def label_hour(time_int):
        if 900 <= time_int < 1200:
            return '08:00:00'
        elif 1200 <= time_int < 1500:
            return '08:05:00'
        elif 1500 <= time_int < 1800:
            return '08:10:00'
        elif 1800 <= time_int < 2100:
            return '08:15:00'
        elif 2100 <= time_int < 2400:
            return '08:20:00'
        elif 2400 <= time_int < 2700:
            return '08:25:00'
        elif 2700 <= time_int < 3000:
            return '08:30:00'
        elif 3000 <= time_int < 3300:
            return '08:35:00'
        elif 3300 <= time_int < 3600:
            return '08:40:00'
        elif 3600 <= time_int < 3900:
            return '08:45:00'
        elif 3900 <= time_int < 4200:
            return '08:50:00'
        elif 4200 <= time_int < 4500:
            return '08:55:00'
        elif time_int == 4500:
            return '09:00:00'
        else:
            return None

    # Apply the label_hour function to create the 'Hour' column
    df['Hour'] = df['TimeInt'].apply(label_hour)

    # Group the data by 'SimulationDay', 'Hour', 'Intersection', and
calculate the average speed
    average_time1 = df.groupby(['SimulationDay', 'Hour',
'VehicleTravelTimeMeasurement'])['TravTm(All)'].mean().reset_index()

```

```

# Define the path to save the average speeds CSV in Google Drive
drive_save_path = '/content/drive/My Drive/7. Corridor/AM
Peak/Time/St-Maurice Corridor_AM Peak_3rd Calib Average Travel Time
NB.csv'

# Export the average speeds DataFrame as a CSV file to Google Drive
average_time1.to_csv(drive_save_path, index=False)

figsize = (12, 1.2 * len(average_time1['SimulationDay'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(average_time1, x='TravTm(All)', y='SimulationDay',
inner='stick', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)

average_time1['TravTm(All)'].plot(kind='hist', bins=11,
title='TravTm(All)')
plt.gca().spines[['top', 'right',]].set_visible(False)

# NORTHBOUND PM PEAK
# Read the CSV file from Google Colab
file_path = '/content/drive/My Drive/7. Corridor/PM Peak/Time/St-
Maurice Corridor_PM Peak_3rd Calib Travel Time NB.csv' # Update with the
actual file path
df = pd.read_csv(file_path)

df['SimRun'] = pd.to_numeric(df['SimRun'],
errors='coerce').astype('Int64')
df = df.dropna(subset=['SimRun'])

df['TimeInt'] = df['TimeInt'].str.extract('(\d+)').astype(float)

# Create a function to label simulation runs as days (7 days from
2022-07-08 to 2022-07-14)
def label_simulation_day(sim_run):
    if 1 <= sim_run <= 4:
        return '2022-07-08'
    elif 5 <= sim_run <= 8:
        return '2022-07-09'
    elif 9 <= sim_run <= 12:
        return '2022-07-10'
    elif 13 <= sim_run <= 16:
        return '2022-07-11'
    elif 17 <= sim_run <= 20:
        return '2022-07-12'
    elif 21 <= sim_run <= 24:

```

```

        return '2022-07-13'
    elif 25 <= sim_run <= 30:
        return '2022-07-14'
    else:
        return None

    # Apply the label_simulation_day function to create the
'SimulationDay' column
    df['SimulationDay'] = df['SimRun'].apply(label_simulation_day)

    # Create a function to label TimeInt as hours with 5-minute
intervals (from 03:30:00 PM to 04:30:00 PM)
    def label_hour(time_int):
        if 900 <= time_int < 1200:
            return '15:30:00'
        elif 1200 <= time_int < 1500:
            return '15:35:00'
        elif 1500 <= time_int < 1800:
            return '15:40:00'
        elif 1800 <= time_int < 2100:
            return '15:45:00'
        elif 2100 <= time_int < 2400:
            return '15:50:00'
        elif 2400 <= time_int < 2700:
            return '15:55:00'
        elif 2700 <= time_int < 3000:
            return '16:00:00'
        elif 3000 <= time_int < 3300:
            return '16:05:00'
        elif 3300 <= time_int < 3600:
            return '16:10:00 '
        elif 3600 <= time_int < 3900:
            return '16:15:00'
        elif 3900 <= time_int < 4200:
            return '16:20:00'
        elif 4200 <= time_int < 4500:
            return '16:25:00'
        elif time_int == 4500:
            return '16:30:00'
        else:
            return None

    # Apply the label_hour function to create the 'Hour' column
    df['Hour'] = df['TimeInt'].apply(label_hour)

```

```

    # Group the data by 'SimulationDay', 'Hour', 'Intersection', and
    calculate the average speed
    average_time2 = df.groupby(['SimulationDay', 'Hour',
'VehicleTravelTimeMeasurement'])['TravTm(All)'].mean().reset_index()

    # Define the path to save the average speeds CSV in Google Drive
    drive_save_path = '/content/drive/My Drive/7. Corridor/PM
Peak/Time/St-Maurice Corridor_PM Peak_3rd Calib Average Travel Time
NB.csv'

    # Export the average speeds DataFrame as a CSV file to Google Drive
    average_time2.to_csv(drive_save_path, index=False)

    figsize = (12, 1.2 * len(average_time2['SimulationDay'].unique()))
    plt.figure(figsize=figsize)
    sns.violinplot(average_time2, x='TravTm(All)', y='SimulationDay',
inner='stick', palette='Dark2')
    sns.despine(top=True, right=True, bottom=True, left=True)

    average_time2['TravTm(All)'].plot(kind='hist', bins=11,
title='TravTm(All)')
    plt.gca().spines[['top', 'right',]].set_visible(False)

    # SOUTHBOUND AM PEAK
    # Read the CSV file from Google Colab
    file_path = '/content/drive/My Drive/7. Corridor/AM Peak/Time/St-
Maurice Corridor_AM Peak 3rd Calib Travel Time SB.csv'
    df = pd.read_csv(file_path)

    df['SimRun'] = pd.to_numeric(df['SimRun'],
errors='coerce').astype('Int64')
    df = df.dropna(subset=['SimRun'])

    df['TimeInt'] = df['TimeInt'].str.extract('(\d+)').astype(float)

    # Create a function to label simulation runs as days (7 days from
2022-07-08 to 2022-07-14)
    def label_simulation_day(sim_run):
        if 1 <= sim_run <= 4:
            return '2022-07-08'
        elif 5 <= sim_run <= 8:
            return '2022-07-09'
        elif 9 <= sim_run <= 12:
            return '2022-07-10'
        elif 13 <= sim_run <= 16:

```

```

        return '2022-07-11'
    elif 17 <= sim_run <= 20:
        return '2022-07-12'
    elif 21 <= sim_run <= 24:
        return '2022-07-13'
    elif 25 <= sim_run <= 30:
        return '2022-07-14'
    else:
        return None

# Apply the label_simulation_day function to create the
'SimulationDay' column
df['SimulationDay'] = df['SimRun'].apply(label_simulation_day)

# Create a function to label TimeInt as hours with 5-minute
intervals (from 08:00:00 AM to 10:00:00 AM)
def label_hour(time_int):
    if 900 <= time_int < 1200:
        return '08:00:00'
    elif 1200 <= time_int < 1500:
        return '08:05:00'
    elif 1500 <= time_int < 1800:
        return '08:10:00'
    elif 1800 <= time_int < 2100:
        return '08:15:00'
    elif 2100 <= time_int < 2400:
        return '08:20:00'
    elif 2400 <= time_int < 2700:
        return '08:25:00'
    elif 2700 <= time_int < 3000:
        return '08:30:00'
    elif 3000 <= time_int < 3300:
        return '08:35:00'
    elif 3300 <= time_int < 3600:
        return '08:40:00'
    elif 3600 <= time_int < 3900:
        return '08:45:00'
    elif 3900 <= time_int < 4200:
        return '08:50:00'
    elif 4200 <= time_int < 4500:
        return '08:55:00'
    elif time_int == 4500:
        return '09:00:00'
    else:
        return None

```

```

# Apply the label_hour function to create the 'Hour' column
df['Hour'] = df['TimeInt'].apply(label_hour)

# Group the data by 'SimulationDay', 'Hour', 'Intersection', and
calculate the average speed
average_time3 = df.groupby(['SimulationDay', 'Hour',
'VehicleTravelTimeMeasurement'])['TravTm(All)'].mean().reset_index()

# Define the path to save the average speeds CSV in Google Drive
drive_save_path = '/content/drive/My Drive/7. Corridor/AM
Peak/Time/St-Maurice Corridor_AM Peak_3rd Calib Average Travel Time
SB.csv'

# Export the average speeds DataFrame as a CSV file to Google Drive
average_time3.to_csv(drive_save_path, index=False)

print(f'Average travel time CSV with day and hour labels saved to
Google Drive at: {drive_save_path}')

figsize = (12, 1.2 * len(average_time3['SimulationDay'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(average_time3, x='TravTm(All)', y='SimulationDay',
inner='stick', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)

average_time3['TravTm(All)'].plot(kind='hist', bins=11,
title='TravTm(All)')
plt.gca().spines[['top', 'right',]].set_visible(False)

# SOUTHBOUND PM PEAK
# Read the CSV file from Google Colab
file_path = '/content/drive/My Drive/7. Corridor/PM Peak/Time/St-
Maurice Corridor_PM Peak_3rd Calib Travel Time SB.csv' # Update with the
actual file path
df = pd.read_csv(file_path)

df['SimRun'] = pd.to_numeric(df['SimRun'],
errors='coerce').astype('Int64')
df = df.dropna(subset=['SimRun'])

df['TimeInt'] = df['TimeInt'].str.extract('(\d+)').astype(float)

# Create a function to label simulation runs as days (7 days from
2022-07-08 to 2022-07-14)

```

```

def label_simulation_day(sim_run):
    if 1 <= sim_run <= 4:
        return '2022-07-08'
    elif 5 <= sim_run <= 8:
        return '2022-07-09'
    elif 9 <= sim_run <= 12:
        return '2022-07-10'
    elif 13 <= sim_run <= 16:
        return '2022-07-11'
    elif 17 <= sim_run <= 20:
        return '2022-07-12'
    elif 21 <= sim_run <= 24:
        return '2022-07-13'
    elif 25 <= sim_run <= 30:
        return '2022-07-14'
    else:
        return None

# Apply the label_simulation_day function to create the
'SimulationDay' column
df['SimulationDay'] = df['SimRun'].apply(label_simulation_day)

# Create a function to label TimeInt as hours with 5-minute
intervals (from 03:30:00 PM to 04:30:00 PM)
def label_hour(time_int):
    if 900 <= time_int < 1200:
        return '15:30:00'
    elif 1200 <= time_int < 1500:
        return '15:35:00'
    elif 1500 <= time_int < 1800:
        return '15:40:00'
    elif 1800 <= time_int < 2100:
        return '15:45:00'
    elif 2100 <= time_int < 2400:
        return '15:50:00'
    elif 2400 <= time_int < 2700:
        return '15:55:00'
    elif 2700 <= time_int < 3000:
        return '16:00:00'
    elif 3000 <= time_int < 3300:
        return '16:05:00'
    elif 3300 <= time_int < 3600:
        return '16:10:00 '
    elif 3600 <= time_int < 3900:
        return '16:15:00'

```

```

        elif 3900 <= time_int < 4200:
            return '16:20:00'
        elif 4200 <= time_int < 4500:
            return '16:25:00'
        elif time_int == 4500:
            return '16:30:00'
        else:
            return None

    # Apply the label_hour function to create the 'Hour' column
    df['Hour'] = df['TimeInt'].apply(label_hour)

    # Group the data by 'SimulationDay', 'Hour', 'Intersection', and
    calculate the average speed
    average_time4 = df.groupby(['SimulationDay', 'Hour',
'VehicleTravelTimeMeasurement'])['TravTm(All)'].mean().reset_index()

    # Define the path to save the average speeds CSV in Google Drive
    drive_save_path = '/content/drive/My Drive/7. Corridor/PM
Peak/Time/St-Maurice Corridor_PM Peak_3rd Calib Average Travel Time
SB.csv'

    # Export the average speeds DataFrame as a CSV file to Google Drive
    average_time4.to_csv(drive_save_path, index=False)

    print(f'Average travel time CSV with day and hour labels saved to
Google Drive at: {drive_save_path}')

    figsize = (12, 1.2 * len(average_time4['SimulationDay'].unique()))
    plt.figure(figsize=figsize)
    sns.violinplot(average_time4, x='TravTm(All)', y='SimulationDay',
inner='stick', palette='Dark2')
    sns.despine(top=True, right=True, bottom=True, left=True)

    average_time4['TravTm(All)'].plot(kind='hist', bins=11,
title='TravTm(All)')
    plt.gca().spines[['top', 'right']].set_visible(False)

    # MERGED APPROACHES AM PEAK HISTOGRAM
    bin_edges = np.arange(60, 222, 12)
    plt.figure(figsize=(10, 6))
    combined_freq, bins, _ = plt.hist(
        np.concatenate([average_time1['TravTm(All)'],
average_time3['TravTm(All)']],
        bins=bin_edges,

```

```

        color='black',
        label='Combined Dataset',
        histtype='stepfilled',
        alpha=1.0
    )
    plt.gca().spines[['top', 'right']].set_visible(False)
    plt.yticks(np.arange(0, 81, 10))
    bin_centers = (bins[:-1] + bins[1:]) / 2
    for i in range(len(combined_freq)):
        plt.text(bin_centers[i], combined_freq[i], '', ha='center',
va='bottom')
    plt.title('Distribution of Simulated Travel Time Counts (3rd Calib
Model) - AM Peak')
    plt.xlabel('Travel Time (s)')
    plt.ylabel('Counts')

    plt.show()

# MERGED APPROACHES PM PEAK HISTOGRAM
bin_edges = np.arange(60, 222, 12)
plt.figure(figsize=(10, 6))
combined_freq, bins, _ = plt.hist(
    np.concatenate([average_time2['TravTm(All)'],
average_time4['TravTm(All)']],
    bins=bin_edges,
    color='black',
    label='Combined Dataset',
    histtype='stepfilled',
    alpha=1.0
)
    plt.gca().spines[['top', 'right']].set_visible(False)
    plt.yticks(np.arange(0, 81, 10))
    bin_centers = (bins[:-1] + bins[1:]) / 2
    for i in range(len(combined_freq)):
        plt.text(bin_centers[i], combined_freq[i], '', ha='center',
va='bottom')
    plt.title('Distribution of Simulated Travel Time Counts (3rd Calib
Model) - PM Peak')
    plt.xlabel('Travel Time (s)')
    plt.ylabel('Counts')

    plt.show()

```

## Appendix F – VISSIM Speed Compiler Python Code

```
import os
import glob
import pandas as pd
import numpy as np
from google.colab import drive
from matplotlib import pyplot as plt
import seaborn as sns

# Mount Google Drive
drive.mount('/content/drive')

# NORTHBOUND AM PEAK
# Read the CSV file from Google Colab
file_path = '/content/drive/My Drive/7. Corridor/AM Peak/Speed/St-
Maurice Corridor_AM Peak_3rd Calib Speed NB.csv'
df = pd.read_csv(file_path)

df['SimRun'] = pd.to_numeric(df['SimRun'],
errors='coerce').astype('Int64')
df = df.dropna(subset=['SimRun'])

df['TimeInt'] = df['TimeInt'].str.extract('(\d+)').astype(float)

# Create a function to label simulation runs as days (7 days from
2022-07-08 to 2022-07-14)
def label_simulation_day(sim_run):
    if 1 <= sim_run <= 4:
        return '2022-07-08'
    elif 5 <= sim_run <= 8:
        return '2022-07-09'
    elif 9 <= sim_run <= 12:
        return '2022-07-10'
    elif 13 <= sim_run <= 16:
        return '2022-07-11'
    elif 17 <= sim_run <= 20:
        return '2022-07-12'
    elif 21 <= sim_run <= 24:
        return '2022-07-13'
    elif 25 <= sim_run <= 30:
        return '2022-07-14'
    else:
        return None
```

```

    # Apply the label_simulation_day function to create the
'SimulationDay' column
    df['SimulationDay'] = df['SimRun'].apply(label_simulation_day)

    # Create a function to label TimeInt as hours with 5-minute
intervals (from 8:00:00 AM to 10:00:00 AM)
    def label_hour(time_int):
        if 900 <= time_int < 1200:
            return '08:00:00'
        elif 1200 <= time_int < 1500:
            return '08:05:00'
        elif 1500 <= time_int < 1800:
            return '08:10:00'
        elif 1800 <= time_int < 2100:
            return '08:15:00'
        elif 2100 <= time_int < 2400:
            return '08:20:00'
        elif 2400 <= time_int < 2700:
            return '08:25:00'
        elif 2700 <= time_int < 3000:
            return '08:30:00'
        elif 3000 <= time_int < 3300:
            return '08:35:00'
        elif 3300 <= time_int < 3600:
            return '08:40:00'
        elif 3600 <= time_int < 3900:
            return '08:45:00'
        elif 3900 <= time_int < 4200:
            return '08:50:00'
        elif 4200 <= time_int < 4500:
            return '08:55:00'
        elif time_int == 4500:
            return '09:00:00'
        else:
            return None

    # Apply the label_hour function to create the 'Hour' column
    df['Hour'] = df['TimeInt'].apply(label_hour)

    # Group the data by 'SimulationDay', 'Hour', 'Intersection', and
calculate the average speed
    average_speeds1 = df.groupby(['SimulationDay', 'Hour',
'DataCollectionMeasurement'])['SpeedAvgArith(All)'].mean().reset_index()

    # Define the path to save the average speeds CSV in Google Drive

```

```

drive_save_path = '/content/drive/My Drive/7. Corridor/AM
Peak/Speed/St-Maurice Corridor_AM Peak_3rd Calib Average Speeds NB.csv'
# Export the average speeds DataFrame as a CSV file to Google Drive
average_speeds1.to_csv(drive_save_path, index=False)

figsize = (12, 1.2 *
len(average_speeds1['DataCollectionMeasurement'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(average_speeds1, x='SpeedAvgArith(All)',
y='DataCollectionMeasurement', inner='stick', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)

average_speeds1['SpeedAvgArith(All)'].plot(kind='hist', bins=20,
title='SpeedAvgArith(All)')
plt.gca().spines[['top', 'right',]].set_visible(False)

# NORTHBOUND PM PEAK
# Read the CSV file from Google Colab
file_path = '/content/drive/My Drive/7. Corridor/PM Peak/Speed/St-
Maurice Corridor_PM Peak_3rd Calib Speed NB.csv'
df = pd.read_csv(file_path)

df['SimRun'] = pd.to_numeric(df['SimRun'],
errors='coerce').astype('Int64')
df = df.dropna(subset=['SimRun'])

df['TimeInt'] = df['TimeInt'].str.extract('(\d+)').astype(float)

# Create a function to label simulation runs as days (7 days from
2022-07-08 to 2022-07-14)
def label_simulation_day(sim_run):
    if 1 <= sim_run <= 4:
        return '2022-07-08'
    elif 5 <= sim_run <= 8:
        return '2022-07-09'
    elif 9 <= sim_run <= 12:
        return '2022-07-10'
    elif 13 <= sim_run <= 16:
        return '2022-07-11'
    elif 17 <= sim_run <= 20:
        return '2022-07-12'
    elif 21 <= sim_run <= 24:
        return '2022-07-13'
    elif 25 <= sim_run <= 30:
        return '2022-07-14'

```

```

        else:
            return None

    # Apply the label_simulation_day function to create the
'SimulationDay' column
    df['SimulationDay'] = df['SimRun'].apply(label_simulation_day)

    # Create a function to label TimeInt as hours with 5-minute
intervals (from 03:30:00 PM to 04:30:00 PM)
    def label_hour(time_int):
        if 900 <= time_int < 1200:
            return '15:30:00'
        elif 1200 <= time_int < 1500:
            return '15:35:00'
        elif 1500 <= time_int < 1800:
            return '15:40:00'
        elif 1800 <= time_int < 2100:
            return '15:45:00'
        elif 2100 <= time_int < 2400:
            return '15:50:00'
        elif 2400 <= time_int < 2700:
            return '15:55:00'
        elif 2700 <= time_int < 3000:
            return '16:00:00'
        elif 3000 <= time_int < 3300:
            return '16:05:00'
        elif 3300 <= time_int < 3600:
            return '16:10:00 '
        elif 3600 <= time_int < 3900:
            return '16:15:00'
        elif 3900 <= time_int < 4200:
            return '16:20:00'
        elif 4200 <= time_int < 4500:
            return '16:25:00'
        elif time_int == 4500:
            return '16:30:00'
        else:
            return None

    # Apply the label_hour function to create the 'Hour' column
    df['Hour'] = df['TimeInt'].apply(label_hour)

    # Group the data by 'SimulationDay', 'Hour', 'Intersection', and
calculate the average speed

```

```

average_speeds2 = df.groupby(['SimulationDay', 'Hour',
'DataCollectionMeasurement'])['SpeedAvgArith(All)'].mean().reset_index()

# Define the path to save the average speeds CSV in Google Drive
drive_save_path = '/content/drive/My Drive/7. Corridor/PM
Peak/Speed/St-Maurice Corridor_PM Peak_3rd Calib Average Speeds NB.csv'

# Export the average speeds DataFrame as a CSV file to Google Drive
average_speeds2.to_csv(drive_save_path, index=False)

figsize = (12, 1.2 *
len(average_speeds2['DataCollectionMeasurement'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(average_speeds2, x='SpeedAvgArith(All)',
y='DataCollectionMeasurement', inner='stick', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)

average_speeds2['SpeedAvgArith(All)'].plot(kind='hist', bins=20,
title='SpeedAvgArith(All)')
plt.gca().spines[['top', 'right',]].set_visible(False)

# SOUTHBOUND AM PEAK
# Read the CSV file from Google Colab
file_path = '/content/drive/My Drive/7. Corridor/AM Peak/Speed/St-
Maurice Corridor_AM Peak_3rd Calib Speed SB.csv'
df = pd.read_csv(file_path)

df['SimRun'] = pd.to_numeric(df['SimRun'],
errors='coerce').astype('Int64')
df = df.dropna(subset=['SimRun'])

df['TimeInt'] = df['TimeInt'].str.extract('(\d+)').astype(float)

# Create a function to label simulation runs as days (7 days from
2022-07-08 to 2022-07-14)
def label_simulation_day(sim_run):
    if 1 <= sim_run <= 4:
        return '2022-07-08'
    elif 5 <= sim_run <= 8:
        return '2022-07-09'
    elif 9 <= sim_run <= 12:
        return '2022-07-10'
    elif 13 <= sim_run <= 16:
        return '2022-07-11'
    elif 17 <= sim_run <= 20:

```

```

        return '2022-07-12'
    elif 21 <= sim_run <= 24:
        return '2022-07-13'
    elif 25 <= sim_run <= 30:
        return '2022-07-14'
    else:
        return None

    # Apply the label_simulation_day function to create the
'SimulationDay' column
    df['SimulationDay'] = df['SimRun'].apply(label_simulation_day)

    # Create a function to label TimeInt as hours with 5-minute
intervals (from 8:00:00 AM to 10:00:00 AM)
    def label_hour(time_int):
        if 900 <= time_int < 1200:
            return '08:00:00'
        elif 1200 <= time_int < 1500:
            return '08:05:00'
        elif 1500 <= time_int < 1800:
            return '08:10:00'
        elif 1800 <= time_int < 2100:
            return '08:15:00'
        elif 2100 <= time_int < 2400:
            return '08:20:00'
        elif 2400 <= time_int < 2700:
            return '08:25:00'
        elif 2700 <= time_int < 3000:
            return '08:30:00'
        elif 3000 <= time_int < 3300:
            return '08:35:00'
        elif 3300 <= time_int < 3600:
            return '08:40:00'
        elif 3600 <= time_int < 3900:
            return '08:45:00'
        elif 3900 <= time_int < 4200:
            return '08:50:00'
        elif 4200 <= time_int < 4500:
            return '08:55:00'
        elif time_int == 4500:
            return '09:00:00'
        else:
            return None

    # Apply the label_hour function to create the 'Hour' column

```

```

df['Hour'] = df['TimeInt'].apply(label_hour)

# Group the data by 'SimulationDay', 'Hour', 'Intersection', and
calculate the average speed
average_speeds3 = df.groupby(['SimulationDay', 'Hour',
'DataCollectionMeasurement'])['SpeedAvgArith(All)'].mean().reset_index()

# Define the path to save the average speeds CSV in Google Drive
drive_save_path = '/content/drive/My Drive/7. Corridor/AM
Peak/Speed/St-Maurice Corridor_AM Peak_3rd Calib Average Speeds SB.csv'

# Export the average speeds DataFrame as a CSV file to Google Drive
average_speeds3.to_csv(drive_save_path, index=False)

figsize = (12, 1.2 *
len(average_speeds3['DataCollectionMeasurement'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(average_speeds3, x='SpeedAvgArith(All)',
y='DataCollectionMeasurement', inner='stick', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)

average_speeds3['SpeedAvgArith(All)'].plot(kind='hist', bins=20,
title='SpeedAvgArith(All)')
plt.gca().spines[['top', 'right',]].set_visible(False)

# SOUTHBOUND PM PEAK
# Read the CSV file from Google Colab
file_path = '/content/drive/My Drive/7. Corridor/PM Peak/Speed/St-
Maurice Corridor_PM Peak_3rd Calib Speed SB.csv'
df = pd.read_csv(file_path)

df['SimRun'] = pd.to_numeric(df['SimRun'],
errors='coerce').astype('Int64')
df = df.dropna(subset=['SimRun'])

df['TimeInt'] = df['TimeInt'].str.extract('(\d+)').astype(float)

# Create a function to label simulation runs as days (7 days from
2022-07-08 to 2022-07-14)
def label_simulation_day(sim_run):
    if 1 <= sim_run <= 4:
        return '2022-07-08'
    elif 5 <= sim_run <= 8:
        return '2022-07-09'
    elif 9 <= sim_run <= 12:

```

```

        return '2022-07-10'
    elif 13 <= sim_run <= 16:
        return '2022-07-11'
    elif 17 <= sim_run <= 20:
        return '2022-07-12'
    elif 21 <= sim_run <= 24:
        return '2022-07-13'
    elif 25 <= sim_run <= 30:
        return '2022-07-14'
    else:
        return None

    # Apply the label_simulation_day function to create the
'SimulationDay' column
    df['SimulationDay'] = df['SimRun'].apply(label_simulation_day)

    # Create a function to label TimeInt as hours with 5-minute
intervals (from 03:30:00 PM to 04:30:00 PM)
    def label_hour(time_int):
        if 900 <= time_int < 1200:
            return '15:30:00'
        elif 1200 <= time_int < 1500:
            return '15:35:00'
        elif 1500 <= time_int < 1800:
            return '15:40:00'
        elif 1800 <= time_int < 2100:
            return '15:45:00'
        elif 2100 <= time_int < 2400:
            return '15:50:00'
        elif 2400 <= time_int < 2700:
            return '15:55:00'
        elif 2700 <= time_int < 3000:
            return '16:00:00'
        elif 3000 <= time_int < 3300:
            return '16:05:00'
        elif 3300 <= time_int < 3600:
            return '16:10:00 '
        elif 3600 <= time_int < 3900:
            return '16:15:00'
        elif 3900 <= time_int < 4200:
            return '16:20:00'
        elif 4200 <= time_int < 4500:
            return '16:25:00'
        elif time_int == 4500:
            return '16:30:00'

```

```

else:
    return None

# Apply the label_hour function to create the 'Hour' column
df['Hour'] = df['TimeInt'].apply(label_hour)

# Group the data by 'SimulationDay', 'Hour', 'Intersection', and
calculate the average speed
average_speeds4 = df.groupby(['SimulationDay', 'Hour',
'DataCollectionMeasurement'])['SpeedAvgArith(All)'].mean().reset_index()

# Define the path to save the average speeds CSV in Google Drive
drive_save_path = '/content/drive/My Drive/7. Corridor/PM
Peak/Speed/St-Maurice Corridor_PM Peak_3rd Calib Average Speeds SB.csv'

# Export the average speeds DataFrame as a CSV file to Google Drive
average_speeds4.to_csv(drive_save_path, index=False)

figsize = (12, 1.2 *
len(average_speeds4['DataCollectionMeasurement'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(average_speeds4, x='SpeedAvgArith(All)',
y='DataCollectionMeasurement', inner='stick', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)

average_speeds4['SpeedAvgArith(All)'].plot(kind='hist', bins=20,
title='SpeedAvgArith(All)')
plt.gca().spines[['top', 'right',]].set_visible(False)

# MERGED APPROACHES AM PEAK HISTOGRAM
plt.figure(figsize=(10, 6))
combined_freq, bins, _ = plt.hist(
    np.concatenate([average_speeds1['SpeedAvgArith(All)'],
average_speeds3['SpeedAvgArith(All)']]),
    bins=20,
    range=(20, 55),
    alpha=1.0,
    color='black',
    label='Combined Dataset',
    histtype='stepfilled',
    edgecolor='none'
)
plt.clf()
plt.bar(bins[:-1], combined_freq, width=(bins[1] - bins[0]),
color='black', alpha=1.0)
plt.xlabel('Speed (km/h)')

```

```

plt.ylabel('Counts')
plt.title('Distribution of Simulated Speed Counts (3rd Calib Model)
- AM Peak')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.yticks(np.arange(0, 61, 10))
plt.show()

# MERGED APPROACHES PM PEAK HISTOGRAM
plt.figure(figsize=(10, 6))
combined_freq, bins, _ = plt.hist(
    np.concatenate([average_speeds2['SpeedAvgArith(All)'],
average_speeds4['SpeedAvgArith(All)']],
    bins=20,
    range=(20, 55),
    alpha=1.0,
    color='black',
    label='Combined Dataset',
    histtype='stepfilled',
    edgecolor='none'
)
plt.clf()
plt.bar(bins[:-1], combined_freq, width=(bins[1] - bins[0]),
color='black', alpha=1.0)
plt.xlabel('Speed (km/h)')
plt.ylabel('Counts')
plt.title('Distribution of Simulated Speed Counts (3rd Calib Model)
- PM Peak')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.yticks(np.arange(0, 61, 10))
plt.show()

```

## Appendix G – SSAM Conflicts Coordinates Conversion and Cleanup Python Code

```
!pip install pandas matplotlib
!pip install gmaps
!pip install matplotlib geopandas shapely
!pip install mplcursors
!pip install ipyml
import pandas as pd
from google.colab import drive
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
import numpy as np
from sklearn.neighbors import BallTree
import gmaps
import gmaps.datasets
import folium
from folium.plugins import HeatMap
from ipywidgets import interactive
from math import sin, cos, sqrt, atan2, radians, degrees
import geopandas as gpd
from shapely.geometry import Point, box
from folium import plugins
from IPython.display import display
import mplcursors
from google.colab import output

drive.mount('/content/drive')

output.enable_custom_widget_manager()

# Enable interactive plotting in Google Colab
percentmatplotlib widget

# ALIGN SIMULATED CONFLICTS WITH REAL GEOCOORDINATES
# Load your CSV file with X & Y coordinates
df = pd.read_csv('/content/drive/My Drive/7. Corridor/AM
Peak/SSAM/3rd Calib Model/St Maurice Corridor_AM_3rd Calib_001.csv')

# Filter rows where ConflictType is 'crossing'
crossing_data = df[df['ConflictType'].str.lower() ==
'crossing'].copy()

# Manually input the cluster centers
```

```

cluster_centers = np.array([
    [389, -2837],
    [487, -2751],
    [602, -2649],
    [693, -2567],
    [880, -2401],
    [1033, -2262]
])

num_clusters = len(cluster_centers)

# Use K-means clustering to assign clusters based on manual centers
kmeans = KMeans(n_clusters=num_clusters, init=cluster_centers,
n_init=1, random_state=42)
crossing_data.loc[:, 'Cluster'] =
kmeans.fit_predict(crossing_data[['xMinPET', 'yMinPET']])

# Plot the data with cluster centers for crossing conflicts
fig, ax = plt.subplots(figsize=(10, 8))
sc = ax.scatter(crossing_data['xMinPET'], crossing_data['yMinPET'],
c=crossing_data['Cluster'], cmap='viridis', marker='o', label='Crossing
Conflicts')
centers = ax.scatter(cluster_centers[:, 0], cluster_centers[:, 1],
c='red', marker='X', s=200, label='Cluster Centers')

ax.set_title(f'Crossing Conflicts X-Y Plot with {num_clusters}
Cluster Centers')
ax.set_xlabel('xMinPET (X-coordinate)')
ax.set_ylabel('yMinPET (Y-coordinate)')
ax.legend()
ax.grid(True)

plt.show()

# Display the cluster centers for crossing conflicts
cluster_centers_df = pd.DataFrame(cluster_centers,
columns=['Center_X', 'Center_Y'])
cluster_centers_df = cluster_centers_df.sort_values(by=['Center_Y',
'Center_X']).reset_index(drop=True)

print("Cluster Centers for Crossing Conflicts:")
print(cluster_centers_df)

# SSAM Coordinates
cluster_centers = pd.DataFrame({

```

```

        'Center_X': [389, 1033],
        'Center_Y': [-2837, -2262]
    })

    # Display the cluster centers for crossing conflicts
    print("Cluster Centers for Crossing Conflicts:")
    print(cluster_centers)

    # Manually provide the two GPS coordinates for regression
    # Example coordinates; replace with your actual values
    gps_coordinates = np.array([
        [-72.54811356714647, 46.350883549367005],
        [-72.53975312861061, 46.3560497271529]
    ])

    # Ensure that the number of clusters matches the number of GPS
coordinates
    if cluster_centers.shape[0] != gps_coordinates.shape[0]:
        raise ValueError("Number of clusters and GPS coordinates must be
the same.")

    # Fit a linear regression model for Lon (X-axis)
    lon_regression_model = LinearRegression()
    lon_regression_model.fit(cluster_centers[['Center_X']],
gps_coordinates[:, 0])

    # Fit a linear regression model for Lat (Y-axis)
    lat_regression_model = LinearRegression()
    lat_regression_model.fit(cluster_centers[['Center_Y']],
gps_coordinates[:, 1])

    # Display the coefficients (slope and intercept) for Lon (X-axis)
    lon_slope = lon_regression_model.coef_[0]
    lon_intercept = lon_regression_model.intercept_
    print(f"Lon Regression: y = {lon_slope:.6f} * x +
{lon_intercept:.6f}")

    # Display the coefficients (slope and intercept) for Lat (Y-axis)
    lat_slope = lat_regression_model.coef_[0]
    lat_intercept = lat_regression_model.intercept_
    print(f"Lat Regression: y = {lat_slope:.6f} * x +
{lat_intercept:.6f}")

    # Function to predict GPS coordinates based on X, Y cluster centers

```

```

def predict_gps_coordinates(cluster_coordinates):
    lon_prediction =
lon_regression_model.predict(np.array([[cluster_coordinates[0]]]))[0]
    lat_prediction =
lat_regression_model.predict(np.array([[cluster_coordinates[1]]]))[0]
    return lon_prediction, lat_prediction

# Apply the function to predict GPS coordinates for each cluster
center
cluster_centers['Predicted_GPS_Lon'],
cluster_centers['Predicted_GPS_Lat'] = zip(*cluster_centers[['Center_X',
'Center_Y']].apply(predict_gps_coordinates, axis=1))

# Display the predicted GPS coordinates for each cluster center
print("Predicted GPS coordinates for Cluster Centers:")
print(cluster_centers[['Center_X', 'Center_Y', 'Predicted_GPS_Lon',
'Predicted_GPS_Lat']])

# Assuming lon_regression_model and lat_regression_model are
previously defined
lon_regression_model = LinearRegression()
lat_regression_model = LinearRegression()

# Example for lon_regression_model training
lon_regression_model.fit(result_table[['Center_X']],
result_table['GPS_Lon'])

# Example for lat_regression_model training
lat_regression_model.fit(result_table[['Center_Y']],
result_table['GPS_Lat'])

# Initialize new columns for Predicted_GPS_Lon and Predicted_GPS_Lat
df['Predicted_GPS_Lon'] = np.nan
df['Predicted_GPS_Lat'] = np.nan

# Iterate over rows in df
for index, row in df.iterrows():
    # Use the Lon and Lat regression models to predict GPS Lon and
Lat
    predicted_lon =
lon_regression_model.predict([[row['xMinPET']]])[0]
    predicted_lat =
lat_regression_model.predict([[row['yMinPET']]])[0]

    # Update the new columns in df with predicted values

```

```

df.at[index, 'Predicted_GPS_Lon'] = predicted_lon
df.at[index, 'Predicted_GPS_Lat'] = predicted_lat

# Extract columns in df with specific names
calib_df = df[['trjFile', 'xMinPET', 'yMinPET', 'zMinPET', 'PET',
'ConflictAngle', 'ClockAngle', 'ConflictType', 'Predicted_GPS_Lon',
'Predicted_GPS_Lat']]

# Save Results
calib_df.to_csv('/content/drive/My Drive/7. Corridor/AM
Peak/SSAM/3rd Calib Model/St Maurice Corridor_AM_3rd Calib_001_GPS
COORD.csv', index=False)

# CLEANUP ADDITIONAL SIMULATED CONFLICTS OUTSIDE OF LIDAR COVER
# Create a base map centered at the mean latitude and longitude
center_lat = calib_df['Predicted_GPS_Lat'].mean()
center_lon = calib_df['Predicted_GPS_Lon'].mean()
map_center = [center_lat, center_lon]

# Create a Folium map
m = folium.Map(location=map_center, zoom_start=15)

# Add Google Maps satellite layer to the map
google_satellite_layer = folium.TileLayer(
    tiles='https://mt1.google.com/vt/lyrs=s&x={x}&y={y}&z={z}',
    attr='Google Satellite',
    name='Google Satellite',
    overlay=True,
).add_to(m)

# Define intersection centers and buffer size in meters
intersection_buffers = {
    'Laviolette': {'center': (46.350883549367005, -
72.54811356714647), 'buffer_size': 60.0},
    'Francois Xavier': {'center': (46.35166548724671, -
72.54686701278713), 'buffer_size': 45.0},
    'Angele': {'center': (46.35254956209795, -72.54537632663315),
'buffer_size': 30.0},
    'Paul': {'center': (46.353292405572454, -72.54417348556174),
'buffer_size': 30.0},
    'Nicolas Perrot': {'center': (46.35477840255693, -
72.54171687131769), 'buffer_size': 55.0},
    'Chenau': {'center': (46.3560497271529, -72.53975312861061),
'buffer_size': 50.0}

```

```

}

# Function to create an interactive heatmap for all predicted points
def update_heatmap_and_table(radius):
    # Create a Folium map
    m = folium.Map(location=map_center, zoom_start=15)

    # Add Google Maps satellite layer to the map
    google_satellite_layer.add_to(m)
    # Plot calib_df points as heatmap with adjusted radius
    heat_data = [[point['Predicted_GPS_Lat'],
point['Predicted_GPS_Lon']] for index, point in calib_df.iterrows()]
    heatmap = HeatMap(data=heat_data, radius=radius, blur=3,
name='Predicted Points')
    m.add_child(heatmap)

    # Add markers and rectangles for each intersection
    for intersection, params in intersection_buffers.items():
        # Add marker for intersection center
        folium.Marker(location=params['center'],
popup=intersection).add_to(m)

        # Calculate rectangle bounds based on buffer size
        half_buffer_size_deg = params['buffer_size'] / 111000 #
Approximate conversion from meters to degree
        rectangle = folium.Rectangle(
            bounds=[
                (params['center'][0] - half_buffer_size_deg,
params['center'][1] - half_buffer_size_deg),
                (params['center'][0] + half_buffer_size_deg,
params['center'][1] + half_buffer_size_deg)
            ],
            color='blue',
            fill=True,
            fill_color='blue',
            fill_opacity=0.3,
            popup=f"{intersection} Buffer"
        ).add_to(m)

    # Display the map
    display(m)

# Use interactive with explicit widgets
interactive_plot = interactive(update_heatmap_and_table, radius=(1,
10, 1))

```

```

# Display the interactive plot
display(interactive_plot)

# Initialize clean_calib_df in the global scope
clean_calib_df = pd.DataFrame()

# Function to check if a point is within any rectangle
def is_within_any_rectangle(point, rectangles):
    for rectangle in rectangles:
        if point.within(rectangle):
            return True
    return False

# Function to create an interactive heatmap for all predicted points
and export conflicts within rectangles
def update_heatmap_and_table(radius):
    global clean_calib_df # Declare clean_calib_df as a global
variable

    # Initialize a GeoDataFrame for rectangles
    gdf_rectangles = gpd.GeoDataFrame(columns=['intersection',
'geometry'])

    # Add markers and rectangles for each intersection
    for intersection, params in intersection_buffers.items():
        # Calculate rectangle bounds based on buffer size
        half_buffer_size_deg = params['buffer_size'] / 111000 #
Approximate conversion from meters to degrees
        rectangle_bounds = (
            params['center'][1] - half_buffer_size_deg, # minx
            params['center'][0] - half_buffer_size_deg, # miny
            params['center'][1] + half_buffer_size_deg, # maxx
            params['center'][0] + half_buffer_size_deg # maxy
        )

        # Create a rectangle geometry
        rectangle_geometry = box(*rectangle_bounds)

        # Add rectangle to GeoDataFrame
        gdf_rectangles = pd.concat([gdf_rectangles,
gpd.GeoDataFrame([{'intersection': intersection, 'geometry':
rectangle_geometry}]), ignore_index=True)

    # Check if each conflict point falls within any rectangle

```

```

        calib_df['Within Limits'] = calib_df.apply(
            lambda row: 'Yes' if
is_within_any_rectangle(Point(row['Predicted_GPS_Lon'],
row['Predicted_GPS_Lat']), gdf_rectangles['geometry']) else 'No',
            axis=1
        )

        # Create a DataFrame for conflicts_within_rectangles
        clean_calib_df = calib_df[calib_df['Within Limits'] ==
'Yes'].copy()
        update_heatmap_and_table(radius=3)

```

## Appendix H – VISSIM COM Python Code

```
from __future__ import print_function
import os
import win32com.client as com
import csv
import threading
import time # Import time module for sleep

def load_parameters_from_csv(csv_file):
    with open(csv_file, 'r', encoding='utf-8-sig') as file:
        reader = csv.DictReader(file)
        parameters_list = [dict(row) for row in reader]
    return parameters_list

def ensure_directory_exists(directory):
    if not os.path.exists(directory):
        os.makedirs(directory)

def run_simulation(network_file, layout_file, parameters, base_report_dir,
run_number):
    # Create a new instance of the Vissim COM object
    vissim = com.Dispatch("Vissim.Vissim")

    # Load network and layout files
    vissim.LoadNet(network_file)
    vissim.LoadLayout(layout_file)

    # Counter for simulation runs
    simulation_run_count = 0

    for param_combination in parameters:
        driving_behavior = vissim.Net.DrivingBehaviors.ItemByKey(1)

        print(f"\nRun {run_number} - Parameter Combination:")
        for key, value in param_combination.items():
            print(f"{key}: {value}")

        for key, value in param_combination.items():
            try:
                if key != 'CarFollowModType':
                    if key != 'LnChgRule' and key != 'DesLatPos':
                        driving_behavior.SetAttValue(key, float(value))
                    else:
                        driving_behavior.SetAttValue(key, value)
            except Exception as e:
```

```

        print(f"Error setting attribute {key} with value {value}:
{e}")

    # Run simulation
    vissim.Simulation.RunContinuous()

    # Introduce a sleep delay to allow the simulation to settle
    time.sleep(5)

    # Increment the simulation run count
    simulation_run_count += 1
    print(f"Simulation Run {simulation_run_count} completed.")

    # Revert parameter values to the original state
    for key, original_value in param_combination.items():
        try:
            if key != 'CarFollowModType':
                if key != 'LnChgRule' and key != 'DesLatPos':
                    driving_behavior.SetAttValue(key,
float(original_value))
                else:
                    driving_behavior.SetAttValue(key, original_value)
            except Exception as e:
                print(f"Error reverting attribute {key} to original value
{original_value}: {e}")

    # Save and close VISSIM
    vissim.Simulation.Stop()
    vissim.SaveNet()
    vissim = None

if __name__ == "__main__":
    # Load parameter combinations from CSV files
    csv_file_path1 = r"C:\Users\natal\OneDrive\Desktop\Masters\Model
Parameters\Corridor_Effectiveness Check AM.csv"
    # csv_file_path2 = r"C:\Users\natal\OneDrive\Desktop\Masters\Model
Parameters\Corridor_Effectiveness Check PM.csv"

    parameters1 = load_parameters_from_csv(csv_file_path1)
    # parameters2 = load_parameters_from_csv(csv_file_path2)

    # Specify the base directory for reports
    base_report_dir = r'C:\Users\natal\OneDrive\Desktop\Masters\VISSIM
Models\St Maurice Corridor Effectiveness Model\Reports'

```

```

        # Specify the paths for the network and layout files
        network_file1 = r"C:\Users\natal\OneDrive\Desktop\Masters\VISSIM
Models\St Maurice Corridor Effectiveness Model\St Maurice
Corridor_AM_Effect.inpx"
        layout_file1 = r"C:\Users\natal\OneDrive\Desktop\Masters\VISSIM
Models\St Maurice Corridor Effectiveness Model\St Maurice
Corridor_AM_Effect.layx"

        # network_file2 = r"C:\Users\natal\OneDrive\Desktop\Masters\VISSIM
Models\St Maurice Corridor Effectiveness Model\St Maurice
Corridor_PM_Effect.inpx"
        # layout_file2 = r"C:\Users\natal\OneDrive\Desktop\Masters\VISSIM
Models\St Maurice Corridor Effectiveness Model\St Maurice
Corridor_PM_Effect.layx"

        # Ensure the base report directory exists
        ensure_directory_exists(base_report_dir)

        # Create threads for parallel execution
        thread1 = threading.Thread(target=run_simulation, args=(network_file1,
layout_file1, parameters1, base_report_dir, 1))
        # thread2 = threading.Thread(target=run_simulation,
args=(network_file2, layout_file2, parameters2, base_report_dir, 2))

        # Start the threads
        thread1.start()
        # thread2.start()

        # Wait for both threads to finish
        thread1.join()
        # thread2.join()

        print("Simulation completed.")

```