Semi-Automated Microscopic Traffic Flow Simulation Development Using Smart City Data

by

Qiao (Ray) Lei

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Civil Engineering

Waterloo, Ontario, Canada, 2021
© Qiao (Ray) Lei 2021

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 models have been widely used by transportation planners and engineers for conducting various road network planning and traffic engineering tasks. Before a traffic simulation model is applied, it must be calibrated carefully to the ground truth using traffic data collected in the field. Due to data limitation, traffic simulation models are often calibrated on the basis of macroscopic traffic measures such as traffic volume, travel time, and traffic stream fundamentals. In recent years, emerging smart city sensor technologies, such as video cameras, Bluetooth/Wi-Fi detectors, and Lidar, are enabling continuous collection of large volume, high-resolution trajectory data of road users, making it possible to estimate some behaviour parameters of traffic simulation models directly from these data. This thesis research is intended to explore this opportunity with the specific objective of developing methodology to estimate traffic simulation model parameters from smart city data with automated or semi-automated calibration procedures. A comprehensive set of calibration procedures are proposed, including both direct methods of estimating model parameters from data and indirect methods of estimating some model parameters using an optimization algorithm. Most of the proposed procedures are designed in such a way so that they can be completed in a semi-automated way with multiple Python scripts.

The developed methodology is illustrated in a case study involving calibration of a VISSIM simulation model using an available dataset of vehicle trajectories - NGSIM (Next Generation Simulation) traffic data. While most parameters can be directly determined from the dataset, some parameters from the selected parameter set are determined using Neutral Network. The modelling results suggested that the best performed parameter set generates less than 10% error compared to the field measurements in term of travel time and speed, respectively.

Acknowledgements

I would like to thank my supervisors, Dr. Chris Bachmann, and Dr. Liping Fu, for guiding me through the process to complete this thesis. A special thanks goes to Dr. Chris Bachmann for providing lots of supports to me both academically and emotionally throughout the tough pandemic period.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Traffic Simulation Models

Traffic simulation models are analysis tools that aim to reproduce real-world traffic behaviour in a computer environment. They are widely used in roadway design, urban planning, and traffic management (Alexiadis & Chandra, 2004). Compared to a traditional traffic analysis tool such as the Highway Capacity Manual (National Research Council (U.S.). Transportation Research Board, 2010), traffic simulation models are more effective at analyzing congested traffic systems and can help modellers understand congestion formation and its impact on the wider system behaviour. In addition, some real-world factors that are not considered in the traditional traffic analysis tools, such as interaction between individual vehicles and variability in driver/vehicle characteristics, can be modelled in traffic simulation models.

Traffic simulation models are generally divided into three categories according to their modelling scope: macroscopic, microscopic, and mesoscopic. The macroscopic modeling of traffic flows is based on the continuum theory of traffic flow, which entails the description of the time(t)–space(x) evolution of the three major traffic stream variables characterizing traffic flows in a macroscopic perspective: speed (u), flow (q), and density (k). It is assumed that between two locations in a motorway section without entrances and exits, the number of vehicles is conserved (Barceló, 2010). This theory is represented by the conservation equation in hydrodynamics:

$$\frac{\partial q}{\partial x} + \frac{\partial k}{\partial t} = 0 \tag{1}$$

where q is flow ($\frac{vehicle}{hr}$), x is distance (km), k is density ($\frac{vehicle}{km}$) and t is time (hr). Equation 1 can be extended to include entrance and off ramps.

For a given road section, it is empirically known that speed is a function of density. For example, Greenshields (1934) proposed a linear relationship between speed and density:

$$u = u_f - \frac{u_f}{k_j} k \tag{2}$$

where u is speed (km/hr), u_f is free flow speed km/hr), k is density ($vehicle/km$), and $k_j$is jam density ($vehicle/km$).

The Greenshields model is now used as a textbook model because later studies found that the linear speed-density relationship may not be the best representation of the observed data. Many researchers started to develop non-linear models for speed-density relationship (Greenberg, 1959 and Underwood, 1959). Van Aerde (1995) proposed a non-linear model represented by five equations. Although Van Aerde model requires more parameters, it provides more degrees of freedom to reflect different traffic behavior across different roadway facilities (Rakha & Crowther, 2002).

Finally, the relationship between the macroscopic traffic flow variables is governed by a definitional equation:

$$q = uk \tag{3}$$

where q is traffic flow rate ($vehicle/hr$), u is speed (km/h), and k is density ($vehicle/km$).

A macroscopic simulation model takes place on a road section-by-section basis, repeatedly applying equations 1 to 3 through the simulation horizon. For example, Payne (1979) developed FREFLOW, Haj Salem et al. (1994) developed METACOR, and Papageorgiou et al. (2010) developed METANET. Similarly, Daganzo (1994) proposed a cell transmission model (CTM) that describes the traffic's evolution over time (e.g. traffic building, propagation, and dissipation of queues) by applying flow-density relationship to each section of road. Due to limited levels of detail, they do not have the ability to analyze transportation improvements in as much detail as the microscopic models. Therefore, use of macroscopic simulation models is relatively uncommon in practice (Papageorgiou et al. 2010).

Microscopic models simulate the movement of individual vehicles based on car-following and lane-changing theories. Pipe (1953) developed car-following model based on the concept of distance headway. He pointed out that the following vehicle needs to maintain a safe distance with the vehicle at least the length of a car for every ten miles per hour of speed at which the following vehicle is traveling. In the late 1950s, the

General Motors Group developed a series of mathematical models based on extensive field experiments. The main idea of GM models is that the driver's response to accelerate and deaccelerate is in proportion to the magnitude of the stimulus occurring at time t and begins after a time lag T. This theory can be expressed as a stimulus-response function:

$$Response\ (t+T) = Sensitivity \times Stimulus\ (t) \tag{4}$$

By utilizing Equation 4, the acceleration or deceleration of the following vehicle can be modelled as Equation 5:

$$a_{n+1}^{t} = \left[\alpha \frac{(v_{n+1}^{t+T})^{m}}{(x_n^t - x_{n+1}^t)^{l}}\right][v_n^t - v_{n+1}^t] \tag{5}$$

where a_{n+1}^{t} is the acceleration(m/s^2) of following vehicle at time t, T is the reaction time, x_n^t and x_{n+1}^t are locations (m) of leading vehicle and following vehicle, v_n^t and v_{n+1}^t are speeds (m/s) of leading vehicle and following vehicle, *l* is a distance headway exponent, *m* is a speed exponent, and α is a sensitivity coefficient. The follower's response to leading vehicle is represented by acceleration or deceleration and the stimulus is represented by the variation in the relative speeds. Gipps (1981) improves the stimulus-response model by imposing limitations for desired speed, maximum braking, safety distance, and reaction time.

A different approach was taken by Wiedemann (1974) who proposed the psycho-physical models that have been implemented in several popular microsimulation software packages such as VISSIM and PARAMICS (PTV AG, 2020). These models assume that the behaviour of the following vehicle is not influenced by the speed difference at large spacing; at small spacing, the behaviour of the following vehicle is influenced by the combination of relative speeds and distance headways.

In the microscopic modelling process, stochastic arrivals are applied for vehicles entering a transportation network and vehicles are tracked through the network over small time intervals. Also, upon entry, each vehicle is assigned a destination, a vehicle type, and a driver type according to the modelling settings. Hence, microscopic models require much more data and computing power than macroscopic models. Nonetheless, microscopic simulation models are usually used for the operational design of transportation systems

with several intersections or major corridors. Due to the extensive and complex calculations required for microscopic traffic simulations, software is needed. PTV VISSIM is one of the leading microscopic simulation programs for modeling multimodal transport operations. It is used world-wide for traffic studies, city planning, and development evaluation purposes. VISSIM is used as the microscopic traffic flow simulation tool in this research.

Mesoscopic simulation models combine the properties of both microscopic and macroscopic simulation models. As in microscopic models, the mesoscopic models' unit of traffic flow is the individual vehicle. Their movement, however, follows the approach of the macroscopic models and is governed by the average speed on the travel link (Alexiadis & Chandra, 2004). For example, in the DYNASMART mesoscopic simulation model, the section speeds are usually calculated using the Greenshields speed-density relationship and each individual traveler will seek their best path (Jayakrishan et al., 1994). The INTEGRATION model applied a similar approach in which the individual vehicle desired speed is determined based on a link specific microscopic car following relationship that is calibrated macroscopically to yield the appropriate target aggregate speed-flow attributes for that particular link (Van Aerde, 1996). Dynameq also uses a similar approach in which the optimal assignment for each individual vehicle is determined based on the simulated path travel times (Mahut and Florian, 2010).

1.2 Smart City Data Applications

A smart city is an urban area that uses information and communication technologies to collect data to improve the operational efficiency, quality of government services and societal welfare (Rouse, 2019). The development of smart city technology is stimulating significant changes in urban transportation management with many application examples such as adaptive traffic signal control, active pedestrian safety measures, and innovative infrastructure development projects.

Congestion is one of the most prominent issues of urban transportation management. Traffic congestion not only reduces mobility but also has negative impacts on the environment, public health, and quality of life. Smart city technology can help cities

mitigate congestion by monitoring traffic in real-time, providing travel advice to road users, and adjusting signal timing systems to better facilitate traffic (Guo et al., 2020).

Another use of smart city technology that benefits the transportation system is road safety management. For example, while traditional data collection only records the number of reported accidents at specific areas, smart city technology can identify hot spots for both accidents and near-miss events (i.e., conflicts). This information can be used to develop risk mitigation strategies before tragic accidents occur (Mer Group, 2020).

Data collected by smart city technology can also help governing bodies to make evidence-based decisions. With substantial smart city data such as traffic volume, travel time, and road user behaviours, governments can make more educated decisions on urban planning and infrastructure improvements.

In the past, traffic data were often collected by loop detectors, which can detect a vehicle and determine its speed when passing over the induction loops. However, this method only collects data from the vehicles for a brief moment and it can not track how vehicles move along the roadway. More recently, Bluetooth/Wi-Fi detector is used to collect traffic data because the network with multiple Bluetooth/Wi-Fi detectors can track a vehicle's movement such as average travel time within a road section and turning movements by tracking the MAC address of electronic devices on board (Hidayat et al., 2018). However, this technology still cannot record the more detailed continuous movement of the vehicle in small timesteps. In the last decade, video cameras have become popular due to its wide range of uses. Tracking algorithms can recognize different types of road users from the video footage and track the continuous movement of objects. However, this method it limited under adverse weather conditions and the video footage are often protected due to privacy issues. In addition, because the video footage is a 2D representation of the 3D real-world, the video needs to be properly calibrated, therefore the accuracy of the traffic data is criticized by many researchers (Coifman and Li, 2017). While video cameras are still a major technology used for traffic data collection at present, more advanced smart city technologies such as Radar (Radio Detection and Ranging) and LiDAR (Light Detection and Ranging) are emerging. These technologies emit radio waves or light and receive a returned signal after being bounced

back from the objects in view. LiDAR is capable of producing high-resolution imaging to replace video camera data while most Radar is less capable. Table 1 lists the types of traffic data collected by different technologies.

Table 1 Traffic Data Collection Technologies

Technology	Trajectory	Traffic Volume	Segment Travel time	Spot Speed	Turning movement counts	Lane change	Data source (Example)
Loop detector	×	✓	✓	×	×	×	Government Agencies, NGSIM
Bluetooth/ Wi-Fi detector	×	×	✓	×	×	×	SMATS
Video Camera	✓	✓	✓	✓	✓	✓	Miovision, NGSIM
Radar/ LiDAR	✓	✓	✓	✓	✓	✓	Blue city

In the current practice, smart city traffic data are mostly collected by video cameras and Radar/ LiDAR with some supplementary data from loop detectors and Bluetooth/Wi-Fi detectors. All sensors in the road network collect data concurrently and over long time-periods, allowing for time-synchronized datasets reflecting true and complete observations of traffic conditions in the road network over time. Therefore, smart city traffic data provide more detailed information of traffic conditions compared to traditional data collection methods.

1.3 Calibration of Microscopic Simulation Models Using Smart City Data

Although traffic simulation modelling has already been used in practice for decades to help planners and engineers assess design and management alternatives, it is often viewed by non-modellers as an inexact science at best, and as an unreliable “black-box” technology at worst (Hellinga, 1998). Traffic simulation models must be calibrated to local conditions before they can be used as a trustworthy traffic analysis tool. Historically, the main challenge to calibration has been very limited data that reflect real-world traffic conditions and some data (e.g. driver behaviours) are hard to collect (Wunderlich et al., 2019). Many calibration studies have been done for traffic simulation modelling. However, most of these studies determined model parameter values by

solving an optimization problem with the objective of minimizing the differences between simulated and observed aggregated traffic behaviors (e.g., traffic volume, travel time, and speed-density relationship) (Ma and Abdulhai, 2002; Kim, et al., 2005; Park and Qi, 2005; Yu et al., 2006; Lidbe et al., 2017) while some studies only estimated very few VISSIM parameters with a relatively small sample size of traffic data (Lu et al., 2016).

In recent years, the emerging smart city technologies provide an opportunity towards better traffic simulation modelling by providing more reliable and detailed traffic data for input calibration and validation. Microscopic simulation models are important tools to utilize the smart city data and design solutions for different transportation improvements. With the availability of smart city data, it becomes possible to obtain the characteristics of each individual vehicle such as vehicle speed, vehicle acceleration, following distance, etc., and use this information to directly calibrate the constituent user behaviour models of microscopic simulation models. However, the smart city data are often used as measure of effectiveness to validate the calibrated model instead of model calibration.

While the development of smart city technology gives transportation engineers an opportunity towards better modelling of the real-world traffic, there is currently no guidelines or recommended processes to determine what information can be extracted from smart city data and how to incorporate these data to build more representative traffic simulation models.

1.4 Research Objective

This research has three primary objectives, which aim to improve the overall veracity and efficiency of the microscopic traffic simulation modelling process. The first objective is to make microscopic traffic simulation models better reflect the real-world traffic conditions by leveraging smart city data. The second objective is to automate the modelling process as much as possible by developing semi-automated calibration techniques. The third objective is to propose a standardized calibration method for microscopic simulation models using smart city data for future applications. These objectives involve the following major research tasks:

- Literature Review
- Data analysis
- Network development
- Parameter selection
- Sensitivity analysis
- Parameter determination
- Model calibration
- Model evaluation

A literature review is conducted on the selection of high-priority microscopic traffic simulation parameters, and the associated methods used to determine their values. Data analysis consists of analyzing the collected smart city traffic data and extracting the elements required for modelling. A VISSIM model of the transportation network within a study area is built and required parameters for modelling are input to VISSIM. The model outputs are compared with the observed traffic measures to validate the model. Modelling parameters without real-world observations are calibrated to minimize the difference between modelling results and the real-world scenario using an Artificial Neural Network (ANN). The data input and calibration processes are automated as much as possible using scripts and algorithms.

1.5 Scope

This research uses traffic data collected from an arterial section on Peachtree Street - a north-south urban arterial road in Atlanta, GA. The section selected for data collection crosses the central business district of Midtown Atlanta, which can represent the typical urban traffic condition in north America large cities. This research focuses on the traffic conditions of the urban area. Analysis of freeway traffic is not within the scope of this thesis. The traffic data was collected in 2006 by Federal Highway Administration (FHWA). This dataset was selected because it represents the types of data that could be expected from smart city sensors such as video cameras and Lidar. The data was collected by synchronized video cameras, including high-resolution vehicle trajectories that include many attributes necessary for determining VISSIM parameters. The quality

of the data may not be as accurate as data collected by current data collection technologies. However, the data quality does not impact the results of this research because this research focuses on developing methodologies to calibrate VISSIM parameters. The microscopic traffic simulation software used in this study is VISSIM. Implementation in other microscopic traffic simulation software is not considered.

1.6 Structure of Thesis

The structure of the thesis is as follows:

Chapter 2 provides an in-depth review of the literature on VISSIM model calibrations. The first part discusses how critical VISSIM parameters are selected. This review includes the introduction of a typical parameter selection process and a summary of parameters used by other studies. The second part of Chapter 2 focuses on different traffic data collection methods and how driving behaviours are determined from the field data. The last part of Chapter 2 introduces different VISSIM model calibration methods from early approaches to more recent optimization algorithms.

Chapter 3 describes the proposed methodology for microscopic traffic simulation model calibration. The data used for this study come from the Next Generation Simulation (NGSIM) dataset. An urban arterial corridor is selected as the study site. The VISSIM network is constructed based on the signal plan and road geometry when the traffic data were collected. The parameters selected for this study focus on vehicle performance and driving behaviour models including following behaviour, car following model (Wiedemann 74) and lane changing. Then an ANOVA (analysis of variance) sensitivity test is proposed to identify key parameters and eliminate parameters that have less effect on model results. However, for the sake of developing procedures to determine different parameter values from smart city data, all parameters are kept for further analysis regardless of the sensitivity test results. Next, the methods of using smart city data to determine values of selected parameters are discussed. For parameters that cannot be determined directly from the traffic data, a calibration method using ANN is adopted.

Chapter 4, Chapter 5, and Chapter 6 present the key analysis results obtained by following the methodology proposed in Chapter 3. Chapter 4 discusses the results of the parameter determination based on the smart city data. A summary section is included to summarize

main findings from this chapter. Chapter 5 discusses the results of the model calibration and model evaluation. The objection function values, measuring speed and travel time errors, are compared between different calibrations to evaluate the performance of each parameter set. Chapter 6 summaries the proposed model calibration process based on the calibration methodologies and results discussed in the previous sections.

Chapter 7 summarizes the thesis. Key findings and a summary of contributions of the thesis are presented. In addition, areas for future work are also discussed.

2 Literature Review

This literature review is comprised of two main sections. The first part discusses the selection of microscopic traffic simulation parameters. Secondly, methods used to determine the microscopic traffic simulation parameter values are reviewed.

2.1 Parameter Selection

Regardless of traffic simulation software package being used, a microscopic traffic simulation model generally includes a set of model parameters that can be changed by users to control simulation output. For example, in VISSIM, parameters can be generally divided into two categories: (a) driver behaviour parameters and (b) vehicle performance parameters (Kim et al., 2005). Driver behaviour parameters define drivers' behaviour when they perform car following and lane changing maneuvers. Vehicle performance parameters define a vehicle's speed and acceleration on the road. Dozens of parameters in VISSIM can be changed in the calibration process to match observed traffic data. While these parameters give modellers many degrees of freedom when modelling, it also raises the question of which parameters are most important for matching observed traffic data.

To answer this question, Miller (2009) proposes a parameter selection process for VISSIM modelling. The first step of the process is to eliminate certain parameters immediately based on engineering judgment. The modeller needs to make decisions based on their priori knowledge about which parameters will not meaningfully impact the simulation performance and which parameters are not applicable for the study scope. For example, the Wiedemann 74 car following model is often used for arterial traffic, while the Wiedemann 99 car following model is often used for freeway traffic. If the model does not include one of these facility types, then only the parameters for the relevant car following model require calibration. Dowling et al. (2004) also suggests following the software documentation to determine the set of parameters that affect simulation performance, depending on the specific car-following and lane-changing logic implemented in the software.

After the first step, if it is unclear whether a parameter should be eliminated, it is recommended to conduct a sensitivity analysis to quickly determine if the parameter in question could potentially affect the measures of effectiveness of the model. Jie et al. (2011) and Lu et al. (2016) conduct sensitivity analysis by changing each parameter value by a definite amount while other parameters were set to default values. The simulation result is then compared with the result received from the default parameter set. Inconsequential parameters were excluded for further consideration in subsequent model calibration.

Most research related to VISSIM modelling follows a similar process. Park and Qi (2005) select the modelling parameters by firstly conducting a trial-and-error test for each parameter to see if the parameter affects the simulation results. Rather than simply changing each parameter value by a definite amount, they utilize the analysis of variance (ANOVA) to identify which parameters are more critical to the simulation result. The values of each parameter were made discrete, separated into several groups, and a one-way ANOVA procedure was then used to test the null hypothesis that the means for two or more groups were equal. If the results were sensitive to the parameter, the means for different groups should be statistically different.

By utilizing these parameter selection techniques, some parameters are often selected by researchers for calibration, while other parameters were shown to be inconsequential to the modelling results. Table 2 lists the parameters that are often selected to calibrate in different studies.

Table 2 Summary of Important Traffic Simulation Modelling Parameters from Literature

Parameter	Reference
Desired speed	Park and Schneeberger (2003), Park and Qi (2005), Park et al. (2006), Miller (2009), Jie et al. (2011), Lu et al. (2016)
Desired acceleration/deceleration	Jie et al. (2011), Lu et al. (2016)
Number of interaction objects	Park and Schneeberger (2003), Park and Qi (2005), Kim et al. (2005), Park et al. (2006), Miller (2009)
Look ahead distance	Park and Schneeberger (2003), Kim et al. (2005), Miller (2009)
Average standstill distance	Park and Schneeberger (2003), Park and Qi (2005), Kim et al. (2005), Park et al. (2006), Yu et al. (2006), Miller (2009), Lu et al. (2016), Lidbe et al. (2017)
Additive part of desired safety distance	Park and Qi (2005), Kim et al. (2005), Park et al. (2006), Yu et al. (2006), Miller (2009), Lu et al. (2016), Lidbe et al. (2017)
Multiple part of desired safety distance	Park and Qi (2005), Kim et al. (2005), Park et al. (2006), Yu et al. (2006), Miller (2009), Lu et al. (2016), Lidbe et al. (2017)
Maximum deceleration	Park et al. (2006), Yu et al. (2006), Miller (2009), Lidbe et al. (2017)
-1 m/s^2 per distance (Reduction rate)	Park et al. (2006), Yu et al. (2006), Miller (2009), Lidbe et al. (2017)
Accepted deceleration	Park et al. (2006), Yu et al. (2006), Miller (2009), Lidbe et al. (2017)
Waiting Time Before Diffusion	Park and Schneeberger (2003), Park et al. (2006), Yu et al. (2006), Miller (2009), Lidbe et al. (2017)
Minimum Headway for lane changing	Park and Schneeberger (2003), Park et al. (2006), Yu et al. (2006), Miller (2009), Lidbe et al. (2017)
Safety distance reduction factor	Miller (2009), Lidbe et al. (2017)

2.2 Parameter Determination with Field Data

After the important parameters are selected for calibration, the next step is to determine their values, since the default parameter values provided by VISSIM could lead to discrepancies between the simulated results and observed field data, and ultimately to inaccurate results (Rrecaj, 2015). In general, there are two approaches to determine parameter values: (a) direct estimation and (b) calibration. Direct estimation obtains the parameter values directly from observed field data. Calibration refers to selecting a combination of parameter values that produce a simulation result that best matches real-world traffic observations (often aggregates, such as travel times). These two approaches can also be used together to improve the accuracy of the calibration.

2.2.1 Direct Estimation

The most reliable way to determine parameter values in VISSIM is to directly estimate them from field data. For example, the desired speed, desired acceleration, and average standstill distance can be estimated from the real-world traffic. However, due to limitations in data collection technology in the past, there is often a lack of traffic data available for modellers to use. In most cases, only hourly volume and aggregated turning movement counts are collected from traditional sensors (e.g., loop detectors). Therefore, it can be hard for modellers to determine these parameter values in VISSIM. To make models better represent real-world traffic, different technologies can be leveraged to collect more traffic data that can capture detailed vehicle characteristics and driving behaviours.

2.2.1.1 On-board Data Collection

Survey-based approaches usually rely on instrumented vehicles (IVs), equipped with GPS, radar, cameras or other sensors. The behaviour of vehicle itself, as well as the surrounding vehicles, are recorded for analysis.

Bifulco et al. (2014) present a large survey based on the naturalistic (on-the-road) observation of driving behaviour with a detailed data collection methodology. The first step of the survey is to define participants. A systematic method is used to recruit participants based on their gender, age, and education level. The distribution of those attributes should be comparable to the real-world population data. In addition, drivers should be divided into different clusters based on their driving behaviour by answering a pre-selection questionnaire.

The second step of Bifulco et al. (2014)'s methodology is to conduct the experiment. The testing route is divided into different sections with different road characteristics. Drivers are assigned to different time slots from 8:30 a.m. to 6:30 p.m. After the experiment is finished, each driver is asked to fill a post-driving questionnaire in order to ascertain in what way the driver's mood was influenced by the experiment. The trajectories of each driver and surrounding vehicles are collected by the experiment. Researchers can analyze the trajectory data to determine different trip characteristics and driving behaviour

parameters. Although this method can collect very accurate and detailed driving data, the time and cost of this approach is very high and collected data set is usually small. In addition, because drivers are aware that their behaviours are being recorded, the experimental result might be unrepresentative.

To make it easier to obtain driving behaviour survey data, the second Strategic Highway Research Program (SHRP 2) was initiated to study the role of driving performance and behaviour in road safety. Over 3,000 volunteer passenger vehicle drivers from different sex and age groups participated this naturalistic driving study. Data collected include speed, acceleration, braking, lane position, camera views of forward, rear, and driver's face and hands. These data were used to develop and evaluate safety countermeasures designed to prevent and reduce the risk of traffic incidents (Hallmark et al., 2014). Another alternative is to retrieve data from transport companies. For example, Jie et al. (2011) uses GPS tracking data from more than 6,000 taxis to determine the acceleration and deceleration profiles. However, this data may not be representative of the general public's driving behaviour.

2.2.1.2 Roadside Data Collection

There are many roadside installed technologies that can be used to collect traffic data. For example, the radar speed detectors deployed in school zones and video cameras installed at the intersections both can be used to collect more detailed traffic data. Different from using on-board sensors to record movement of individual vehicles, roadside equipment can track the trajectories of many vehicles at the same time.

The most used roadside equipment for traffic data collection is video cameras. Although video cameras are already deployed in many cities, they are mostly used for the purpose of law enforcement rather than traffic analysis due to limitations on the number of monitored road sections and lack of expertise. Nonetheless, a number of researchers have developed tracking algorithm for vehicles. Video-based traffic analysis consists of four main steps: data collection, preprocessing, processing, and analysis of video data (Ismail, 2010). Jackson et al. (2013) has done a thorough analysis on each of the four steps. For data collection, a mobile flexible camera unit is preferred since surveillance video

cameras are not always available and the quality of the video often does not meet the basic study requirements. The preprocessing includes vibration correction and camera calibration which allows the projection of real-world measurements onto the image camera space. The processing consists of extracting the road user trajectories from the video data. Analysis includes the manipulation and interpretation of trajectories, speeds, and other parameters of interest.

Saunier and Sayed (2006) developed an open-source software, Traffic Intelligence, to track the movement of different objects. During video processing, individual pixels are detected and tracked from frame to frame and recorded as feature trajectories. A moving object will have multiple features on it, which must therefore be grouped. Then, feature trajectories are grouped based on consistent common motion.

Several researchers have applied this software for video processing. St-Aubin et al. (2013) conduct a surrogate safety analysis at a highway ramp to study the effectiveness of a lane-change ban treatment. The trajectory data processed by Traffic Intelligence are used to generate a Time-to-Collision (TTC) measurement between any pair of road users. Lu et al. (2016) proposes a video-based approach to incorporate direct estimations of car-following parameters into the process of VISSIM model calibration. In their study, desired speeds and desired acceleration rates from a stationary position through an intersection are extracted from vehicle trajectories processed by the Traffic Intelligence software.

The video-based approach is also employed in the Next Generation Simulation program (NGSIM), a public-private project between the Federal Highway Administration (FHWA) of USA and several commercial micro-simulation software developers. Many researchers have used shared trajectories from NGSIM to perform traffic analysis. Montanino and Punzo (2013) propose a multistep filtering procedure for NGSIM trajectory data. The filtering procedure consists of the following steps: (1) remove the outliers; (2) cut off the high- and medium-frequency responses in the speed profile; (3) remove the residual unphysical acceleration values and preserve the consistency requirements; and (4) cut off the high- and medium-frequency responses eventually generated from Step 3. Zhong et al. (2016) utilize the data collected on April 13, 2005,

which is a 15-min time frame observation (4:00–4:15 pm) on a stretch of Interstate 80 in San Francisco, California. The Intelligent Driving Model (IDM), a widely used car following model, was calibrated using this dataset.

Other than cameras, Radar and LiDAR are new technologies have been applied for traffic data collection (Hilpert et al., 2018; Xu et al., 2018). Compared to camera, Radar/LiDAR can collect data under adverse weather conditions. More importantly, LiDAR is able to construct 3D imaging of the roadway and identify different road activities. However, the application of Radar and LiDAR is currently limited by their expensive equipment costs.

2.2.2 Model Calibration

In practice, many microscopic simulation models are calibrated on a trial-and-error basis to determine which parameter values minimize the estimation error between simulation results and field observations of certain macroscopic traffic flow measures, such as travel time and flow rate.

Hellinga (1998) summarizes the common issues related to the calibration of traffic simulation models and describes the overall calibration process. The first phase of calibration is to understand the modelling objective, the available data, and the evaluation criteria. This phase is commonly conducted for all calibration methods prior to the commencement of any modelling. The second phase is initial calibration of network coding, link characteristics, driving behaviour and original-destination traffic demands. In state-of-the-art microscopic traffic simulation software, this step can be done easily by using various built-in functionalities. The third phase is to compare the simulation result with field conditions and test against the previously established criteria. If these criteria are met, then the model is considered to be adequately calibrated. The flow chart for each phase is shown in Figure 1.

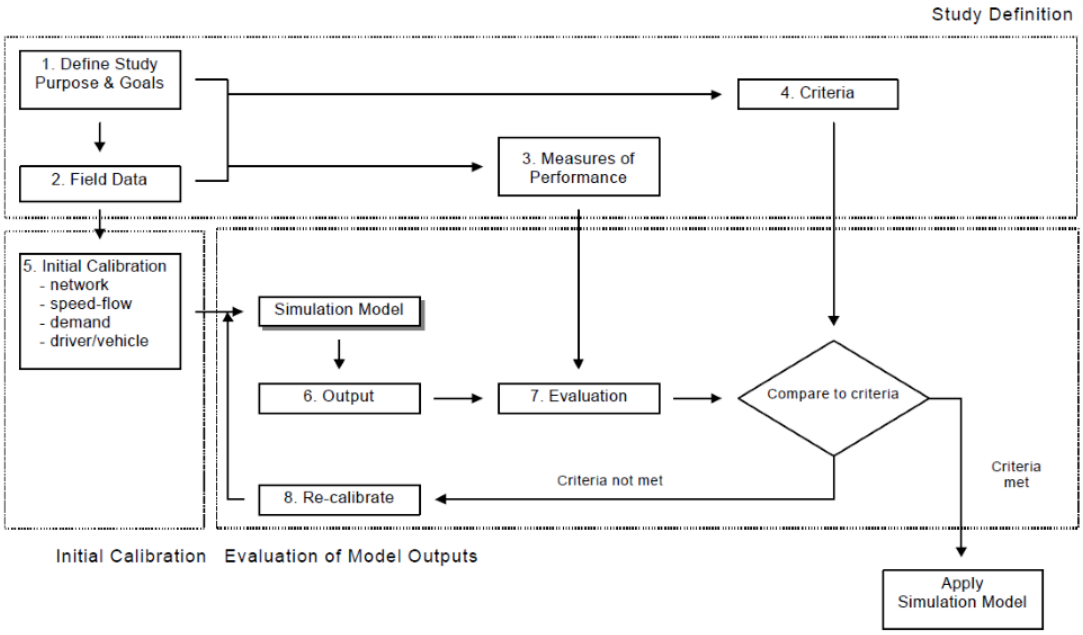

Figure 1 Calibration Process (Hellinga, 1998)

The model calibration framework for microsimulation traffic models proposed by Hellinga (1998) is still applicable today. However, various approaches have been developed to find the "best parameter set".

2.2.2.1 Early Approaches

Several approaches for microsimulation model calibration were proposed before artificial intelligence techniques were developed and popularized.

The principle of manual search entails searching for the model parameters manually on a trial-and-error basis. While using this approach, a modeller changes the value of a selected parameter based on previous knowledge and experience. This approach is commonly used in practice and easy to understand, however, the calibration result is often not robust (Kim, et al., 2005).

Because the relationship between parameters and simulation outputs are complicated by various interactions within the simulation, some researchers tried to use regression models to mathematically relate the influence of parameters on the simulation results. Park and Schneeberger (2003) use the Latin hypercube sampling (LHS) method to

generate hundreds of combinations of selected VISSIM parameters and build a regression model where independent variables are VISSIM parameters and the dependent variable is travel time. Then, the field measured travel time is set as the target value to determine combinations of parameters producing travel time values close to the field measured travel time. Afterwards, travel times are collected from multiple simulation runs with selected parameter sets. The t-test is applied with simulated travel times to confirm the statistic significance between simulated travel times and field data.

The gradient approach changes initial parameters based on the perceived direction of the maximum increase of the objective function. Each parameter is changed in proportion to the magnitude of its slope with the simulation model. The goal is to produce an optimal value of the objective functions (Kim, et al., 2005). However, this method may only capture the local optimal rather than the global optimal value (Ma and Abdulhai, 2002).

2.2.2.2 Genetic Algorithm

Because the calibration of a microscopic simulation model is very complex and stochastic in nature, it is sometimes formulated as an optimization problem and solved by heuristic methods (Ma et al. 2007). A genetic algorithm (GA) is a meta-heuristic optimization technique that uses the concept from evolutionary biology to search for a global minimum. The name “GA” comes from the fact that the algorithms are mimicking evolutionary biology techniques. A GA works by starting with an initial generation of candidate solutions, akin to chromosomes, that are tested against the objective function. Then, subsequent generations are generated from the first generation through selection, crossover, and mutation. Selection means to retain the best-performing parent from one to the next generation. Crossover means to combine the genetic information of two parents to generate new offspring. Mutation is the process to take a parent and mutate certain variables to create a child. This process allows a GA to avoid falling into local minima and helps them to fully explore the solution space.

GAs has been employed in calibrating simulation models in many studies to create the best set of input parameters so that the model produces results similar to reality (Miller, 2009). The list of studies using GAs for model calibration is shown in Table 3.

Table 3 Summary of Studies Using GAs

Authors	Measure of Performance	Fitness Function	Software
Ma and Abdulhai (2002)	Link flow	Global Relative Error	PARAMICS
Yu et al (2003)	Speed	Sum of Squared Error	VISSIM
Kim et al. (2005)	Travel time	Moses' test, Wilcoxon test, and Kolmogorov–Smirnov test	VISSIM
Park and Qi (2005)	Travel time	fitness value	VISSIM
Yu et al. (2006)	Speed, traffic volume	Sum of Squared Error	VISSIM
Ma et al. (2007)	Capacity, Critical Occupancy	GEH (Geoffrey E. Havers)	PARAMICS
Abdalhaq and Baker (2014)	Travel time	Average Error	SUMO
Lidbe et al. (2017)	Traffic volume, Travel times	GEH	VISSIM
Yu and Fan (2017)	Flow, speed	Mean Absolute Normalized Error	VISSIM

Park and Qi (2005) utilize the GA for VISSIM model parameter calibration. Before the calibration starts, critical parameters include simulation resolution, number of observed preceding vehicles, maximum look-ahead distance, average standstill distance, saturation flow rate, minimum headway, minimum gap time, desired speed, and their appropriate ranges are identified from a sensitivity test and feasibility test. Then, the GA starts by generating a number of individuals in the population, each of which represents a feasible set of parameters. The parameter sets are inputted to the simulation model to obtain the simulation result (e.g., travel time). After each run, or generation, the results are measured and assigned a level of fitness. The fitness function shown in Equation 6 is used to compare the relative difference between the simulated results and field measurements:

$$FV = \frac{|TT_{field} - TT_{sim}|}{TT_{field}} \quad (6)$$

where FV is fitness value, TT_{field} is the average travel time (seconds) from the field, and TT_{sim} is the average travel time (seconds) from the simulation.

Based on the level of fitness, the parameter sets with good fitness value are selected as parents to populate the next generation (i.e., next parameter sets). Figure 2 demonstrates

the convergence of fitness value over subsequent generations. The results of the study by Park and Qi (2005) are based on 10 generations with a population size of 20. The crossover rate is 0.8 which means 80% of offspring are made by crossover and the mutation rate is 0.05 which means 5% of parameter sets should be mutated in one generation.

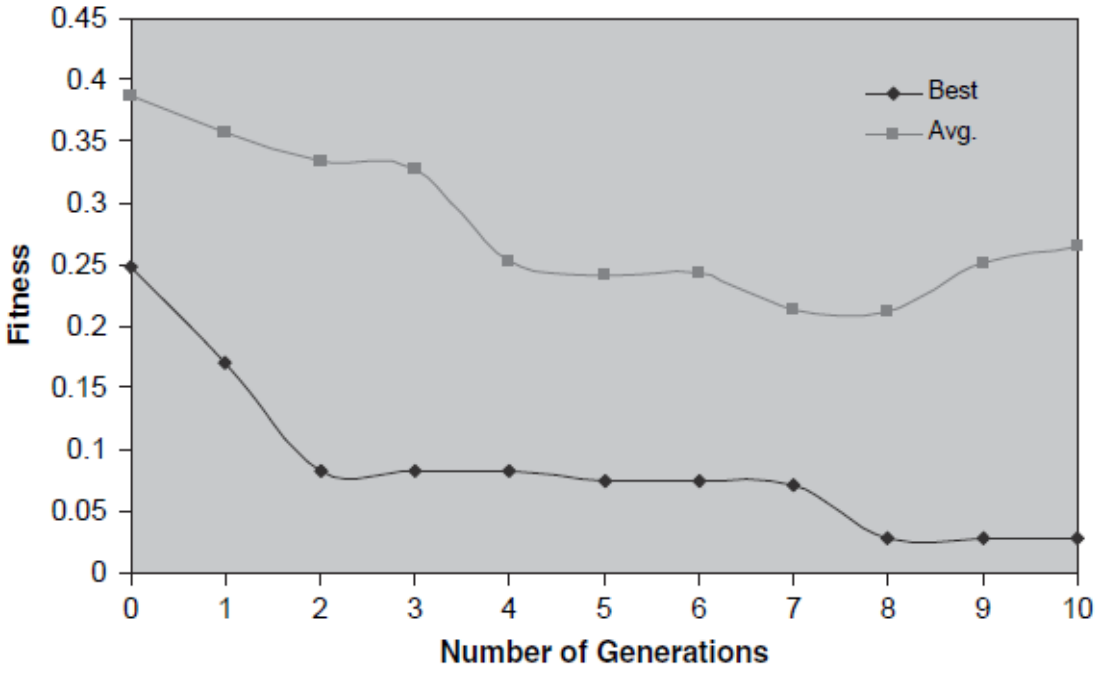

Figure 2 Convergence of GA Fitness Value with Generation (Park and Qi, 2005)

In addition to the average travel time, Park and Qi (2005) also compare the travel time distribution of different parameter sets. Other than the GA-optimized parameter set with best fitness value, they also run simulations with the VISSIM default parameter set and a best-guess parameter set developed on the basis of the engineers' knowledge of local traffic conditions. According to Figure 3, the travel time distribution obtained from default parameter set and the best-guess parameter set are far away from the field travel times collected over 3 days. On the other hand, all three field measurements fall within the simulated distribution of the GA-optimized parameter set.

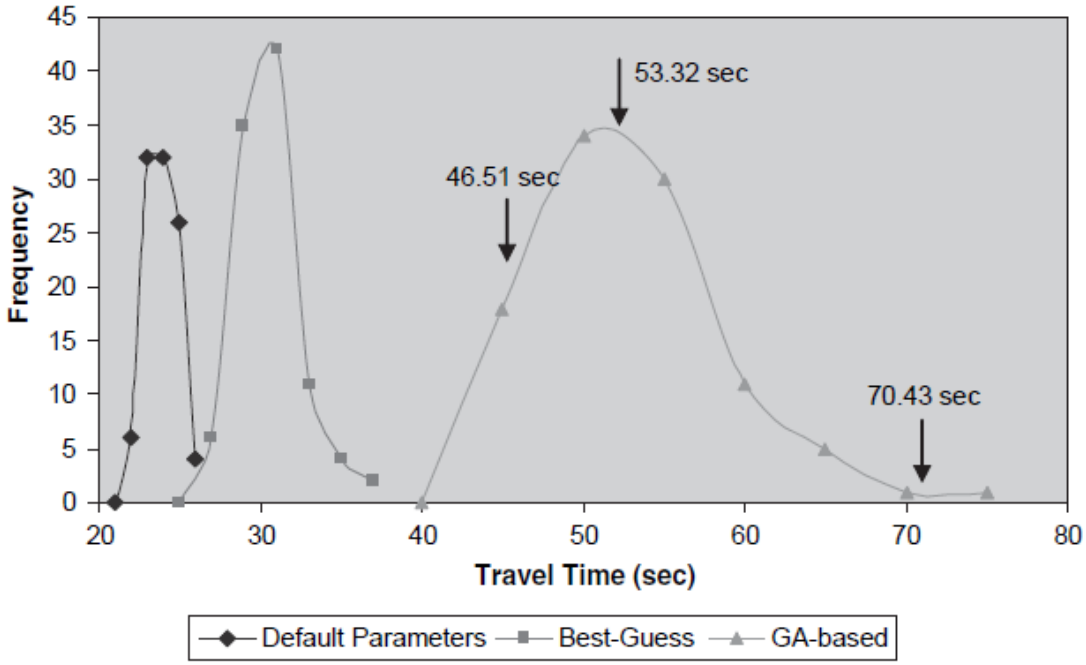

Figure 3 Comparison of Travel Times by Different Parameter Sets (Park and Qi, 2005)

Kim et al. (2005) also utilizes a GA to calibrate a VISSIM model. Different from Park and Qi (2005), they recognize that the best parameter set found by minimizing aggregated performance measures (e.g., average travel time) based objective function may not be valid because this assumption is true only when the distributions for the simulated and observed travel times are identical. Therefore, instead of using average travel time collected from the field, they collect the travel time of individual vehicles from the site. In the GA process, nonparametric statistical tests (e.g. Moses' test, Wilcoxon test, and Kolmogorov Smirnov test) are used to evaluate each candidate solution (i.e., parameter set) based on the mean and dispersion of the individual travel time distribution. After the accepted parameter sets are selected, then the mean absolute error ratio (MAER) shown in Equation 7 is used to measure the difference between simulated result and filed measurement.

$$MAER = \frac{\sum_{i=1}^{n} \left| \frac{s_i - o_i}{o_i} \right|}{n} \quad (7)$$

where S_i is the travel time from simulation model, O_i is observed travel time, and n is number of observations.

Ma and Abdulhai (2002), Yu et al. (2003), Yu et al. (2006), Abdalhaq and Abu Baker (2014), and Lidbe et al. (2017) use a similar methodology with different tuning on the GA parameters (e.g., crossover rate and mutation rate). For example, Abdalhaq and Abu Baker (2014) use the average crossover method which is taking the average value of the parameters from all the parent parameter sets to create the next generations. In addition, Ma et al. (2007) compares the GA with another heuristic algorithm, the simultaneous perturbation stochastic approximation (SPSA). The detail of the SPSA is introduced in the next section.

2.2.2.3 Simultaneous Perturbation Stochastic Approximation

Simultaneous perturbation stochastic approximation (SPSA) is an efficient method for optimizing computationally expensive, ‘‘black-box’’ traffic simulations (Hale et al., 2014). The fundamental idea of SPSA is to search for the optimal point that corresponds to the zero gradient of the objective function. To avoid the solution falling into local optima, the algorithm uses stochastic vector to determine the direction of choosing values to calculate the gradient (Abdalhaq and Abu Baker, 2014). The list of studies using SPSA for model calibration is shown in Table 4.

Table 4 Summary of Studies Using SPSA

Authors	Measure of Performance	Fitness Function	Software
Ma et al. (2007)	Capacity	GEH	PARAMICS
Lee and Ozbay (2009)	Flow, Speed	Mean Square Variation	PARAMICS
Paz et al. (2012)	Vehicle counts, speed	GEH	CORSIM
Abdalhaq and Abu Baker (2014)	Travel time	Average Error	SUMO
Hale et al. (2015)	Speed, Density	Minimize the difference between simulated and field measured outputs	FRESIM

Ma et al. (2007) successfully calibrate the PARAMICS traffic model with the SPSA. In that study, the SPSA is set up in the following way. For the system, an objective function $L(\theta)$ is used to evaluate the fitness between the simulated results and field measurements,

where θ are the parameters selected to be calibrated. The fitness between simulated results and field conditions as measured by GEH statistic is shown in Equation 8:

$$GEH = \sqrt{\frac{2(V_p - V_m)^2}{(V_p + V_m)}} \tag{8}$$

where V_p is the traffic volume (veh/hr) predicted by the model and V_m is the traffic volume (veh/hr) measured in the field.

Assuming that $L(\theta)$ is differentiable over θ, the minimum of $L(\theta)$ can be obtained at a zero gradient:

$$g(\theta) = \left|\frac{\partial L(\theta)}{\partial \theta}\right|_{\theta=\theta^*} = 0 \tag{9}$$

Starting at an initial guess of θ_0, the SPSA method applies a series of stochastic perturbations to the candidate parameter sets to update the best solutions until the approximation of the $g(\theta)$ converges to zero.

Ma et al. (2007) also compares the calibration results of using the SPSA and a GA. As shown in Figure 4, for the SPSA method, the fitness values converge very quickly after few iterations, but they become very oscillatory during the remaining process, which is sensible given the stochastic perturbations involved at each iteration. On the other hand, the GA method has a smoother convergence process and the difference between maximum and minimum fitness values become smaller which means that the GA method reaches a more stable optimal solution.

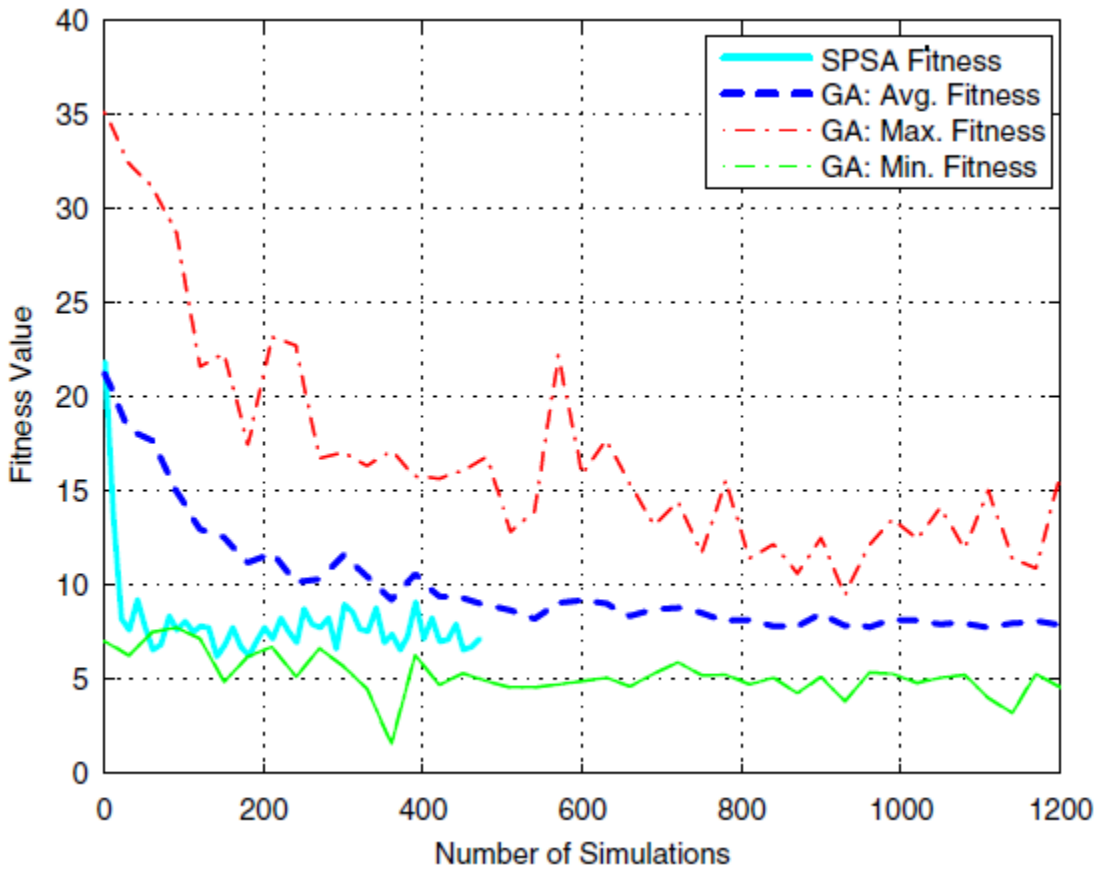

Figure 4 Local Parameter Calibration Convergence Diagram (Ma et al., 2007)

Abdalhaq and Baker (2014) also compare the performance of the SPSA and GA. They conduct calibration by using the SPSA and GA on two different sites. The average fitness result shows that the SPSA performs well on a simple network while the GA performs better on a more complex network. Lee and Ozbay (2009) propose enhanced SPSA (E-SPSA) by combining the Bayesian sampling approach and SPSA. Similar to the study done by Kim et al. (2005) that is introduced in the previous section, the Kolmogorov–Smirnov test is performed to ensure that the distribution of the simulation results represented real traffic conditions.

2.2.2.4 Artificial Neural Network

An Artificial Neural Networks (ANN) is an Artificial Intelligence (AI) technique that emulates the function of the human brain. ANN develop their understandings by finding relationships and patterns in data and learn through experience. An ANN is formed from

artificial neurons connected with coefficients, which constitute the neural structure, and are organized in layers (Agatonovic-Kustrin and Beresford, 2000). The representation of a basic ANN is shown in Figure 5.

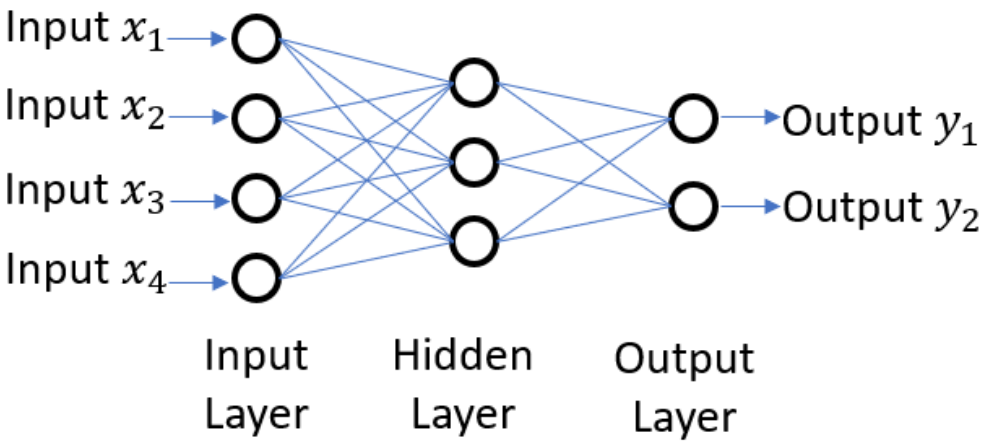

Figure 5 Neural Network Representation

In the ANN, input nodes can take inputs to the neural network. Then, based on the connection weights, the artificial neuron computes the weighted sum of the inputs and modifies the data received through the transfer function in the hidden layer to generate an output signal, which represents the activation of the neuron. The transfer function is used to introduce non-linearity to the network. Common transfer functions used for ANN include Sigmoid Function, Hyperbolic Tangent Function, and Gaussian Function. The representation of this process is shown in Figure 6. Training of the ANN is the process of adjusting the connection weights between neurons to reproduce the input-output results of the training dataset with minimal error (Daguano, 2019).

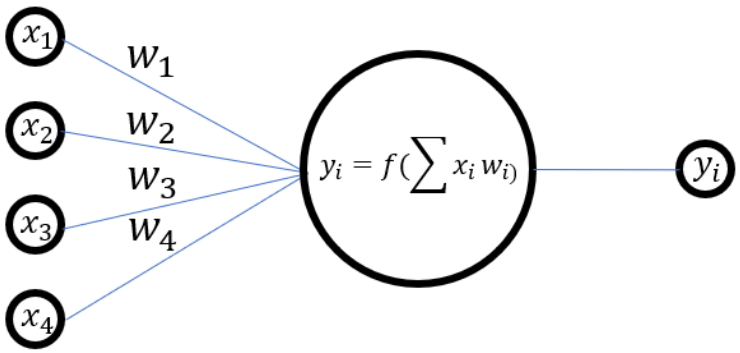

Figure 6 Perceptron Model

Daguano (2019) proposes a methodology to train ANN that can calibrate microscopic traffic simulation models. As for other calibration methods, calibration using ANN also starts with selecting parameters and defining ranges for those parameters. After identifying the parameters to calibrate, a user can collect desired network performance measurements output by running the VISSIM model with different input parameters many times (e.g., 2000 times). Because the purpose of this study is to determine VISSIM parameter values from field measurements, the input/output order of the traffic simulator is reversed when the dataset for ANN training is constructed. In other words, the simulation outputs (network performance measurements) are used as inputs to the ANN, and the simulation inputs (calibration parameters) are the desired outputs of the modeled ANN. After the training of the neural network is completed, the neural network can generate a model that describes the relationship between network performance measurements and VISSIM parameters. Therefore, when aggregate network performance measurements (travel time, speed, etc.) are available, the model can determine the appropriate VISSIM parameter values that will generate similar results from the VISSIM model.

Different from the other calibration methods, ANN are designed to establish the relationship between microscopic traffic simulation parameters and outputs, rather than find the “best” parameter set by trial-and-error. ANN users can obtain parameter sets for different field measurement calibration targets, while other calibration methods would

require users to rerun all of their simulations again if the calibration targets are changed. In addition, the formulation of the genetic algorithm is relatively complicated: the fitness function, population size, rate of mutation and crossover, and selection criteria for the new population need to be carefully selected (Yang, 2014). Therefore, an ANN is selected for calibration of unobservable parameters in this research.

2.3 Gaps

In summary, numerous studies have been conducted on the calibration of microscopic traffic simulation parameters. However, there are a few gaps between existing literature and the emerging real-world application needs and opportunities.

The techniques used for the calibration of microsimulation parameters rely heavily on engineering judgement and optimization-based calibration algorithms. Although some studies generate results that are close to the pre-defined measures of performance, the wider representativeness of the calibrated models outside of these measures remains questionable. The fundamental reason for relying on optimization-based calibration is the historic lack of data. In most existing studies, the type of traffic data collected are very limited and they are often used as a measure of performance for validation of the model. However, the development of smart city technology provides an opportunity to modellers. It is likely that in the foreseeable future, it will be easier and cheaper to collect various types of traffic data for transportation studies. Hence, there is an opportunity to improve the accuracy of the microsimulation models since many parameters can be directly determined from the field data and multiple measures of performance can be used for model validation.

Few studies have utilized field data to calibrate traffic modelling parameters and they are not directly compatible with VISSIM model calibration. For instance, Zhong et al. (2016) utilized field traffic data to calibrate the IDM car following model but it is not used in the VISSIM model. The other example is Lu et al. (2016), which however focused only on calibrating the VISSIM car following model parameters. This research focuses on not only the calibration of VISSIM traffic model, but also developing methodologies of determining as many parameter values as possible. At the same time, standardized

approaches need to be developed to ensure appropriate and consistent use of these data for model development. The best practices are introduced, and some recommendations are made while introducing the proposed methodologies. The following parts of this thesis utilize smart city data to improve the representativeness of microscopic traffic simulation models and develop standardized approaches to use these data for calibration purposes.

3 Data and Methodology

3.1 Field Data

This study uses the Next Generation Simulation (NGSIM) dataset to test the usage of smart city data in microscopic traffic simulation calibration. The NGSIM program was initiated by the Federal Highway Administration (FHWA) with a primary focus on collecting supporting data and documentation for microscopic traffic modelling. The data was collected by synchronized video cameras, as would be the case in many smarty city implementations. Hence, the processed data contains similar data compared to the outputs of other smart city data sources.

An arterial section on Peachtree Street in Atlanta, GA, was selected from the NGSIM database to model the urban traffic environment. This location was selected because its road geometry (e.g., two-lane highway, signalized intersection and stop-controlled intersection) is comparable to many North America urban roads. Figure 7 shows the schematic of the study area.

The traffic data were collected from 4:00 p.m. to 4:15 p.m on November 9th, 2006. The posted speed is 35 mph (56 km/h). This arterial section is approximately 640 m in length and includes five intersections—four signalized intersections and one stop–controlled intersection. The raw video data were processed by NGVIDEO, a customized software that can convert video to vehicle trajectory data. The vehicle information is updated at one-tenth of second intervals to capture the continuous movement of the vehicle. The vehicle data collected in 15 minutes consists of 873,887 observations of 1,545 individual vehicles. Information related to vehicle movements can be determined from the trajectory data. Table 5 lists the data types that are available for developing a microscopic traffic simulation model.

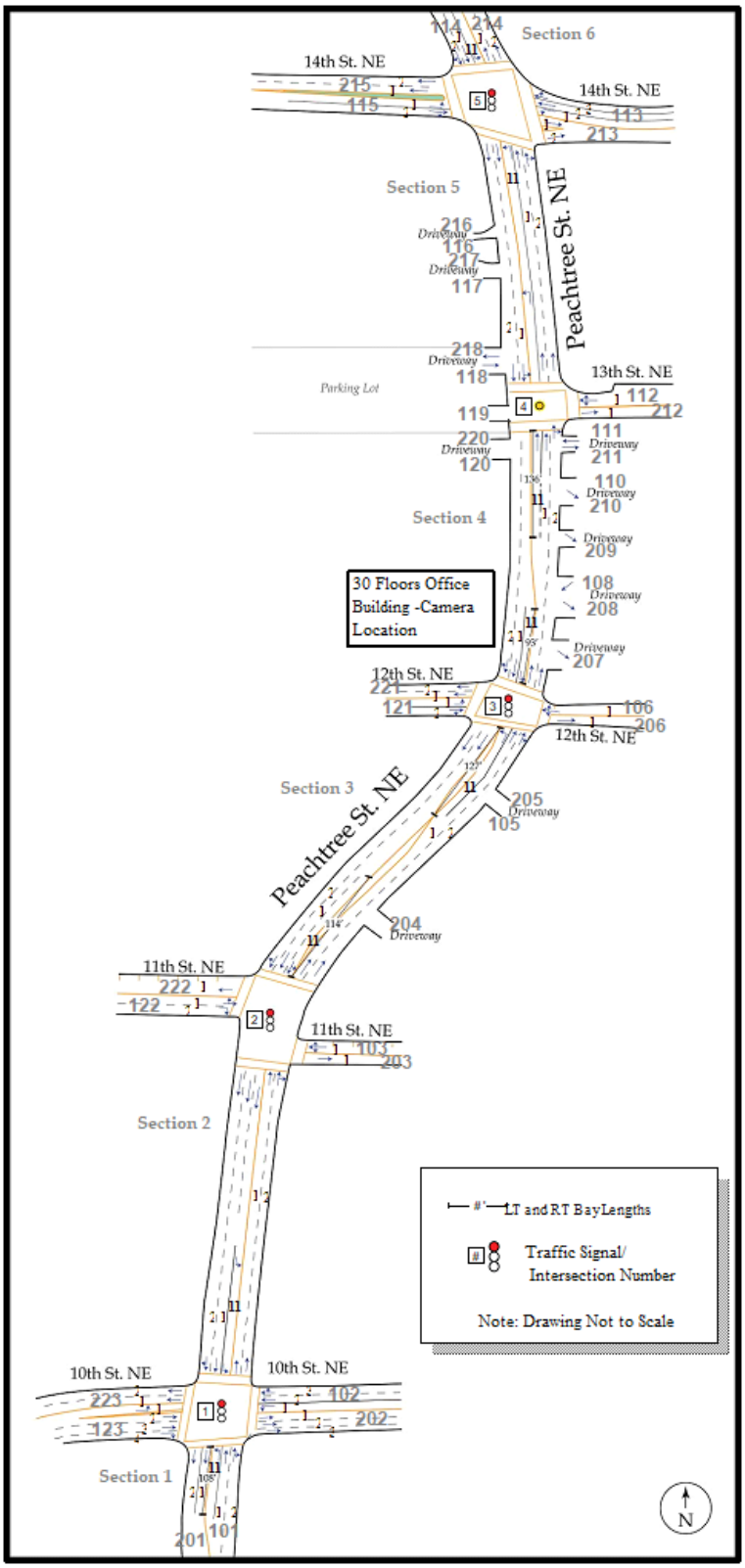

Figure 7 Peachtree St from 10th St to 14th St (Cambridge Systematics Inc., 2007)

Table 5 NGSIM Data Description

Attributes	Description
Vehicle ID	Vehicle identification number
Vehicle Speed	Instantaneous velocity of vehicle
Vehicle Acceleration	Instantaneous acceleration of vehicle
Vehicle Class	Vehicle type: motorcycle, auto, truck
Vehicle Length	Length of vehicle
Lane ID	Current lane position of vehicle
Origin	Origin zones of the vehicle
Destination	Destination zones of the vehicle
Direction	Moving direction of the vehicle: EB, NB, WB, SB
Intersection	Intersection in which the vehicle is traveling
Movement	Movement of the vehicle: through, left-turn, right-turn
Preceding Vehicle	Vehicle ID of the lead vehicle in the same lane.
Following Vehicle	Vehicle ID of the vehicle following the subject vehicle in the same lane
Space Headway	Spacing provides the distance between the front-center of a vehicle to the front-center of the preceding vehicle
Time Headway	Time Headway provides the time to travel from the front-center of a vehicle (at the speed of the vehicle) to the front-center of the preceding vehicle

Among these attributes, vehicle speed, vehicle acceleration, vehicle's space headway and time headway, and lane position are most important for VISSIM parameter analysis. The distributions of these attributes are shown in Figure 8 to Figure 11.

There are some errors in the database. Some records have zero space and time headways (measured from their preceding vehicles), which implies that one vehicle is on the top of another one. Also, time headways at a speed of zero are removed because they are shown as either 0 or 9999.99 in the database.

Figure 8 demonstrates the distribution of instantaneous speed of each vehicle record. The unit of speed is ft/s in the original database. All units in this study are converted to the

metric system. Most vehicles have speeds less than 60 km/h.

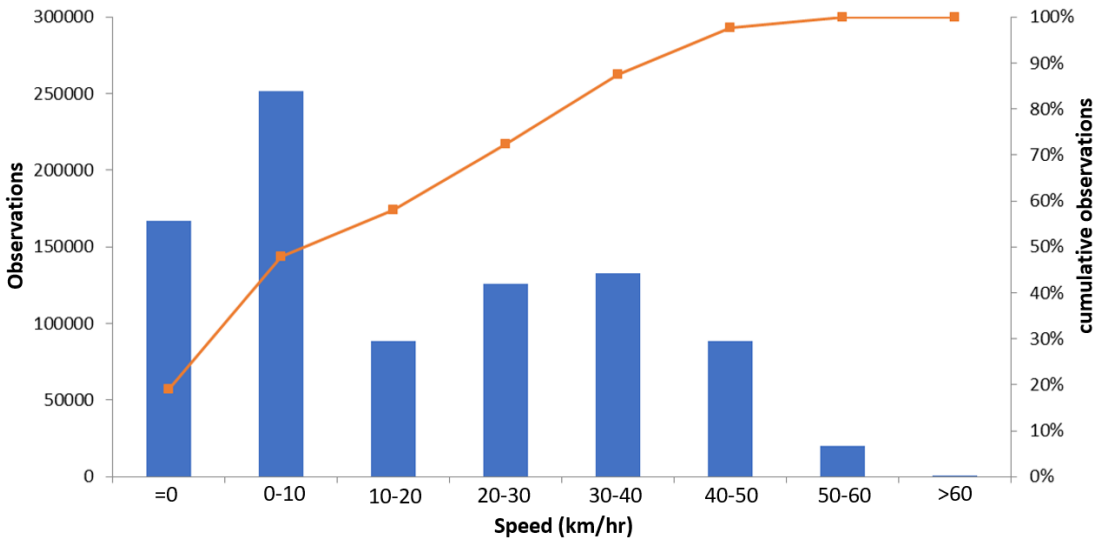

Figure 8 Vehicle Speed Distribution

Figure 9 demonstrates the distribution of vehicle acceleration. The acceleration is numerically derived from the tracked vehicle positions (Thiemann et al., 2008). The unit of acceleration was ft/s^2 in the original database. The most notable feature of this figure is that about 70% of acceleration observations are within -1 to 0 m/s^2 (deceleration). This is because more than 50% of all acceleration observations have an acceleration of zero. Part of the reason is that about 20% of vehicle records are in a standstill position. However, other than records with zero acceleration, a large portion of observations are within -1 to 1 m/s^2. Because human drivers are not perfect, it is very common for them to accelerate or decelerate a little bit when they want to maintain their speed.

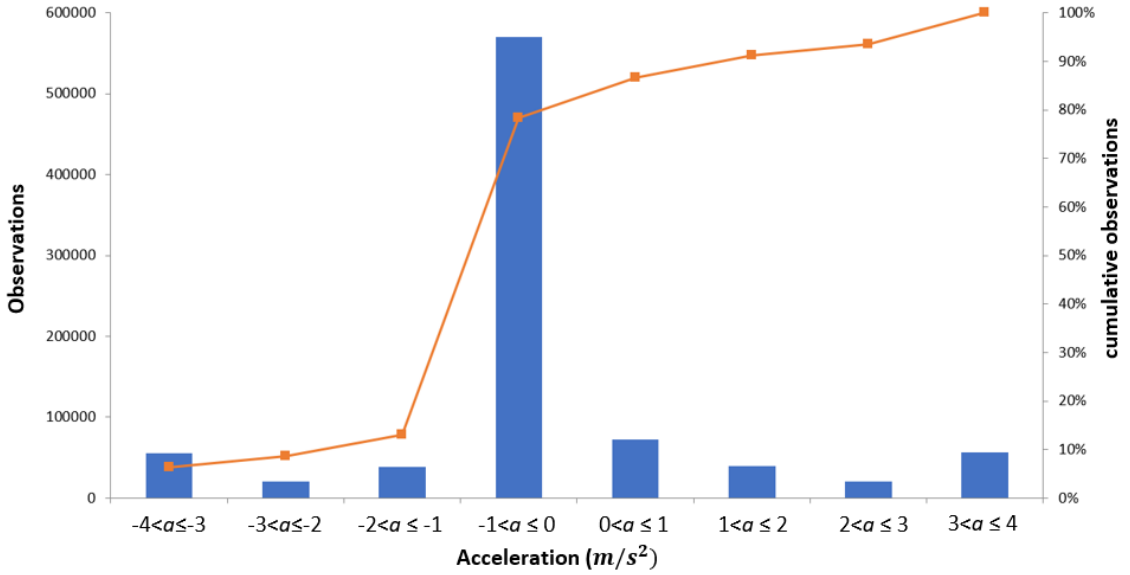

Figure 9 Vehicle Acceleration Distribution

Figure 10 shows the distribution of the space headway. The lowest space headway is around 5m since it includes the car length of the leading vehicle. More than 60% of space headway observations are within 5m to 10m which implies this range is the most common space headway between two vehicles.

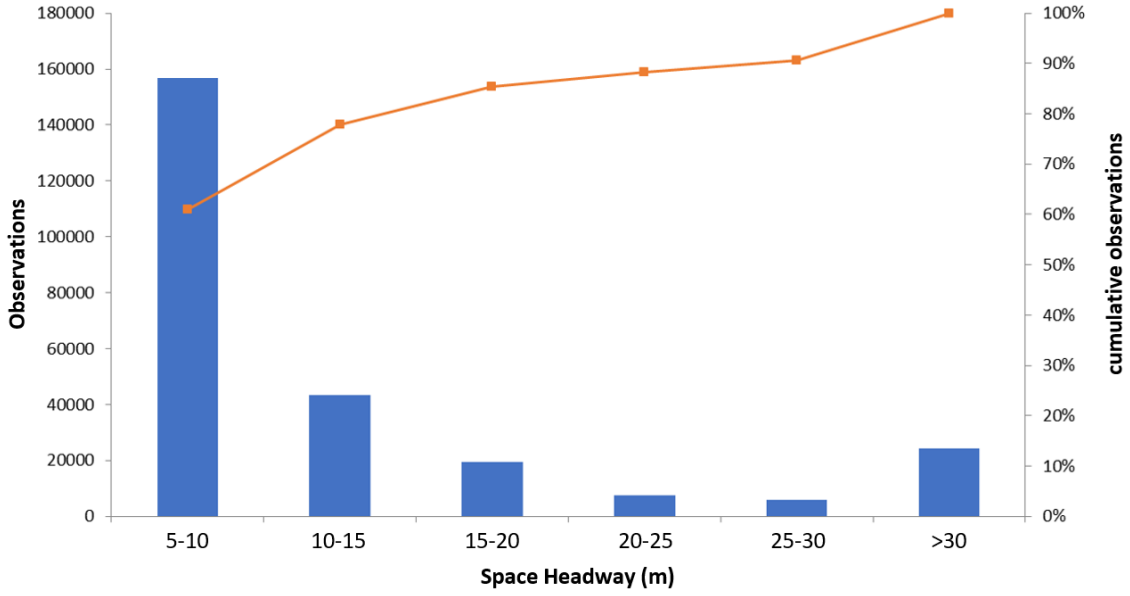

Figure 10 Space Headway Distribution

Figure 11 shows the distribution of the time headway. About 40% of time headway observations are within 0s to 6s which implies this range is the most common time headway between two vehicles.

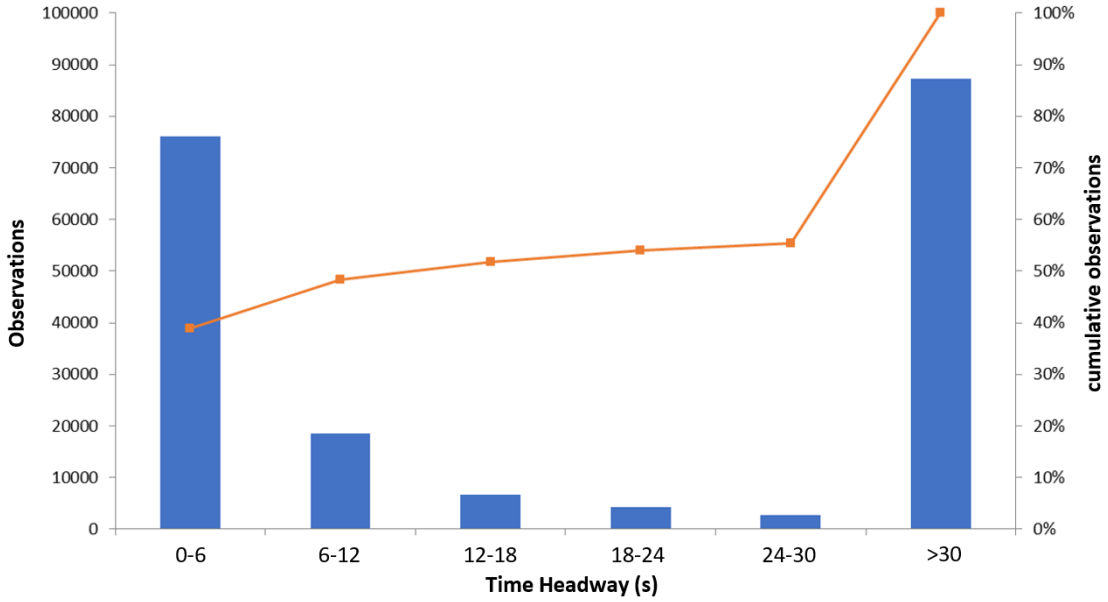

Figure 11 Time Headway Distribution

3.2 Proposed Methodology

The methodology of using smart city data to develop a microscopic simulation model includes six main steps: network building, parameter selection, sensitivity analysis, parameter determination from direct observation, parameter calibration for unobserved parameters, and model evaluation. The details of each step are discussed individually in the following subsections.

3.2.1 Network Building

The microscopic traffic simulation model was created in PTV VISSIM 20 (PTV Group, 2020). The road section shown in Figure 7 was coded in VISSIM. A screenshot of VISSIM road network layout is shown in Figure 12.

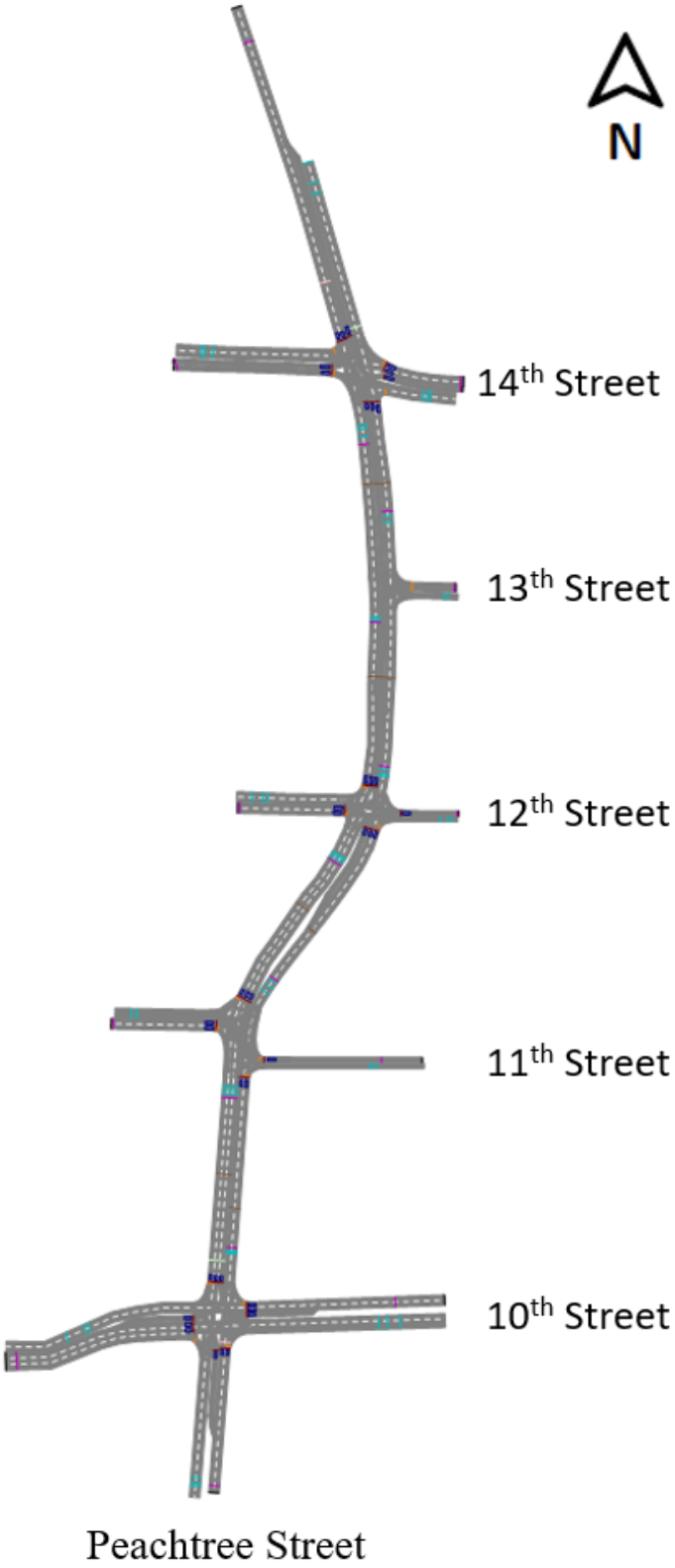

Figure 12 VISSIM Road Network Layout

3.2.1.1 Road Geometry

The road geometry of a VISSIM model is usually determined from the built-in map services in VISSIM or Google maps. However, due to vast changes in the study area since 2006, the geometry of the roadway for this study was constructed based on the aerial photo and CAD diagram provided by the NGSIM dataset, reflecting the roadway length and lane configuration in 2006. A comparison between 2006 and the existing intersection condition at Peachtree Street and 11th street is shown in Figure 13.

Figure 13 Intersection of Peachtree Street & 11th Street (2006 and 2021)

3.2.1.2 Vehicle Volume and Turning Ratio

The vehicle volume inputs and vehicle routings are usually based on turning movements counts (TMC) provided by the municipalities or data collection companies. In this study, they were determined from the NGSIM Peachtree Street (Atlanta) Data Analysis Summary Report (Cambridge Systematics Inc. 2007), which summarizes the number of vehicles and vehicle turning directions observed from the video data for each intersection.

3.2.1.3 Signal Control

Signal timing plans of the study intersections in 2006 were provided by the Georgia Department of Transportation. The signal timing plans were used to program signal controllers in VISSIM through the Ring Barrier Control (RBC) tool.

3.2.1.4 Collection of Simulation Data

Data collection points and vehicle travel times segments were set up in the VISSIM model to collect desired model outputs such as vehicle speeds and travel times. In practice, the modelling outputs are compared with the field measurements to calibrate the VISSIM model. Smart city data can provide different types of field measurements collected from different places in the network to improve modelling accuracy. Different metrics are used for different modelling purposes. For example, travel time is usually used when the purpose of modelling is to evaluate operational performance, but it may not be a valid measure when the modelling purpose is related to safety or emissions. In this study, northbound and southbound average travel times and the average speed at the midpoints of each section are used to evaluate the model performance. Average travel times and average speeds are arithmetic averages measured over the 15 minutes study period. The average travel times are computed for the entire corridor length because most recorded vehicles are travelling along the Peachtree Street corridor. Figure 14 illustrates the approximate data collection locations placed in the VISSIM model.

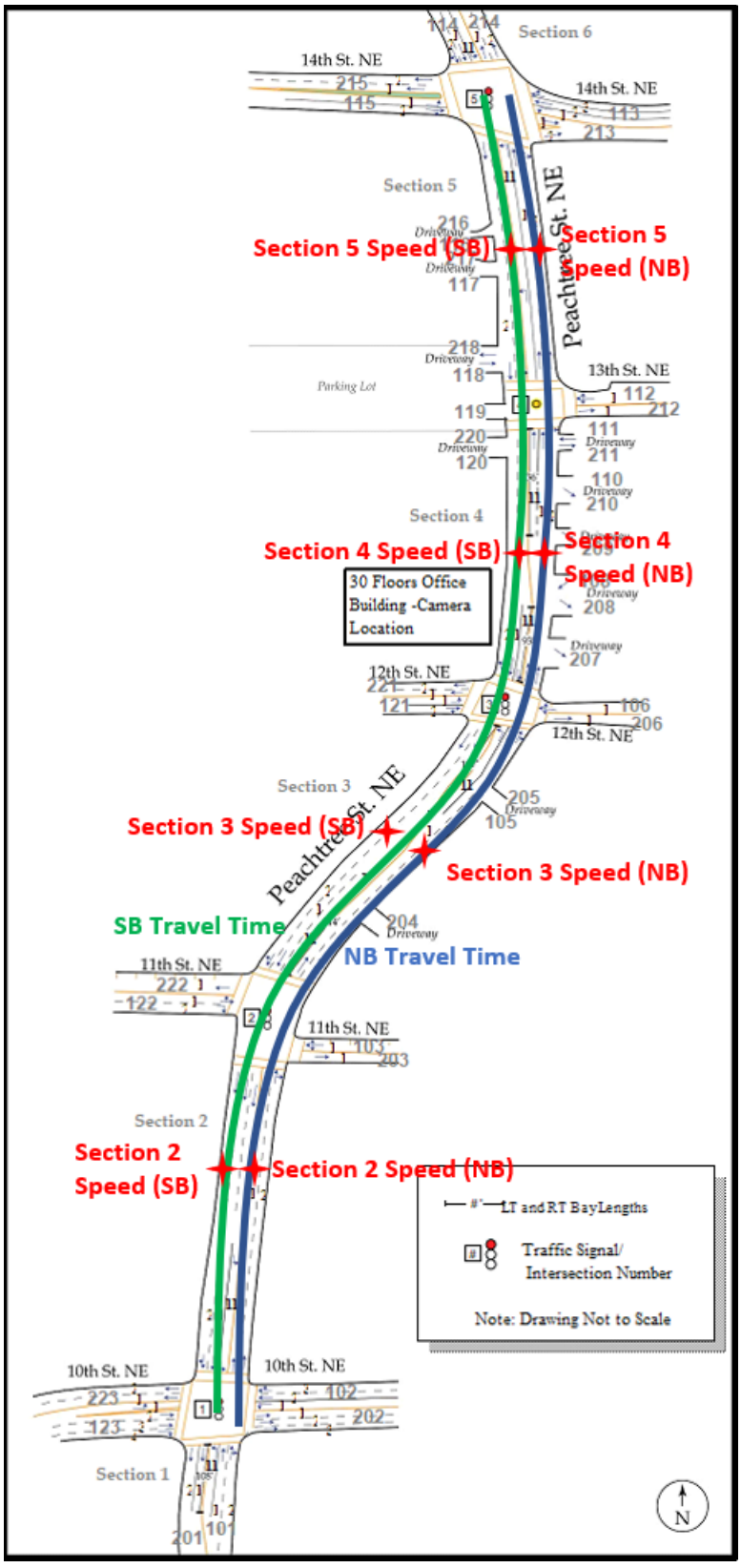

Figure 14 Data Collection Location

3.2.2 **Parameter Selection**

Building a VISSIM model usually consists of two parts as shown in Figure 15. The first part involves building the traffic network by drawing road geometry, inputting traffic volumes, routing vehicles, and designing intersection controls discussed in the previous section. The second part of building a VISSIM model involves defining parameters that are related to vehicle performance and driving behaviour models. The parameters highlighted with red represent the parameters that cannot be extracted from the smart city dataset (i.e., unobserved parameters).

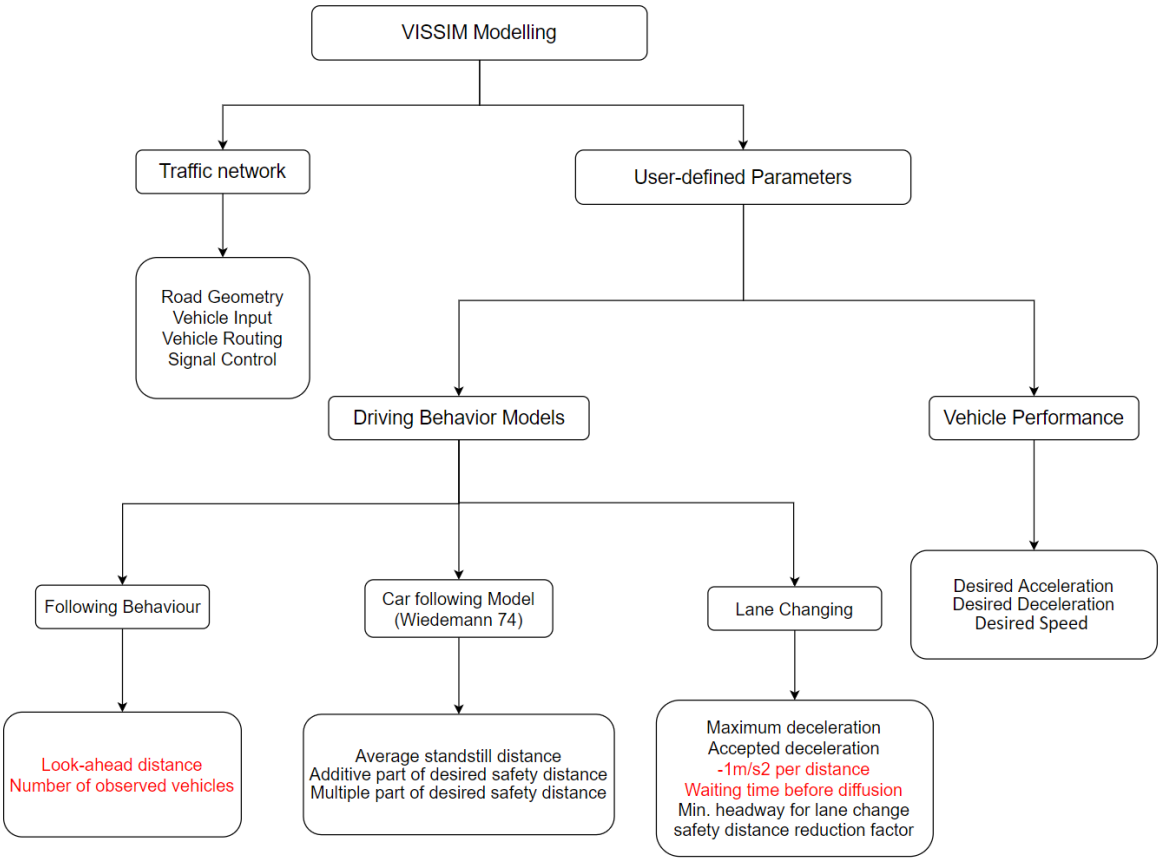

Figure 15 VISSIM Overview

This study focuses on the VISSIM parameters selected for model calibration in previous studies as discussed in the literature review. The selected parameters mainly focus on car following behaviour and lane change behaviour. Traffic network building components

are not considered in this study since they are already well developed in the current practice. The list of selected parameters is shown in Table 6.

Table 6 Parameter Selection

Category	Parameter
Vehicle Performance	Desired speed
	Desired acceleration/deceleration
Following behavior	Look ahead distance (min and max)
	Number of interaction objects
Car following model	Average standstill distance
	Additive part of desired safety distance
	Multiple part of desired safety distance
Lane change	Maximum deceleration (Own and trailing)
	-1 m/s^2 per distance (Own and trailing)
	Accepted deceleration (Own and trailing)
	Waiting Time Before Diffusion
	Minimum Headway for lane changing
	Safety distance reduction factor

3.2.3 Sensitivity Analysis

Due to the complexity of microscopic traffic simulation models, the effect of each parameter to the modelling result is unpredictable for different traffic networks. Therefore, sensitivity analysis is required to identify key parameters and eliminate parameters that have less effect on model results.

One-way ANOVA tests are performed to test the sensitivity of simulation results to the different parameters values. In each test, the selected parameter is changed to a different value to generate multiple modelling results and the one-way ANOVA can test the null hypothesis that the means for two or more groups are equal. If the studied parameter is sensitive to the result, the means for different groups should be statistically different (Park and Qi, 2005). The statistical significance is evaluated by the *p-values*. In this study, parameters with *p*-value less than 0.05 are considered statistically significant. Northbound average travel time and the northbound average speed at road section 3 were selected for the sensitivity analysis to verify the impact of changing parameter values on both travel time and speed. The data collection locations may impact the analysis results.

For example, modellers could get very different results regarding the importance of a parameter from speed data obtained from a mid-block compared to speed data obtained at a roundabout.

Each parameter was tested for three scenarios within its range that was determined from the previous studies and each scenario runs 30 times with different random seeds. Table 7 summarizes the range of each parameter based on previous studies (Park and Qi 2005; Park et al. 2006; Miller 2009). Table 8 summarizes the ANOVA test results for selected VISSIM parameters.

Table 7 VISSIM Parameter Range

Parameters	Range
Desired Speed Distribution (km/h)	±1.0-15.0
Look-ahead distance min (m)	0 to100
Look-ahead distance max (m)	150 to 250
Number of interaction objects	2 to 8
Average standstill distance (m)	1 to 5
Additive part of desired safety distance	1 to 5
Multiple part of desired safety distance	1 to 6
Maximum Deceleration (own) (m/s^2)	-5 to -1
Maximum Deceleration (trailing) (m/s^2)	-5 to -1
-1 m/s^2 per distance (own) (m)	50 to 200
-1 m/s^2 per distance (trailing) (m)	50 to 200
Accepted Deceleration (own) (m/s^2)	-1.5 to -0.1
Accepted Deceleration (trailing) (m/s^2)	-1.5 to -0.1
Min. headway (front/rear) (m)	0.5 to 7
Safety distance reduction factor	0.3 to 0.9
Waiting time before diffusion (s)	40 to 80

Table 8 ANOVA Results

Category	Parameters	*p*-value	
		NB Travel Time	Section 3 Speed (NB)
Vehicle Performance	Desired Speed Distribution	0.0000	0.0000
	Desired Acceleration	0.0000	0.0000
	Desired Deceleration	0.3519	0.0000
Following Behaviour	Look-ahead distance min (m)	0.9872	0.7360
	Look-ahead distance max (m)	0.0019	0.0001
	Number of interaction objects	0.9098	0.0428
Car following model	Average standstill distance (m)	0.0011	0.0000
	Additive part of desired safety distance	0.0000	0.0000
	Multiple part of desired safety distance	0.0000	0.0000
Lane change	Maximum Deceleration (own) (m/s^2)	0.9994	0.9991
	Maximum Deceleration (trailing) (m/s^2)	0.9264	0.8508
	-1 m/s^2 per distance (own) (m)	1.0000	1.0000
	-1 m/s^2 per distance (trailing) (m)	1.0000	1.0000
	Accepted Deceleration (own) (m/s^2)	1.0000	1.0000
	Accepted Deceleration (trailing) (m/s^2)	1.0000	1.0000
	Min. headway (front/rear) (m)	0.4397	0.0238
	Safety distance reduction factor	0.9302	0.7778
	Waiting time before diffusion (s)	0.9995	0.9995

Based on the results shown in Table 8, In the vehicle performance parameters, desired speed distribution and desired acceleration have a statistically significant effect on northbound travel time and all three vehicle performance parameters have a statistically significant effect on northbound speed at section 3. These three parameters are important since they directly impact a vehicle's speed and acceleration in the model.

In the following behaviour parameters, the minimum look-ahead distance does not have a statistically significant effect on travel time and speed since it is not usually considered for longitudinal movement of the vehicle. The maximum look-ahead distance is statistically significant and important for travel time and speed of vehicles since it decides how vehicles react to the preceding vehicles. The number of interaction objects is only critical to vehicle point speed (statistically significant) because it could affect the vehicle decision at some section of the road (e.g., decelerate when too many vehicles are around) but it has minor impact on larger scale travel times (insufficient evidence to reject the null hypothesis).

All car following model parameters have a statistically significant effect on vehicle travel times and vehicle speeds. Theses parameters decide the space headway a following vehicle should keep and when the following vehicle should accelerate or decelerate.

Almost all lane changing parameters do not have much impact on vehicle travel times and vehicle speeds since these parameters are designed to model lane change behaviours.

In practice, parameters that are not statistically significant can be excluded for further calibration consideration. However, for the purpose of this study, all selected parameters are kept in order to explore the connections between smart city data and these parameters.

3.2.4 Model Evaluation

An objective function is used to measure the difference between field observations and simulation results. In this study, the Mean Absolute Normalized Error (MANE) function (Yu and Fan, 2017) is used because it is an appropriate objective function form for problems with multiple types of performance measurements (e.g., travel time and speed). The MANE function is provided by Equation 10.

$$Minimize\ MANE(t,v) = \frac{1}{J}\sum_{j=1}^{J}\frac{|t_{obs,j}-t_{sim,j}|}{t_{obs,j}} + \frac{1}{K}\sum_{k=1}^{K}\frac{|v_{obs,k}-v_{sim,k}|}{v_{obs,k}} \quad (10)$$

where

$t_{obs,j}$ is observed average travel time for a given section j.

$t_{sim,j}$ is simulated travel time for a given section j.

$v_{obs,k}$ is observed average speed at a given location k.

$v_{sim,k}$ is simulated average speed at a given location k.

J is total number of travel time collection sections.

K is total number of speed collection locations.

3.2.5 Parameter Determination Using Smart City Data

The key difference of this research compared to previous studies is the focus on determining model parameters directly from traffic data. For this study, the NGSIM dataset is the primary data source used to convert vehicle trajectories to VISSIM parameters. Smart city traffic data from other data sources should also be able to follow the proposed procedures. The proposed procedures are discussed in the following sections and the parameter results are presented in Chapter 4. The Python scripts developed for determining parameter values are included in Appendix A.

3.2.5.1 Following Behavior

The following behavior in VISSIM defines how vehicles interact with the other objects in the network. Look ahead distance defines how far a vehicle can see forward in order to react to other vehicles and interaction objects that are in front of or next to it on the same link. However, look ahead distance and number of interaction objects and are not observable form any type of field traffic data. Therefore, these two parameters need to be calibrated by methods other than direct observation.

3.2.5.2 Desired Speed

Desired speed (*m/s*) is the speed a vehicle wants to maintain if it is not hindered by other vehicles or objects in the network. By default, the desired speed distribution is a uniform distribution and the cumulative distribution appearing as a straight line. For example, the desired speed distribution for a posted speed of 50 km/h is shown in Figure 16. However, the desired speed is affected by many different factors such as local driver population, land use, street parking and sidewalk presence (Silvano et al., 2020). Therefore, VISSIM's default distribution may not be able to capture the characteristics of the modelled roadway.

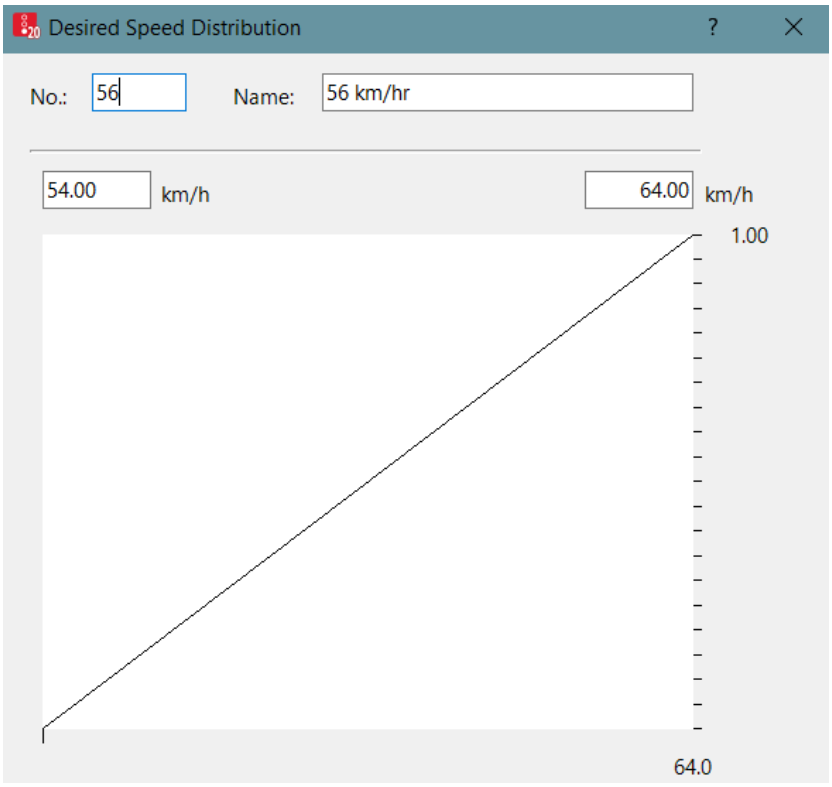

Figure 16 VISSIM Default Desired Speed Distribution (56 km/h)

To modify the desired speed using smart city data, it is necessary to determine if the observed vehicles are travelling at the free flow state. However, the driving state cannot be directly observed. In previous studies related to desired speed estimation, a threshold time headway is normally used. In the study by Vogel (2002), vehicles with different speeds were classified in groups of one second time headways from 1 to 12s. Their result indicated that all vehicles with more than 6s time headway were in the free flow regime. In order to only include vehicles travelling in a free flow state, a 6s headway threshold is used for the desired speed analysis. In addition, to eliminate outliers that are travelling extremely slow, a minimum speed threshold must be applied. The speed threshold for desired speed was obtained by comparing the field observations and simulation results when different minimum speed thresholds were applied. The results of the comparison is shown in Figure 17. Velocity of 40 km/h was used as the minimum speed threshold because it resulted in the lowest MANE. Then, the maximum speed of each recorded vehicle traveling at free flow state can be plotted in a probability plot to show the

cumulative distribution of the desired speed.

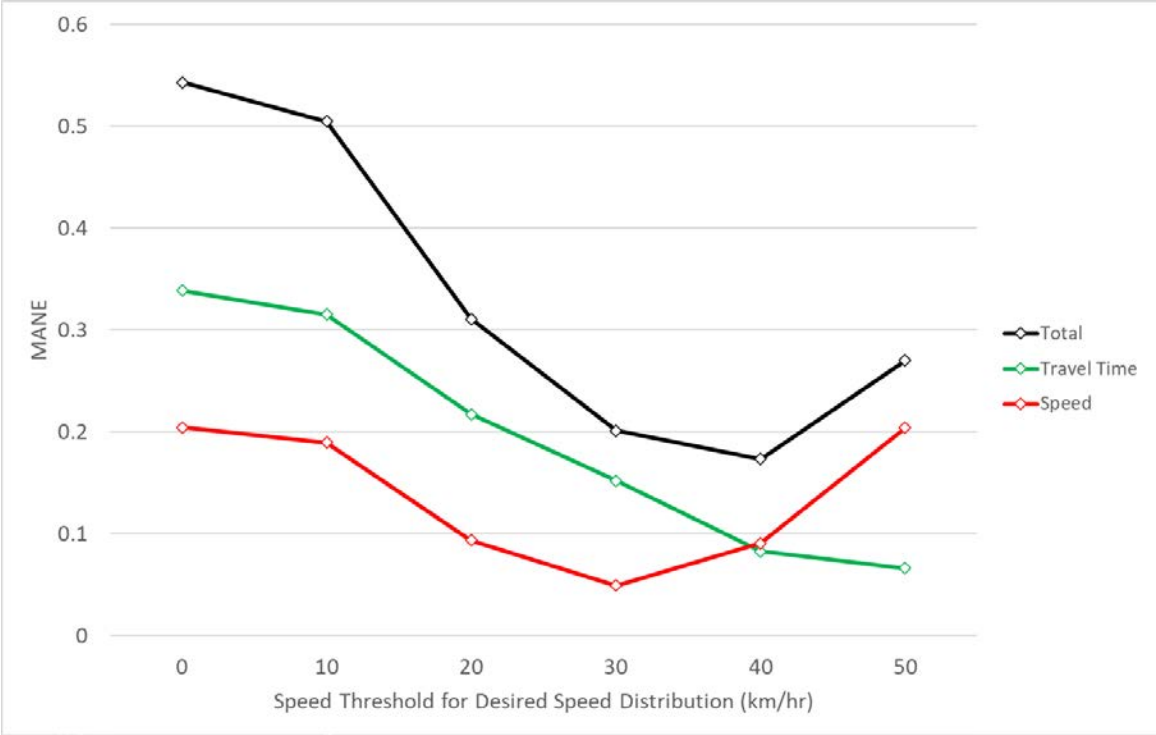

Figure 17 Comparison of MANE Using Different Speed Thresholds

3.2.5.3 Desired Acceleration/Deceleration

In VISSIM, desired acceleration/deceleration (m/s^2) are functions of the vehicle's current speed. As shown in Figure 18, VISSIM defines minimum, median, and maximum desired acceleration/deceleration at a certain speed to account for different driving behaviours and vehicle properties. A vehicle's actual acceleration/deceleration rates at a certain speed will change between the maximum-minimum range following a stochastic distribution.

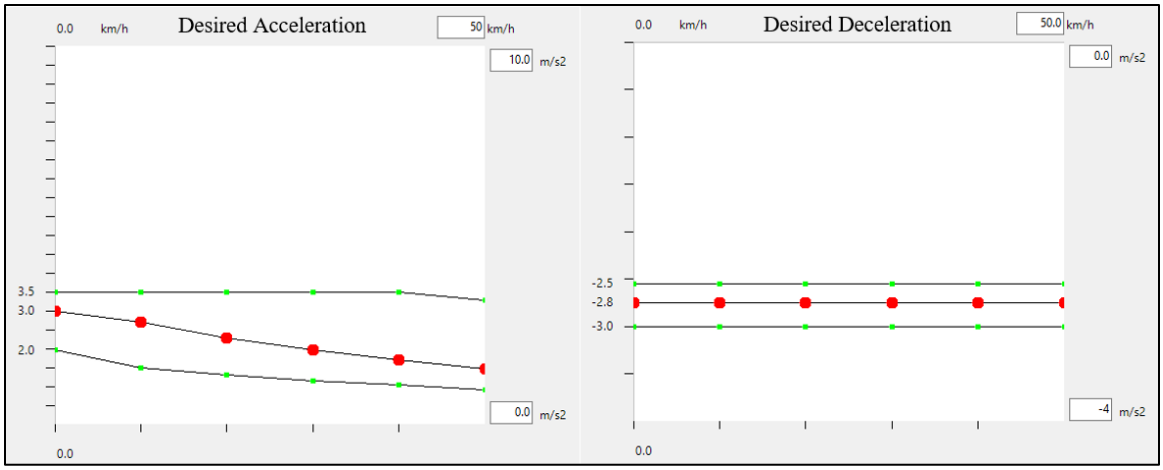

Figure 18 VISSIM Default Desired Acceleration/Deceleration Function

With the acceleration/deceleration and speed data available in the smart city dataset, the relationship between desired acceleration/deceleration and current speed can be developed. As in VISSIM's default settings, the observed speeds can be classified into different intervals for every 10 km/h increment. Percentile thresholds can be used to determine the minimum, median and maximum acceleration/deceleration for each speed interval to eliminate outliers and irregular behaviours.

To decide what percentile thresholds to use, different percentile values can be tested by running the VISSIM model with different percentile thresholds (5^{th}, 10^{th}, and 15^{th} percentiles for minimum acceleration/deceleration, 50^{th} percentile for median acceleration/deceleration, and 95^{th}, 90^{th}, and 85^{th} percentiles for maximum acceleration/deceleration). In this research, the minimum, median and maximum desired acceleration/deceleration determined by using 5^{th}, 50^{th}, and 95^{th} percentiles of the acceleration data were found to generate the least MANE. The results of the threshold test are shown in Table 9.

Table 9 Threshold Test Results for Desired Acceleration/Deceleration

Test	Minimum	Median	Maximum	MANE
#1	5^{th}	50^{th}	95^{th}	0.161
#2	10^{th}	50^{th}	90^{th}	0.175
#3	15^{th}	50^{th}	85^{th}	0.164

3.2.5.4 Car Following Model

In VISSIM, the car following behaviour in urban environments is modelled by the Wiedemann 74 model as shown in Figure 19. This car following model was originally developed by Wiedemann in 1974 and it was recalibrated using instrumented vehicle to measure the thresholds shown in Figure 19 (Reiter, 1994). When a faster vehicle approaches a leading vehicle, the faster vehicle will start decelerating to match the speed of the leading vehicle. However, the speed of the following vehicle may get too low because the driver cannot accurately estimate the leading vehicle speed. Then, the following vehicle will accelerate a little to match the speed of the leading vehicle. This results in an iterative process of acceleration and deceleration due to the driver's imperfections in determining the exact speed of the lead vehicle (Aghabayk et al., 2013). In VISSIM, two thresholds in Figure 19 can be modified by the user: the distance between two stationary vehicles (*ax*) and the minimum following distance which is considered as a safe distance by drivers (*d*).

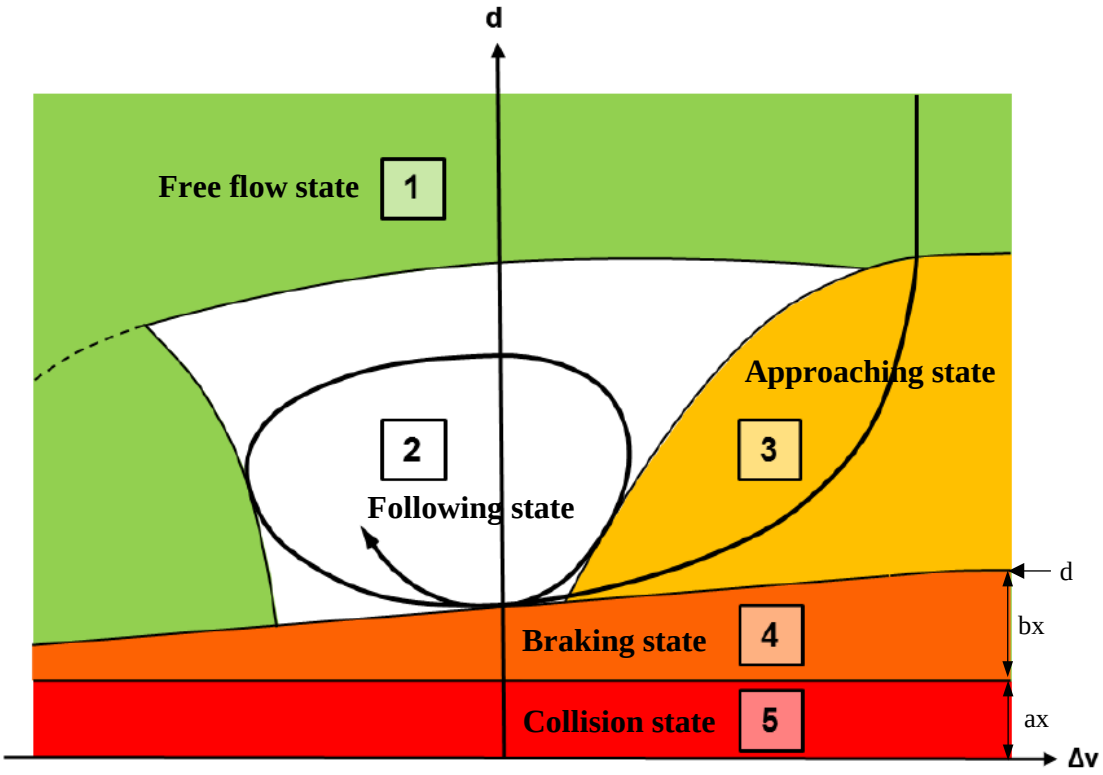

Figure 19 Car-following Model (VISSIM, 2020)

Three parameters can be changed to modify the two-car following behaviour thresholds: average standstill distance, additive part of safety distance, and multiplicative part of safety distance. The relationship between those parameters and desired safety distance is shown in Equation 11 and 12.

$$d = ax + bx \tag{11}$$

$$bx = (bx_{add} + bx_{mult} \times z)\sqrt{v} \tag{12}$$

where d is the desired safety distance (m), ax is the standstill distance (m), bx_{add} is the additive part of safety distance, bx_{mult} is the multiplicative part of safety distance, z is a random value between 0 and 1 that is truncated normally distributed around 0.5 with a standard deviation of 0.15, and v is the following vehicle speed (*m/s*).

To determine parameter values used in Equation 11 and 12, the traffic data must contain following distance between two vehicles and the following vehicle speed. In the NGSIM dataset, space headway is used to record the distance from following vehicle to its proceeding vehicle. The space headway is measured as the distance between the front-center of a vehicle to the front-center of the preceding vehicle. Therefore, the length of the preceding vehicle is subtracted from the space headway to determine the desired safety distance. The desired safety distance can be calculated by Equation 13:

$$following\ distance = \text{space headway} - \text{preceding vehicle length} \tag{13}$$

Desired safety distance is the minimum safety following distance considered by drivers. To account for extreme values and outliers, the 5th, 10th, and 15th percentiles of the following distances for each vehicle recorded in the database is tested to represent the minimum safety following distance. The value of the desired safety distance will impact the values of bx_{add} and bx_{mult}. For this research, the minimum safety following distances determined by using 10th percentile of the following distances was found to generate the least MANE. The results of the threshold test are shown in Table 10. Then, the collected desired safety distances for each vehicle are used to calibrate the other parameters in the Equation 12.

Table 10 Threshold Test Results for Desired Safety Distance

Percentile	MANE
5th	0.172
10th	0.169
15th	0.170

Standstill distance is the distance between two stationary vehicles. To estimate it, all vehicle records that have zero speed and a stationary preceding vehicle are extracted from the database. Then, the standstill distance can be calculated by using Equation 13.

After the standstill distance and desired safety distance between each pair of vehicles are determined, bx can be calculated by Equation 11. Equation 12 can be transformed to Equation 14 by moving known values to one side:

$$\frac{bx}{\sqrt{v}} = (bx_{add} + bx_{mult} \times z) \tag{14}$$

bx_{add} and bx_{mult} are two parameters can not be directly observed from the field traffic data, but they can be estimated from their mathematical relationship with bx. The coefficient $(bx_{add} + bx_{mult} \times z)$ follows a normal distribution with mean of $bx_{add} + 0.5bx_{mult}$ and standard deviation of $0.15bx_{mult}$. Therefore, when mean and standard deviation of bx are known, bx_{add} and bx_{mult} can be solved from the following equations:

$$\mu_{\frac{bx}{\sqrt{v}}} = (bx_{add} + 0.5bx_{mult}) \tag{15}$$

$$\sigma_{\frac{bx}{\sqrt{v}}} = 0.15bx_{mult} \tag{16}$$

where $\mu_{\frac{bx}{\sqrt{v}}}$ is mean of $\frac{bx}{\sqrt{v}}$ and $\sigma_{\frac{bx}{\sqrt{v}}}$ is standard deviation of $\frac{bx}{\sqrt{v}}$.

However, sometimes Equations 15 and 16 results in values of bx_{add} and bx_{mult} that are not within reasonable ranges. According to previous studies, parameter ranges are modified on the basis of the field speed data and the estimated saturation flow rates (Park and Qi, 2005). Coefficient bx_{add} should be between 1.0 to 5.0 and bx_{mult} should be

between 1.0 to 6.0. A linear program is proposed in this research to impose these constraints on bx_{add} and bx_{mult}, which can be solved by minimizing the objective:

$$\sigma_{\frac{bx}{\sqrt{\bar{v}}}} - 0.15bx_{mult} \tag{17}$$

Subject to:

$$\mu_{\frac{bx}{\sqrt{\bar{v}}}} = bx_{add} + 0.5bx_{mult} \tag{18}$$

$$1 \leq bx_{add} \leq 5 \tag{19}$$

$$1 \leq bx_{mult} \leq 6 \tag{18}$$

3.2.5.5 Lane Change

In VISSIM, two types of lane changes are modelled. The first type of lane change is necessary for a vehicle to reach the next connector of a route, while the second type of lane change occurs if there is more space available in an adjacent lane and a higher speed is desired (PTV AG, 2020). However, most users only use the default parameter values because classic traffic data collection does not include lane change information. With smart city traffic data, this study proposes procedures to determine values for the lane change parameters previously shown in Table 6.

Figure 20 shows the entire process of extracting lane change data. Firstly, it is essential to extract vehicle records related to lane changing from the database. In the NGSIM database, Lane ID is used to track the current lane position of a vehicle. To identify lane change behaviour, an algorithm searches for the timestamp when there is a change in Lane ID in a series of vehicle records with same Vehicle ID. Previous study of the NGSIM dataset lane change behaviour indicated that the duration of most lane change behaviors is within 3-6s (Li et al., 2020). Therefore, it is assumed that 2s before and after when the lane change occurs are the times when lane changing starts and ends, respectively. Finally, the acceleration and distance headway data of the lane change vehicles and their trailing vehicles are recorded for further analysis.

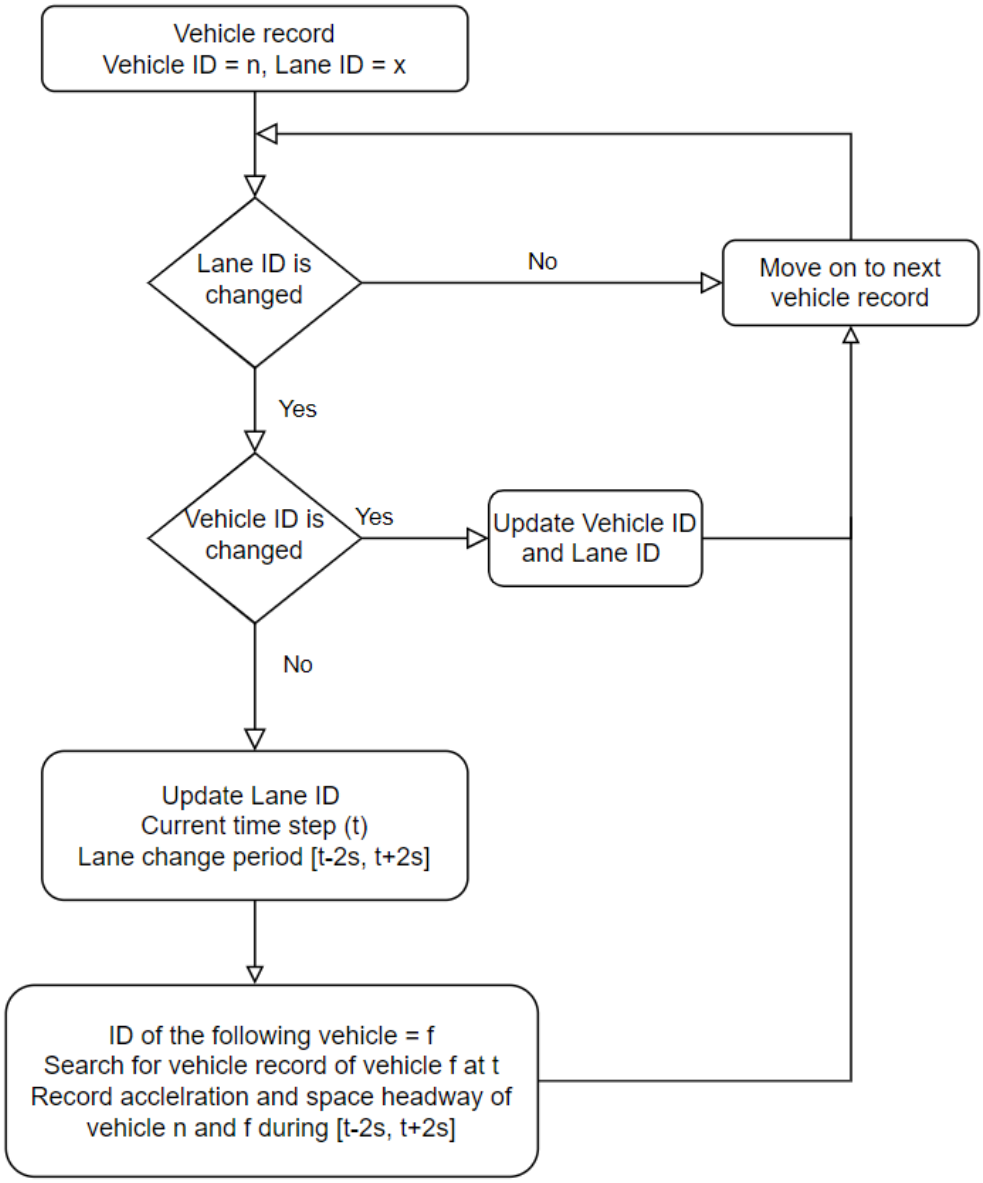

Figure 20 Flow Chart of Extract Lane Change Data

Maximum deceleration (m/s^2) and accepted deceleration (m/s^2) are the upper and lower bounds of deceleration for the lane changing vehicle and trailing vehicle during a necessary lane change. These two parameters are used to model the necessary lane change situation when a lane changing vehicle and trailing vehicle on the desired lane have to decelerate to create a gap for the lane changing vehicle. After obtaining the travel

data of the lane changing vehicle and trailing vehicle, the maximum deceleration and accepted deceleration can be determined from the vehicle deceleration data.

-1 m/s^2 per distance (*m per* -1 m/s^2) is how deceleration changes from maximum deceleration to accepted deceleration with increasing distance from the emergency stop distance. Because there is no information about distance to emergency stop spot, -1 m/s^2 per distance cannot be determined from the database.

Waiting time before diffusion (*s*) is the maximum amount of time a vehicle can wait at the emergency stop distance for a necessary lane change. When this time is reached, the vehicle is removed from the network. This parameter cannot be determined from field data.

Minimum headway (*m*) for lane changing is the minimum distance between two vehicles that must be available after a lane change. This parameter can be determined by recording the headway between a lane changing vehicle and trailing vehicle after the lane change is completed.

Safety distance reduction factor is the percentage of safety following distance that is reduced during lane changing. After a lane change is completed, the original safety distance is taken into account again. The difference between the minimum headway and maximum headway during the lane changing can be used to determine this parameter.

3.2.6 Model Calibration Using ANN

As discussed in the previous section, not all parameters can be observed. Hence, some need to be calibrated by optimizing some goodness of fit measure (e.g., segment speed or travel time). For this research, a neural network is used to calibrate the remaining parameters. The Python scripts used to calibrate VISSIM parameters using Neural Network are modified based on the research done by Daguano (2019) are included in Appendix B and C. VISSIM COM interface and neural networks scripts are coded in Python. COM interface is used to automatically run a VISSIM model with different inputs because the calibration of a VISSIM model requires thousands of runs. TensorFlow (Abadi et al., 2016) is an open-source library released by Google to

implement Machine Learning in Python. It operates as a platform for the machine learning. Keras (Chollet, 2015) is a neural network library that runs on top of TensorFlow and can simplify the creation of feedforward neural network models. It is used to configure the neural network such as define number of inputs and outputs, width of hidden layer, selection of transfer function, etc. The main approach was discussed in Section 2.2.2.4. In this study, based on the analysis results of Daguano (2019) for neural networks with different number of hidden layers and neurons, a neural network with 1 hidden layer and 50 neurons is created. The transfer function used in this study is Sigmoid function. This transfer function is used to ensure the signal output of each neuron is between 0 to 1:

$$y = \frac{1}{1+e^{-x}} \tag{19}$$

where *x* is inputs variables and *y* is neural network signal outputs.

At the beginning of the calibration, 2000 uniformly distributed parameter values are generated for each selected parameter within its defined range. Then, every generated parameter set is used as inputs to the VISSIM model to generate desired outputs (e.g., travel times and speeds) through VISSIM COM interface. After finishing all simulation runs, the inputs and outputs are combined as one dataset. Because the purpose of VISSIM calibration is to find the most appropriate parameter set (i.e., calibration output) based on the known field measurements (i.e., calibration input), the input/output order of the dataset is flipped before neural network training. Neural network training is the iterative process of tuning the weights of the neurons. The objective of the training process is to minimize the difference between outputs of neural network and the desired outputs from the training dataset. The training stops when the maximum training epochs is reached or the validation error starts to increase. After training and testing are completed, a trained neural network that explains the relationship between vehicle performance measures (e.g., travel times and speeds) and VISSIM parameters is created. Then, the testing dataset is processed by the trained Neural network to evaluate the training performance for each variable. Correlations between the predicted parameter outputs and the

parameter outputs from the testing dataset are calculated with Equation 20 (Daguano, 2019).

$$r_{xy} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{(x_i - \bar{x})^2 (y_i - \bar{y})^2}}, -1 \leq r_{xy} \leq 1 \quad (20)$$

where *x* is the predicted parameter outputs from the Neural Network and *y* is the parameter values from the testing dataset.

Parameters with correlations less than 0.3 are considered as low correlation and it is suggested to use the VISSIM default parameters values for these parameters (Ratner, 2009). In the end, field collected measurements can be used as inputs to the trained neural network and selected VISSIM parameters are generated as outputs. The general process for model calibration using a neural network is demonstrated in Figure 21.

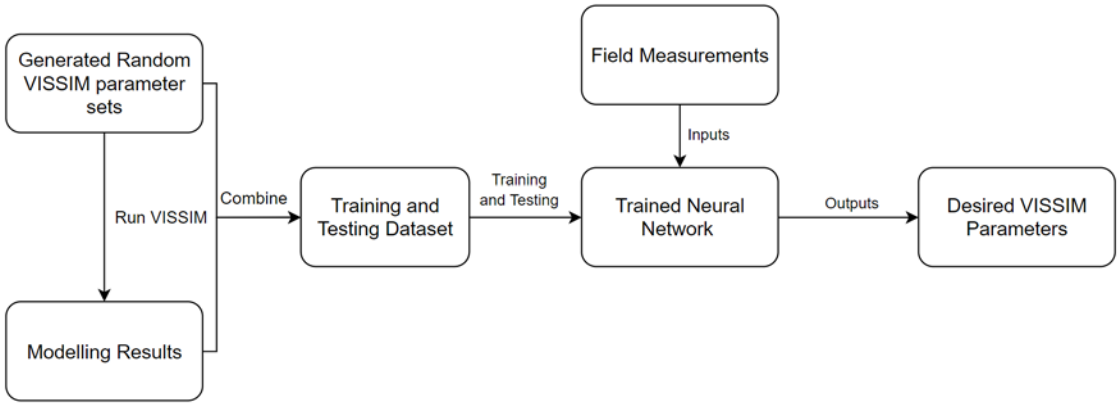

Figure 21 Neural Network Calibration Process

4 Parameter Results from the NGSIM Data

This section presents the VISSIM parameters determined from the NGSIM data by following the methods proposed in Section 3.2.5.

4.1 Desired Speed

Figure 22 demonstrates the desired speed distribution obtained from the NGSIM data. The result obtained from NGSIM data shows a similar distribution pattern compared to the VISSIM default distribution (Figure 16).

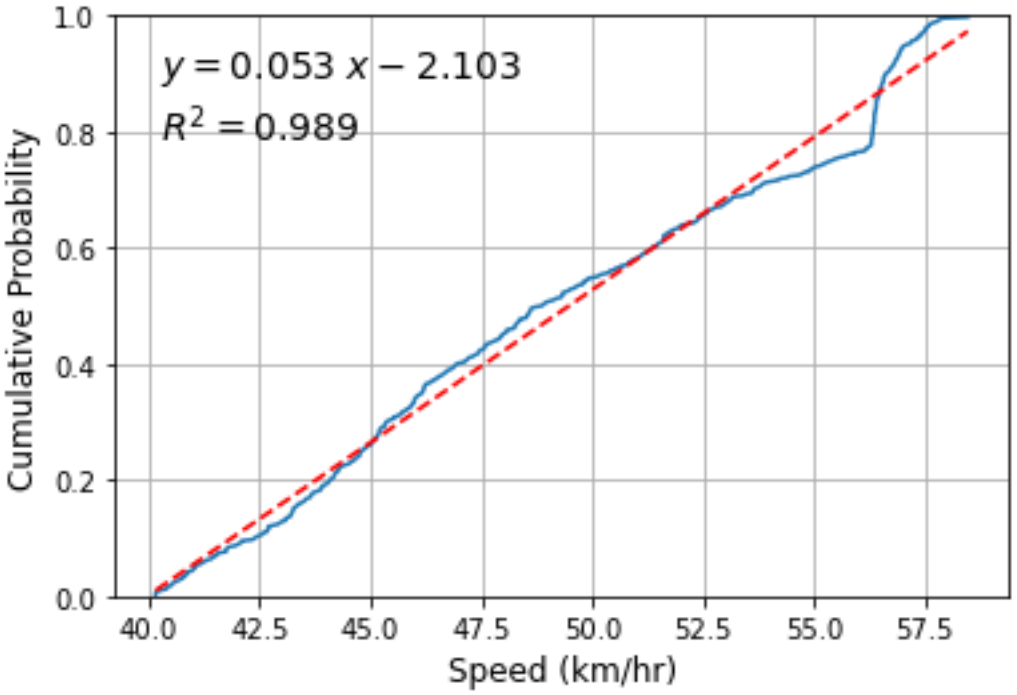

Figure 22 Desired Speed Distribution of Peachtree Street

4.2 Desired Acceleration/Deceleration

Figure 23 shows the desired acceleration/deceleration values from the NGSIM data by following the methodology described in the previous chapter. The minimum, median and maximum desired acceleration/deceleration are determined by using 5^{th}, 50^{th}, and 95^{th} percentiles of the acceleration data. Compared to the VISSIM default setting (Figure 18), the maximum desired acceleration values are similar to the VISSIM defaults, while the

median and minimum values are not. The results shown in Figure 23 share similar trends with the desired acceleration/deceleration relationship obtained in Jie et al. (2011) as shown in Figure 24. However, the results obtained from this study are closer to the VISSIM default setting compare to the results of Jie et al. (2011).

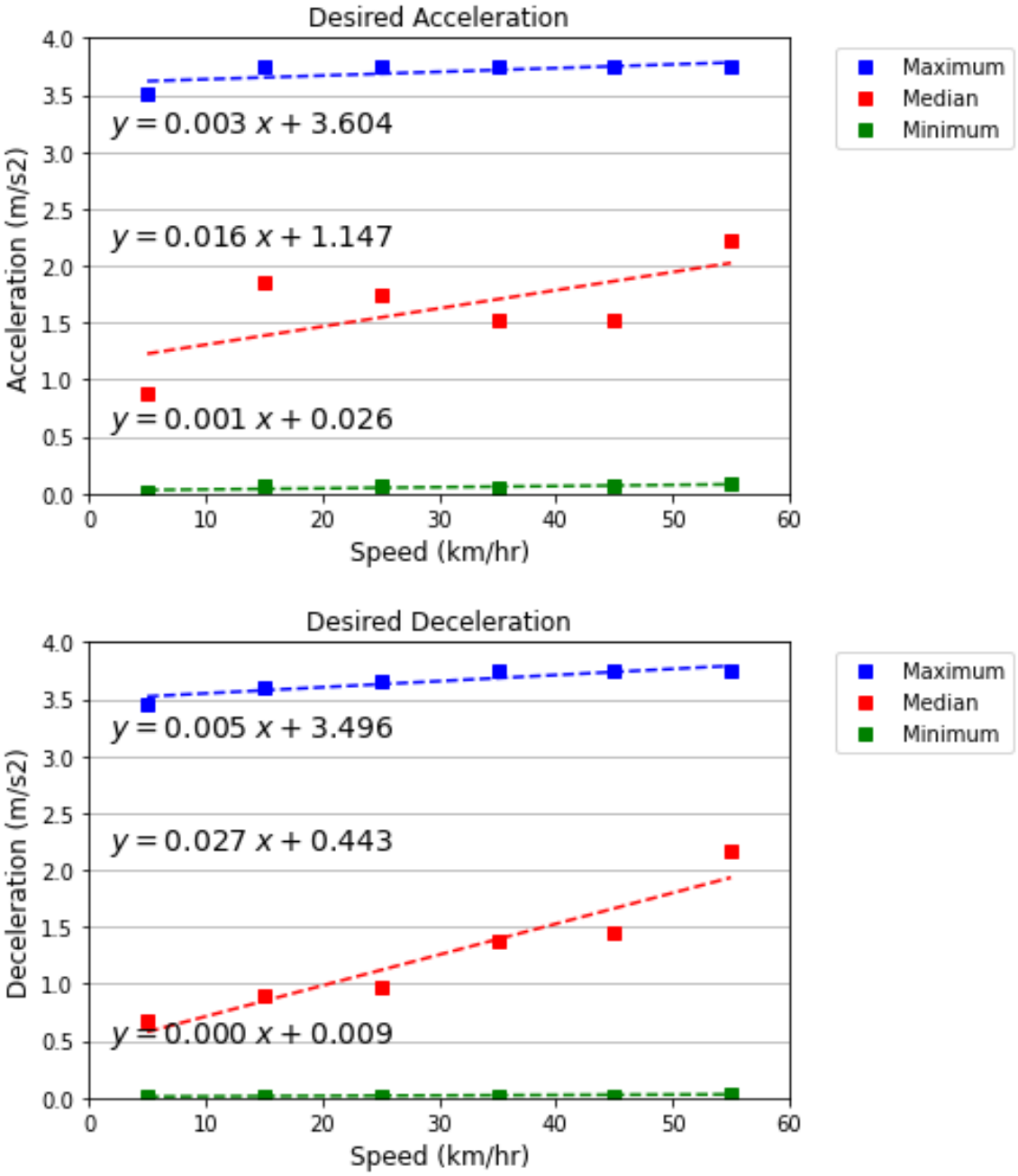

Figure 23 Desired Acceleration/Deceleration Estimated from NGSIM Data

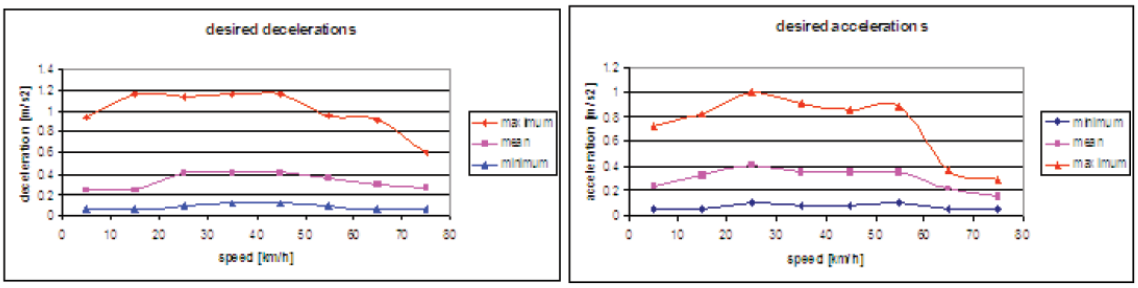

Figure 24 Desired acceleration and deceleration (Jie et al., 2011)

The difference between the VISSIM default settings and the results generated in this study and the previous study are partly because it is hard for drivers to maintain a constant speed. Figure 9 indicates that significant amount of acceleration data is within -1 m/s^2 to 1 m/s^2. Therefore, the minimum desired acceleration/deceleration in both studies are a lot lower than the VISSIM default because they also capture these small speed variances.

In order to generate more accurate results, some acceleration data in the lower ranges should be removed. It is recommended to set a cut-off point somewhere between 0 to 1 m/s^2 (0 to -1 m/s^2 for deceleration) when determining the desired acceleration/deceleration for each speed range to force the trend of each line close to the trend of the VISSIM default desired acceleration/deceleration function. The cut-off points used in this study are listed in Table 11.

Table 11 Cut-off Points for Desired Acceleration/Deceleration Calculation

Speed Interval (km/h)	Acceleration Thresholds (m/s^2)	Deceleration Thresholds (m/s^2)
0-10	1.0	-1.0
10-20	0.5	-0.5
20-30	0.5	-0.5
30-40	0.5	-0.5
40-50	0.5	-0.5
50-60	0.1	-0.1

The modified desired acceleration/deceleration values are shown in Figure 25. The trend lines (dashed-red) for each case are used to generate VISSIM inputs, not the raw values. The modified desired acceleration/deceleration values are more similar to the VISSIM default settings and they are more representative of intended accelerations and decelerations compared to the previous results (Figure 23).

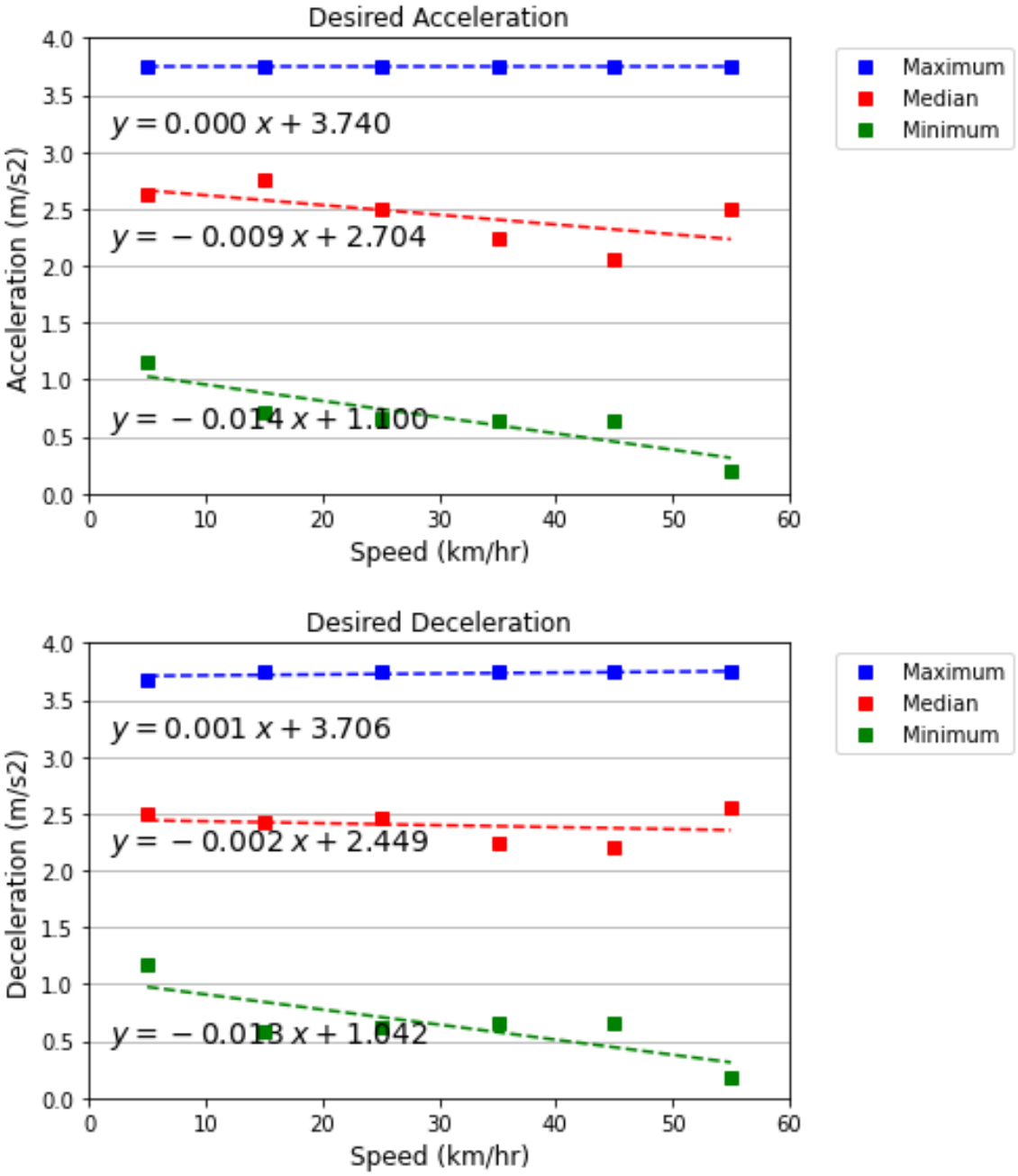

Figure 25 Modified Desired Acceleration/Deceleration Estimated from NGSIM Data

4.3 Car Following Model

4.3.1 Standstill Distance

The distribution of the distance between two stationary vehicles is plotted in Figure 26. Due to errors in automated video processing, some vehicle records might have a space headway less than the length of the preceding vehicle which results in a negative distance between two vehicles. On the other hand, some vehicle records might have very large standstill distance (e.g., > 10m). These data are removed from the results. The standstill distances of 216 vehicles are determined from the NGSIM database after removing unreasonable data and the average standstill distance is about 2.8m. Therefore, 2.8m is used as the parameter value for the average standstill distance.

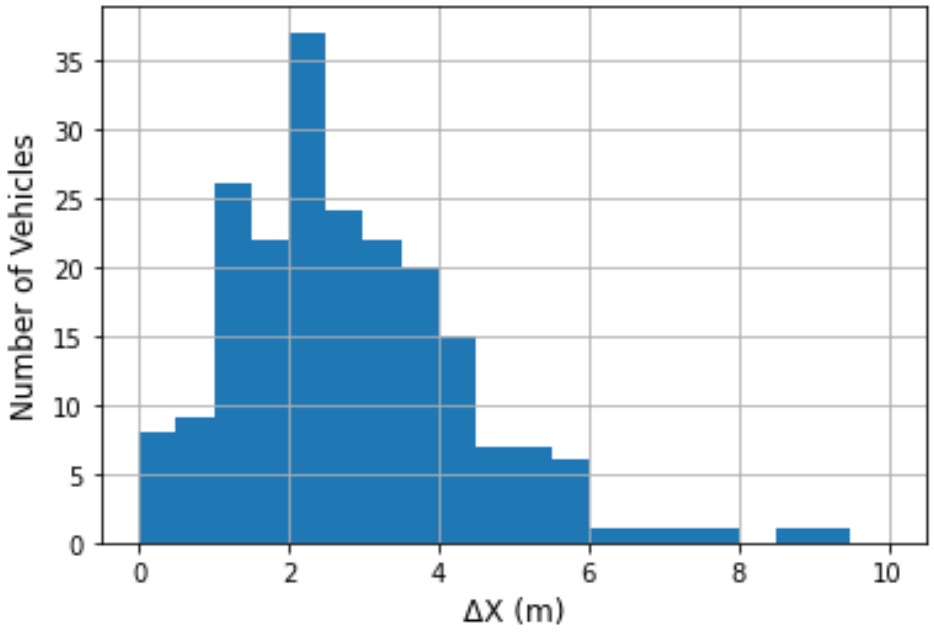

Figure 26 Standstill Distance Distribution

4.3.2 Desired Safety Distance

The distribution of the desired safety distance for each vehicle recorded in the NGSIM database is shown in Figure 27. This figure indicates that most vehicles have a desired safety following distance less than 10m. Therefore, vehicles with desired safety greater than 10m are excluded since they do not represent the general driving behaviour in the study area.

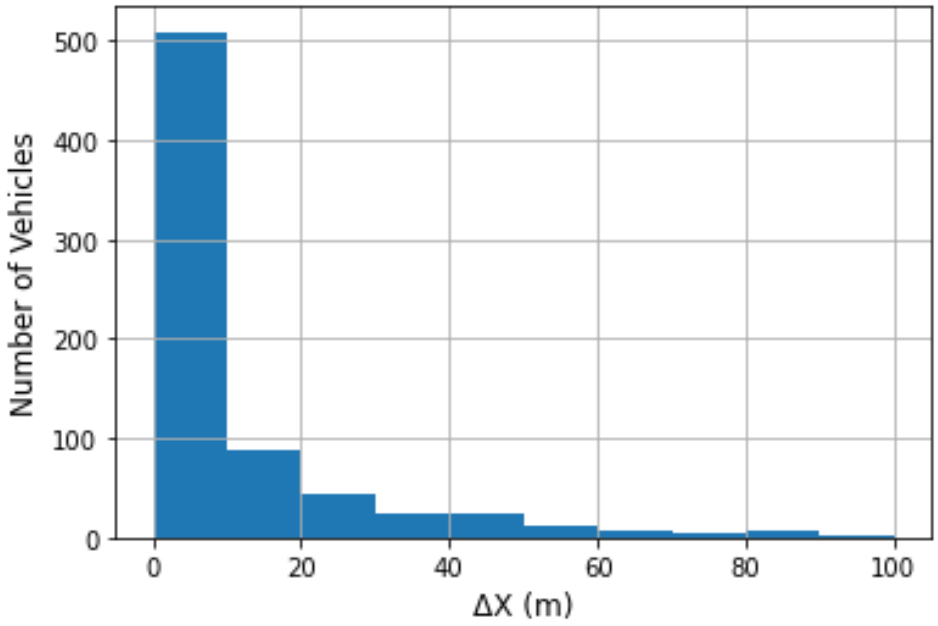

Figure 27 Distribution of the Following Distance

4.3.3 Additive and Multiple Parts of desired safety distance

After determining the standstill distance (2.8m) and desired safety distance between each pair of leading-following vehicles, the other two parameters for the Wiedemann 74 car following model, bx_{add} and bx_{mult}, are derived from Equations 9 to 14, and the linear program formed by Equations 15 to 18. The linear program finds that bx_{add} has a value of 1.0m and bx_{mult} has a value of 3.87m.

4.4 Lane Changing Model

The lane changing data are extracted from the NGSIM database by following the process demonstrated in52 Section 3.2.5.5.

4.4.1 Maximum Deceleration and Accepted Deceleration

The distribution of the lane changing vehicle and trailing vehicle deceleration rates are shown in Figure 28. In both distributions, the percentage of data between -0.5 m/s^2 and 0 m/s^2 is extremely high. It is reasonable to assume that decelerating vehicles in this range did not decelerate to accommodate necessary lane change but rather were in a state of deceleration as a part of the normal driving speed variance. Therefore, for the purpose

of determining the maximum deceleration rate and accepted deceleration rate necessary for lane changes, the data in the range between -0.5 $m/s^2$and 0 m/s^2 are removed. The 95th, 90th, and 85th percentiles are tested to represent the maximum deceleration rate and 5th, 10th, and 15th percentiles are tested to represent the accepted deceleration rate for a lane changing vehicle and trailing vehicle. However, the MANEs are same when using different thresholds for maximum deceleration rate and accepted deceleration rate of a lane changing vehicle and trailing vehicle. Therefore, any threshold value can be used. For this research, 85th and 15th percentiles are chosen because the results they generated are closest to the VISSIM default values. The results are shown in Table 12.

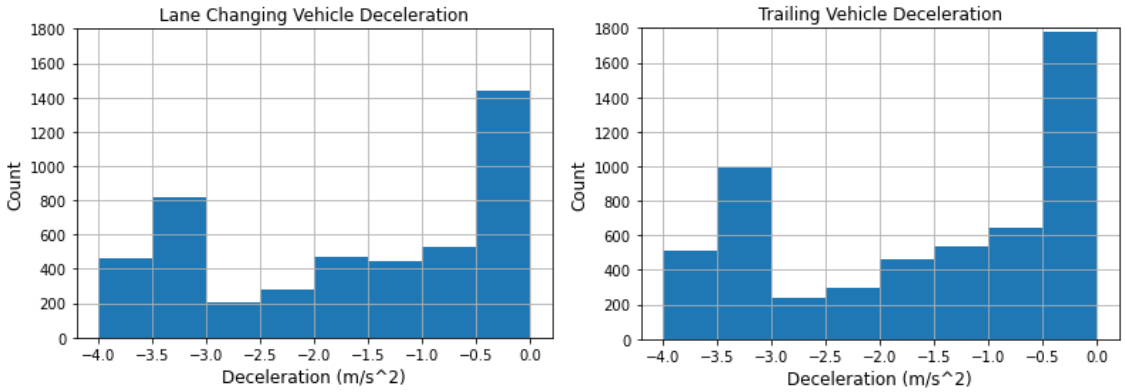

Figure 28 Vehicle Deceleration Distribution

Table 12 Maximum Deceleration and Accepted Deceleration

	Lane changing Vehicle	Trailing Vehicle
Maximum Deceleration (m/s^2)	-3.49 (-4.0*)	-3.47 (-3.0*)
Accepted Deceleration (m/s^2)	-0.95 (-1.0*)	-0.94 (-1.0*)

*Default value in VISSIM

4.4.2 Minimum Headway

The minimum headway that must be available after a lane change is determined by finding the minimum distance from a trailing vehicle to the lane changing vehicle at the end of the lane changing process. The minimum headway of 0.55m is determined from the lane changing data extracted from the MGSIM database; the default value is 0.5m in VISSIM.

4.4.3 Safety Distance Reduction Factor

The safety distance reduction factor is determined by finding the average ratio of minimum distance headway and maximum distance headway between a trailing vehicle and lane changing vehicle during a lane change. Vehicle records with maximum distance headways less than 10m are considered by following the same rule used in section 4.3.2. The safety distance reduction factor of 0.8 is determined from the lane changing data extracted from the NGSIM database; the default value is 0.6 in VISSIM.

4.5 Summary of Key Findings

This section describes a case study on how to calibrate a microscopic traffic simulation model (VISSIM) by applying the proposed methodology described in Chapter 3 using smart city-like vehicle trajectory data (NGSIM data). Table 13 summarizes the differences between the VISSIM default parameters and those estimated from the NGSIM data. The percentage difference between VISSIM default and parameter value estimated from the NGSIM data for car following parameters are relatively large because the additive part and multiple part of desired safety distance cannot be directly estimated in the field. These two parameters are determined by using a linear program that minimizes the difference between the observed following distance and car following model output. Overall, the results indicates that while some VISSIM default parameter values are close to the field measured values, there is still a large gap between default values and estimated values for many parameters. Therefore, it is necessary to utilize smart city data to verify if the VISSIM default parameter values fit the real-world traffic conditions under study.

Table 13 Parameters Determined from NGSIM data

Category	Parameters	Default	Calibrated (Smart city data)	Absolute Difference	% Difference
Vehicle Performance	Desired speed distribution lower bound (km/h)	54	40	14	26%
	Desired speed distribution upper bound (km/h)	64	58	6	9%
	Desired acceleration at 0 -10 km/h (m/s^2)	3.00	2.66	0.34	11%
	Desired deceleration at 0 -10 km/h (m/s^2)	-2.75	-2.44	-0.31	11%
Car following model	Average standstill distance (m)	2	2.8	-0.8	-40%
	Additive part of desired safety distance	2	1	1	50%
	Multiple part of desired safety distance	3	3.87	-0.87	-29%
Lane change	Maximum deceleration (own) (m/s^2)	-4	-3.49	-0.51	13%
	Maximum deceleration (trailing) (m/s^2)	-3	-3.47	0.47	-16%
	Accepted deceleration (own) (m/s^2)	-1	-0.95	-0.05	5%
	Accepted deceleration (trailing) (m/s^2)	-1	-0.94	-0.06	6%
	Min. headway (front/rear) (m)	0.5	0.55	-0.05	-10%
	Safety distance reduction factor	0.6	0.8	-0.2	-33%

The other key finding is that field measurements cannot be directly used to determine VISSIM parameters due to difference in nature between real-world driving and computer simulated driving states. For instance, in the process of determining desired acceleration/deceleration and maximum deceleration/accepted deceleration for lane changes, the data in the low range are removed because these acceleration/deceleration data reflect unconscious reactions of drivers (in the “following state”), rather than intended accelerations and decelerations in another driving “state”. It is important to consider the impact of human factors to the usability of traffic data.

5 Model Calibration and Evaluation

5.1 Evaluation of Parameters Determined from Smart City Data

The value of the parameters determined from smart city data is evaluated by comparing the MANE value of the simulation results with the MANE value generated by the VISSIM default parameters. Parameters that could not be determined directly from the smart city dataset remain at their default values. Each parameter set is implemented in VISSIM and run 200 times with different random seeds. Then, the modelling results are compared with the field measurements. The summary of parameter values and MANE results are shown in Table 14.

Table 14 Parameter Values and MANE Results

Category	Parameters	Default	Calibrated (Smart city data)
Vehicle performance	Desired speed distribution*	54 km/h-64 km/h	40 km/h-58 km/h
	Desired acceleration*	Default	Modified See Figure 25
	Desired deceleration*	Default	Modified See Figure 25
Following behaviour	*Look-ahead distance min (m)***	0	0
	*Look-ahead distance max (m)***	250	250
	*Number of interaction objects***	4	4
Car following model	Average standstill distance (m)*	2	2.8
	Additive part of desired safety distance*	2	1
	Multiple part of desired safety distance*	3	3.87
Lane change	Maximum deceleration (own) (m/s^2)*	-4	-3.49
	Maximum deceleration (trailing) (m/s^2)*	-3	-3.47
	*-1 m/s^2 per distance (own) (m)***	100	100
	*-1 m/s^2 per distance (trailing) (m)***	100	100
	Accepted deceleration (own) (m/s^2)*	-1	-0.95
	Accepted deceleration (trailing) (m/s^2)*	-1	-0.94
	Min. headway (front/rear) (m)*	0.5	0.55
	Safety distance reduction factor*	0.6	0.8
	*Waiting time before diffusion (s)***	60	60
MANE	Travel time	0.071	0.075
	Speed	0.290	0.081
	Total	0.362	0.156

* Directly estimated parameters. *** Parameters cannot be directly measured*.

According to the objective function results, although the travel time errors are similar between the two parameter sets, the performance of the estimated parameter set is more than three times better than the default parameter set in term of reproducing real-world vehicle travel speeds. The average difference between field measurements and modelling results are below 10% for average travel times and average section speeds using the estimated parameter set.

5.2 Neural Network Combination Calibration Results

To further improve the model accuracy (i.e., reduce MANE), the parameters that could not be directly estimated from the smart city data can be calibrated. The VISSIM model is calibrated using a neural network according to the methodology described in the section 3.2.6.

5.2.1 Experiment 1: Smart City Data + NN Calibration

In the first experiment, in addition to the parameters determined in Section 4, the neural network is used to calibrate the model parameters that cannot be directly estimated. The purpose of this experiment is to verify if the calibration performance is improved after using neural network to calibrate these parameters compare to the results in Table 14. The MANE values of the modelling results generated from the combination of directly estimated parameters and neural network calibrated parameters are shown in Table 15. Correlation coefficient value represents the correlation between NN calibrated VISSIM input parameter and the vehicle performance measures. Parameters with correlations less than 0.3 are considered as low correlation and it is suggested to use the VISSIM default parameters values for these parameters.

Table 15 First Experiment Parameter Values and MANE Results

Category	Parameters	Combined (Smart city data and NN)	Correlation Coefficients
Vehicle performance	Desired speed distribution*	40 -58 km/h	-
	Desired acceleration*	Modified See Figure 25	-
	Desired deceleration*	Modified See Figure 25	-
Following behaviour	*Look-ahead distance min (m)***	52.5	-0.004
	*Look-ahead distance max (m)***	251.73	0.84
	*Number of interaction objects***	7	0.68
Car following model	Average standstill distance (m)*	2.8	-
	Additive part of desired safety distance*	1	-
	Multiple part of desired safety distance*	3.87	-
Lane change	Maximum deceleration (own) (m/s^2)*	-3.49	-
	Maximum deceleration (trailing) (m/s^2)*	-3.47	-
	*-1 m/s^2 per distance (own) (m)**	124.88	-0.04
	*-1 m/s^2 per distance (trailing) (m)**	124.48	-0.08
	Accepted deceleration (own) (m/s^2)*	-0.95	-
	Accepted deceleration (trailing) (m/s^2)*	-0.94	-
	Min. headway (front/rear) (m)*	0.55	-
	Safety distance reduction factor*	0.8	-
	*Waiting time before diffusion (s)**	61	0.007
MANE	Travel time	0.074	
	Speed	0.082	
	Total	0.156	

* Directly estimated parameters. *** Parameters calibrated by the NN*.

After calibrating parameters that cannot be directly estimated from the field data, the MANE results are almost the same compared to the results in Table 14 for the estimated parameter set. This result is expected because travel time and speed are not sensitive to most of the calibrated parameters in this experiment according to the sensitivity analysis (section 3.2.3).

5.2.2 Experiment 2: NN Calibration Only

In the second experiment, all selected parameters are calibrated using the neural network assuming there are no smart city data available. The purpose of this experiment is to test the performance of the neural network calibration and compare with the modelling performance of the directly determined parameter set. However, since the desired speed

distribution and desired acceleration/deceleration functions are not editable through the COM interface, three desired speed distribution ranges (e.g., ± 5, ± 10, and ± 15 km/h from posted speed limit) and the default desired acceleration/deceleration functions are used since it is the common practice in the previous studies (Park and Schneeberger 2003; Park and Qi 2005; Park et al. 2006; Miller 2009; Daguano 2019). Two thousand parameter sets are tested in VISSIM. The calibrated parameter set and MANE results are shown in Table 16.

Table 16 Second Experiment Parameter Values and MANE Results

Category	Parameters	Calibrated (NN)	Correlation Coefficients
Vehicle performance	*Desired speed distribution***	46 -70 km/h	0.88
	Desired acceleration	Default	-
	Desired deceleration	Default	-
Following behaviour	*Look-ahead distance min (m)***	63	-0.06
	*Look-ahead distance max (m)***	250	0.66
	*Number of interaction objects***	5	0.05
Car following model	*Average standstill distance (m)***	3.0	0.86
	*Additive part of desired safety distance***	3.0	0.73
	*Multiple part of desired safety distance***	3.5	0.40
Lane change	*Maximum deceleration (own)* (m/s^2)**	-3.0	0.39
	Maximum deceleration (trailing) (m/s^2)**	-3.0	0.19
	-1 m/s^2 *per distance (own) (m)***	125	0.02
	-1 m/s^2 *per distance (trailing) (m)***	125	0.0001
	Accepted deceleration (own) (m/s^2)**	-0.81	0.19
	Accepted deceleration (trailing) (m/s^2)**	-0.76	0.11
	*Min. headway (front/rear) (m)***	3.7	0.62
	*Safety distance reduction factor***	0.6	0.10
	*Waiting time before diffusion (s)***	60	-0.03
MANE	Travel time	0.058	
	Speed	0.224	
	Total	0.282	

* Directly estimated parameters. *** Parameters calibrated by the NN*.

Compared to the results demonstrated in Table 14, although there is a 22% decrease in travel time error, the error in speed measurement almost tripled. The main reason for the increase in speed measurement error is that the neural network did not explore the entire range for the desired speed distribution due to the limitations of the VISSIM COM interface. This result indicates that the neural network is capable of optimizing the

VISSIM model, but because of the limitations in VISSIM, it is difficult to reproduce real-world vehicle speeds.

5.2.3 Experiment 3: NN Calibration + Desired Speed Distribution

In the third experiment, to accommodate the high percentage error in travel speed obtained from the second experiment, the desired speed distribution determined from the smart city data is used for all simulation runs. The purpose of this experiment is to verify if the high percentage error in travel speed in Experiment 2 was due to poorly calibrated desired speed distribution. Two thousand parameter sets are tested in VISSIM. The calibrated parameter set and MANE results are shown in Table 17.

Table 17 Third Experiment Parameter Values and MANE Results

Category	Parameters	Calibrated (NN)	Correlation Coefficients
Vehicle performance	Desired speed distribution*	40 -58 km/h	-
	Desired acceleration	Default	-
	Desired deceleration	Default	-
Following behaviour	*Look-ahead distance min (m)***	51	-0.03
	*Look-ahead distance max (m)***	251	0.80
	*Number of interaction objects***	5	0.16
Car following model	*Average standstill distance (m)***	3.1	0.89
	*Additive part of desired safety distance***	3.0	0.70
	*Multiple part of desired safety distance***	3.7	0.38
Lane change	*Maximum deceleration (own) (m/s^2)***	-3.0	0.17
	*Maximum deceleration (trailing) (m/s^2)***	-2.9	0.10
	*-1 m/s^2 per distance (own) (m)***	124	-0.06
	*-1 m/s^2 per distance (trailing) (m)***	127	-0.05
	*Accepted deceleration (own) (m/s^2)***	-0.8	0.08
	*Accepted deceleration (trailing) (m/s^2)***	-0.8	0.07
	*Min. headway (front/rear) (m)***	3.7	0.60
	*Safety distance reduction factor***	0.6	0.16
	*Waiting time before diffusion (s)***	62	0.03
MANE	Travel time	0.082	
	Speed	0.091	
	Total	0.173	

* Directly estimated parameters. *** Parameters calibrated by the NN*.

Compared to results from Experiment 2, although the travel time error has increased, the speed error is reduced by more than half. This result indicates that the desired speed distribution has a large impact on VISSIM's travel time and speed outputs. Also, because the desired speed distribution cannot be changed through VISSIM COM interface, it is very difficult to find the optimal desired speed distribution by heuristic search. Therefore, it is essential to obtain appropriate desired speed distributions from the field data in order to reproduce real-world traffic conditions.

5.2.1 Experiment 4: NN Calibration + Desired Speed Distribution & Desired Acceleration/Deceleration

The fourth experiment is same as the third experiment except the desired acceleration and desired deceleration are replaced with the estimated parameters from the smart city data. The purpose of this experiment is to determine the importance of the desired acceleration/deceleration to the modelling results. The calibrated parameter set and MANE results are shown in Table 18.

Table 18 Fourth Experiment Parameter Values and MANE Results

Category	Parameters	Calibrated (NN)	Correlation Coefficients
Vehicle Performance	Desired Speed Distribution*	40 -58 km/h	-
	Desired Acceleration*	Modified See Figure 25	-
	Desired Deceleration*	Modified See Figure 25	-
Following Behaviour	*Look-ahead distance min (m)***	51	-0.04
	*Look-ahead distance max (m)***	250	0.70
	*Number of interaction objects***	5	0.04
Car following model	*Average standstill distance (m)***	3.2	0.89
	*Additive part of desired safety distance***	3.0	0.70
	*Multiple part of desired safety distance***	3.6	0.30
Lane change	*Maximum Deceleration (own)* (m/s^2)**	-3.1	0.30
	Maximum Deceleration (trailing) (m/s^2)**	-3.0	0.17
	-1 m/s^2 *per distance (own) (m)***	126	-0.18
	-1 m/s^2 *per distance (trailing) (m)***	128	-0.15
	Accepted Deceleration (own) (m/s^2)**	-0.8	-0.15
	Accepted Deceleration (trailing) (m/s^2)**	-0.4	0.03
	*Min. headway (front/rear) (m)***	3.7	0.38
	*Safety distance reduction factor***	0.6	0.11
	*Waiting time before diffusion (s)***	61	0.07
MANE	Travel Time	0.096	
	Speed	0.081	
	Total	0.177	

* Directly estimated parameters. ** *Parameters calibrated by the NN*.

Compared to the results obtained from the third experiment, there is a slight increase in travel time error and a decrease in speed error.

5.3 Calibration Performance Using Different Field Measurements

In this section, the impact of calibration targets to the calibration performance is evaluated. On the basis of Experiment 3, instead of having two travel times and eight speeds field measurements as calibration targets, different combinations of travel time and speed measurements are explored. Figure 29 demonstrates the change in MANE with different calibration targets.

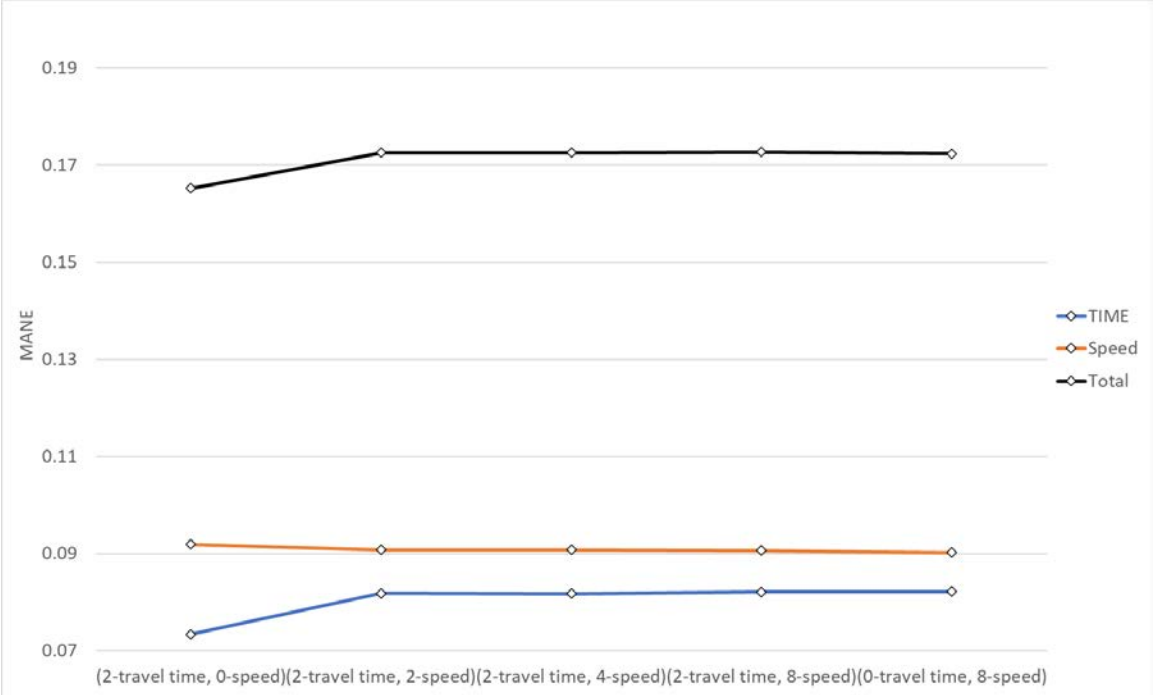

Figure 29 Changing in MANE with Different Calibration Targets

Based on the results in Figure 29, when more travel times are used as calibration targets, the MANE of the travel time decreases, which means the modelled vehicle travel times are getting closer to the real-world vehicle travel times. On the other hand, the MANE of the vehicle speed increases, which means the modelled vehicle speeds deviate from the real-world vehicle speeds. When more speeds measurements are used as calibration targets, the MANE of the travel time increases and the MANE of the vehicle speed increases. The results from this test are intuitive and it also proves the proposed calibration works properly. Overall, the total MANE does not vary much when travel speeds are used as calibration targets. This suggests that the section travel speeds are highly correlated with each other.

5.4 Summary of Key Findings

Table 19 summarizes the evaluation results of the different VISSIM model parameter calibrations. Overall, the VISSIM default parameter values produce the worst fit with field observed measurements. On the other hand, the parameter set determined from the smart city data has the best fit. Experiment 1 of using estimated parameter values and

Neural Network calibration method produces a similar result. However, it was proved in the sensitivity analysis that the Neural Network calibrated parameters in this experiment (#1) are not statistically significant to the modelling results. In the following experiments, more parameters are calibrated using the Neural Network. In Experiment 2, all parameters are calibrated using the Neural Network. It shows better performance than VISSIM default but it is still a lot worse than the estimated parameter set. In Experiment 3, on the basis of Experiment 2, the desired speed distribution is changed to the estimated values, which improves the performance substantially. In Experiment 4, on the basis of Experiment 3, the desired speed distribution and desired acceleration/deceleration rates are changed to the estimated values but it does not produce much of a difference overall compared to Experiment 3 (better speeds but worse travel times).

Table 19 Evaluation Results with Different Parameter Settings

		VISSIM Default	Calibrated (Smart city data)	Smart city data + Neural Network Calibration			
				EXP #1	EXP #2	EXP #3	EXP #4
MANE	Travel time	0.071	0.075	0.074	0.058	0.082	0.096
	Speed	0.29	0.081	0.082	0.224	0.091	0.081
	Total	0.362	0.156	0.156	0.282	0.173	0.177

The following key findings are summarized for this chapter: (1) Using VISSIM default parameter set results in poor fit with the field observations; (2) Using the parameter set determined from the smart city data and calibrated by the Neural Network can improve the modelling performance; (3) The parameter set determined from smart city data shows the best fit with the field observations; and (4) The desired speed distribution has the most impact on the modelling results when the measures of performance are travel times and vehicle speeds.

5.5 Summary of Proposed Calibration Process

This section summarizes the proposed microscopic traffic simulation model calibration process using smart city data based on the calibration methodologies and results discussed in the previous sections. As shown in Figure 30, the entire process is divided into three parts.

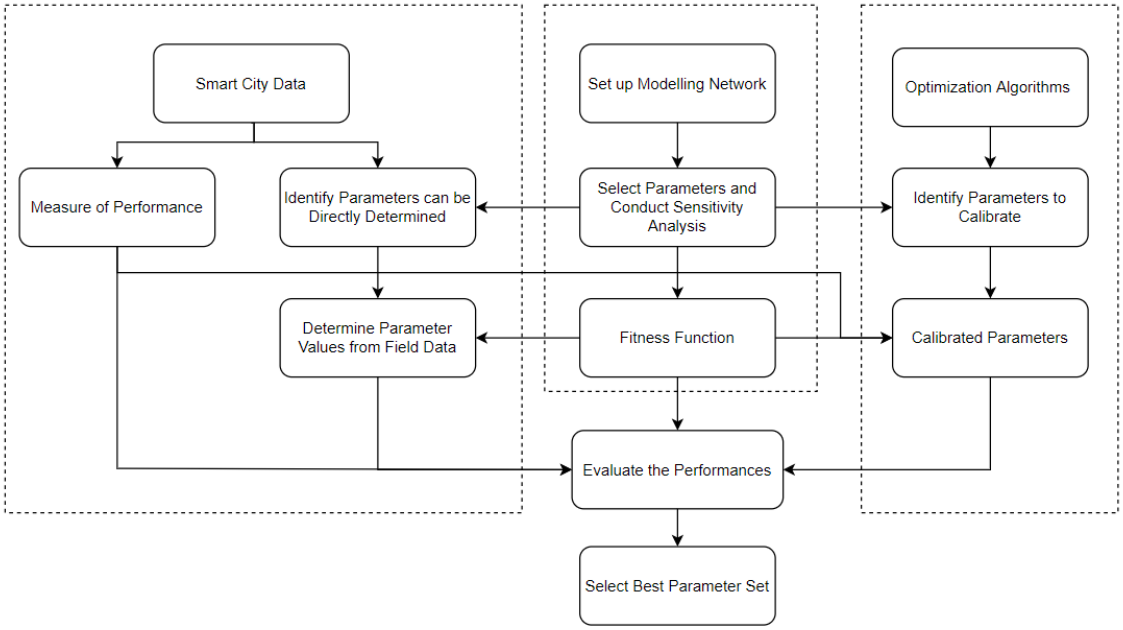

Figure 30 Proposed Calibration Process

In the first part, the modelling network is set up and parameters of interest are selected. A sensitivity analysis should be conducted to identify parameters that are statistically significant to the desired modelling outputs. Parameters that are statistically significant are carried forward to next steps. Then, a fitness function should be established for the purpose of model evaluation.

The second part involves processing the smart city data. The detailed records of vehicle movements can be used to determine modelling parameters directly. Furthermore, smart city data can be processed to provide network performance measures that can be used as calibration targets (e.g., link speeds and travel times).

The third part uses an optimization algorithm as a supplementary method to calibrate parameters that cannot be determined from the field data. The detailed calibration process may vary among different optimization algorithms.

After all three parts are finished, the modeler should be able to obtain one or more parameter sets by combining the parameters determined directly from the field data with those calibrated by the optimization algorithm. The performances of different parameter sets can then be evaluated, and the parameter set with best performance is selected.

6 Conclusion

Microscopic traffic simulation has become an important tool used by planners and engineers for traffic analysis, road network planning, and policy making. However, microscopic traffic modelling is often viewed as an inaccurate science because different calibration approaches are followed and the calibrated parameters are sometimes hard to justify. One of the main reasons behind this situation is that modelers do not have access to the right type of data that allow them to model the micro level behavior of the traffic directly. In recent years, emerging smart city technologies can provide more reliable and detailed traffic data. With availability of smart city data, modelers should be able to better calibrate modelling parameters; however, there is no guidelines or recommended processes to determine what information can be extracted from smart city data and how to incorporate these data to build more representative traffic simulation models.

Literature review of microscopic traffic simulation model calibration yielded very limited useful information about how to utilize smart city data. Instead, studies on microscopic traffic simulation model calibration are mostly related to calibration by solving an optimization problem with the objective of minimizing the differences between simulated and observed aggregated traffic behaviors. A few studies utilize smart city data but some of them are not focusing on microscopic traffic simulation models (Zhong et al. 2016) and some studies only determine a few parameters (Lu et al., 2016). Hence, this research has attempted to address this gap by proposing a methodology for determining modelling parameter values from smart city data as well as standardized procedures of model calibration with these data. The entire process is conducted in a semi-automated way with multiple Python scripts for each module of the research.

6.1 Contributions and Key Findings

This research is conducted to improve the veracity of microsimulation models and the efficiency of the microscopic traffic simulation calibration process. In this aspect, this research has successfully developed standardized processes to determine VISSIM parameters that can better reflect real-world traffic scenarios using smart city data. In addition, a neural network is used for VISSIM parameter calibration as a supplement to

determine VISSIM parameters that are not observable from field data. The performance of this joint calibration technique is compared through several experiments involving a variety of smart city data.

Several key findings were arrived from this study. First, in order to properly calibrate a VISSIM model, real-world traffic data (smart city data) are needed for all types of calibration methods due to the nature of VISSIM. Methodologies for determining VSSIM parameter values are developed and tested with the NGSIM data. The comparison between VISSIM default parameter values and parameter values determined from smart city data indicates there is a large gap between them. In addition, the modelling results show that the parameter set calibrated by smart city data gives the best modelling performance while the VISSIM default generates the worse results in these experiments. ANN calibration is tested both as a supplement technique to calibrate parameters cannot be determined from the field data and an independent calibration technique. The results indicate that ANN calibration does not give the best performance by itself, but it can be used with the smart city data calibration. Next, if smart city traffic data are used for parameter determination, the data need to be carefully processed. Other than removing outliers and unreasonable data points, the range of data should be carefully examined to accommodate the modelling environment. For example, for vehicle acceleration and deceleration data collection, the collection results will have data with very small accelerations or decelerations because sometimes human drivers are not able to keep their speeds constant but they are not intending to accelerate or decelerate (unconscious reactions). Including this portion of data will heavily impact the accuracy of parameters related to *intended* acceleration and deceleration. The impact of each parameter to the modelling results were examined. The one-way ANOVA test was performed to analyze the sensitivity of each parameter to identify key parameters and eliminate parameters that have less effect on model results. Finally, after experimenting with different ANN calibration setups, the desired speed distribution was found to be the most critical VISSIM parameter when performance measurements are travel time and vehicle speed.

6.2 Recommendations

In this research, only travel time and vehicle speed are used for calibration measurements due to lack of network performance data. Therefore, some parameter results cannot be verified since they are not statistically significant affecting travel time and vehicle speed. In the future work, different types of calibration target can be tested to verify these parameters. Also, the data used in this research were collected by video cameras. As other new technologies (i.e., LiDAR) gain popularity in the field of traffic data collection, the calibration process might change due to differences in accuracy or other data attributes.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI'16)* (pp. 265-283).

Abdalhaq, B. K., & Abu Baker, M. I. (2014). Using Meta Heuristic Algorithms to Improve Traffic Simulation. *Journal of Algorithms and Optimization*, *2*(4), 110–128.

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

Aghabayk, K., Sarvi, M., Young, W., & Kautzsch, L. (2013). A Novel Methodology for Evolutionary Calibration of Vissim by Multi-Threading. *Australasian Transport Research Forum*. , Brisbane, Australia.

Alexiadis, J., K, & Chandra, A. (2004). *Traffic Analysis Toolbox Volume I: Traffic Analysis Tools Primer*. FHWA.

Bifulco, G. N., Galante, F., Pariota, L., Russo Spena, M., & Del Gais, P. (2014). Data Collection for Traffic and Drivers' Behaviour Studies: A Large-scale Survey. *Procedia - Social and Behavioral Sciences*, *111*(2014), 721–730. https://doi.org/10.1016/j.sbspro.2014.01.106

Cambridge Systematics, Inc. (2007). *NGSIM Peachtree Street (Atlanta) Data Analysis*. FHWA.

Chollet, F. (2015). *keras*. GitHub. https://github.com/fchollet/keras

Coifman, B., & Li, L. (2017). A critical evaluation of the Next Generation Simulation (NGSIM) vehicle trajectory dataset. *Transportation Research Part B: Methodological*, *105*(4), 362–377. https://doi.org/10.1016/j.trb.2017.09.018

Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, *28*(4), 269–287. https://doi.org/10.1016/0191-2615(94)90002-7

Daguano, R. F. (2019). *Automatic calibration of traffic microsimulations with artificial neural networks* [MSc Thesis].

Dowling, R., Skabardonis, A., Halkias, J., McHale, G., & Zammit, G. (2004). Guidelines for Calibration of Microsimulation Models: Framework and Applications. *Transportation Research Record: Journal of the Transportation Research Board*, *1876*(1), 1–9. https://doi.org/10.3141/1876-01

Gipps, P. G. (1981). A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, *15*(2), 105–111. https://doi.org/10.1016/0191-2615(81)90037-0

Greenberg, H. (1959). An Analysis of Traffic Flow. *Operations Research*, *7*(1), 79–85. https://doi.org/10.1287/opre.7.1.79

Greenshields, B. D. (1933). The Photographic Method of studying Traffic Behaviour. *13th Annual Meeting of the Highway Research Board*.

Guo, Y., Tang, Z., & Guo, J. (2020). Could a Smart City Ameliorate Urban Traffic Congestion? A Quasi-Natural Experiment Based on a Smart City Pilot Program in China. *Sustainability*, *12*(6), 2291. https://doi.org/10.3390/su12062291

Haj Salem, H., Chrisoulakis, J., Papageorgiou, M., Elloumi, N., & Papadakos, P. (1994). The use of METACOR tool for integrated urban and interurban traffic control. Evaluation in corridor peripherique, Paris. *Proceedings of VNIS'94 - 1994 Vehicle Navigation and Information Systems Conference*. https://doi.org/10.1109/vnis.1994.396773

Hale, D. K., Antoniou, C., Brackstone, M., Michalaka, D., Moreno, A. T., & Parikh, K. (2015). Optimization-based assisted calibration of traffic simulation models. *Transportation Research Part C: Emerging Technologies*, *55*(2015), 100–115. https://doi.org/10.1016/j.trc.2015.01.018

Hallmark, S. L., Oneyear, N., Tyner, S., Wang, B., Carney, C., & Mcgehee, D. (2014). *Analysis of Naturalistic Driving Study Data: Roadway Departures on Rural Two-Lane Curves*. Washington, D.C. Transportation Research Board.

Hellinga, B.R. (1998), Requirement for the Calibration of Traffic Simulation Models. Department of Civil Engineering, University of Waterloo.

Hidayat, A., Terabe, S., & Yaginuma, H. (2018). WiFi Scanner Technologies for Obtaining Travel Data about Circulator Bus Passengers: Case Study in Obuse, Nagano Prefecture, Japan. Transportation Research Record: Journal of the Transportation Research Board, 2672(45), 45–54. https://doi.org/10.1177/0361198118776153

Hilpert, M., Johnson, M., Kioumourtzoglou, M.-A., Adria-Mora, B., Peters, A., Ross, J., & Chillrud, S. (2018). A New Approach for Inferring Traffic-Related Air Pollution: Use of Radar-Calibrated Crowd-Sourced Traffic Data. *ISEE Conference Abstracts*, *2018*(1). https://doi.org/10.1289/isesisee.2018.p03.2020

Ismail, K. A. (2010). *Application of computer vision techniques for automated road safety analysis and traffic data collection* [PhD Thesis].

Jackson, S., Miranda-Moreno, L. F., St-Aubin, P., & Saunier, N. (2013). Flexible, Mobile Video Camera System and Open Source Video Analysis Software for Road Safety and Behavioral Analysis. *Transportation Research Record: Journal of the Transportation Research Board*, *2365*(1), 90–98. https://doi.org/10.3141/2365-12

Jaume Barceló. (2010). *Fundamentals of traffic simulation*. Springer.

Jayakrishnan, R., Mahmassani, H. S., & Hu, T.-Y. (1994). An evaluation tool for advanced traffic information and management systems in urban networks. *Transportation Research Part C: Emerging Technologies*, *2*(3), 129–147. https://doi.org/10.1016/0968-090x(94)90005-1

Jie, L., Fangfang, Z., van Zuylen, H., & Shoufeng, L. (2011). Calibration of a micro simulation program for a Chinese city. *Procedia - Social and Behavioral Sciences*, *20*, 263–272. https://doi.org/10.1016/j.sbspro.2011.08.032

Kim, S.-J., Kim, W., & Rilett, L. R. (2005). Calibration of Microsimulation Models Using Nonparametric Statistical Techniques. *Transportation Research Record: Journal of the Transportation Research Board*, *1935*(1), 111–119. https://doi.org/10.1177/0361198105193500113

Lee, J.-B., & Ozbay, K. (2009). New Calibration Methodology for Microscopic Traffic Simulation Using Enhanced Simultaneous Perturbation Stochastic Approximation

Approach. *Transportation Research Record: Journal of the Transportation Research Board*, *2124*(1), 233–240. https://doi.org/10.3141/2124-23

Lidbe, A. D., Hainen, A. M., & Jones, S. L. (2017). Comparative study of simulated annealing, tabu search, and the genetic algorithm for calibration of the microsimulation model. *SIMULATION*, *93*(1), 21–33. https://doi.org/10.1177/0037549716683028

Li, Z., Huang, X., Wang, J., & Tong, T. (2020). Lane Change Behavior Research Based on NGSIM Vehicle Trajectory Data. *Chinese Control and Decision Conference 2020*.

Lu, Z., Fu, T., Fu, L., Shiravi, S., & Jiang, C. (2016). A video-based approach to calibrating car-following parameters in VISSIM for urban traffic. *International Journal of Transportation Science and Technology*, *5*(1), 1–9. https://doi.org/10.1016/j.ijtst.2016.06.001

Mahut, M., & Florian, M. (2010). Traffic simulation with dynameq. In *Fundamentals of traffic simulation* (pp. 323-361). Springer, New York, NY.

Ma, J., Dong, H., & Zhang, H. M. (2007). Calibration of Microsimulation with Heuristic Optimization Methods. *Transportation Research Record: Journal of the Transportation Research Board*, *1999*(1), 208–217. https://doi.org/10.3141/1999-22

Ma, T., & Abdulhai, B. (2002). Genetic Algorithm-Based Optimization Approach and Generic Tool for Calibrating Traffic Microscopic Simulation Parameters. *Transportation Research Record: Journal of the Transportation Research Board*, *1800*(1), 6–15. https://doi.org/10.3141/1800-02

Mer Group. (2020). *Big Data in SMART Cities Saves Lives*. https://mer-group.com/big-data-in-smart-cities-saves-lives/

Miller, D. M. (2009). *Developing a Procedure to Identify Parameters for Calibration of a Vissim Model* [Msc Thesis].

Montanino, M., & Punzo, V. (2013). Making NGSIM Data Usable for Studies on Traffic Flow Theory. *Transportation Research Record: Journal of the Transportation Research Board*, *2390*(1), 99–111. https://doi.org/10.3141/2390-11

National Research Council (U.S.). Transportation Research Board. (2010). *HCM 2010: Highway Capacity Manual*. Transportation Research Board.

Papageorgiou, M., Papamichail, I., Messmer, A., & Wang, Y. (2010). Traffic Simulation with METANET. *Fundamentals of Traffic Simulation*, 399–430. https://doi.org/10.1007/978-1-4419-6142-6_11

Park, B. (Brian), & Qi, H. (Maggie). (2005). Development and Evaluation of a Procedure for the Calibration of Simulation Models. *Transportation Research Record: Journal of the Transportation Research Board*, *1934*(1), 208–217. https://doi.org/10.1177/0361198105193400122

Park, B. (Brian), & Schneeberger, J. D. (2003). Microscopic Simulation Model Calibration and Validation: Case Study of VISSIM Simulation Model for a Coordinated Actuated Signal System. *Transportation Research Record: Journal of the Transportation Research Board*, *1856*(1), 185–192. https://doi.org/10.3141/1856-20

Park, B. (Brian), Won, J., & Yun, I. (2006). Application of Microscopic Simulation Model Calibration and Validation Procedure. *Transportation Research Record:*

Journal of the Transportation Research Board, *1978*(1), 113–122. https://doi.org/10.1177/0361198106197800115
Payne, H. J. (1979). FREFLO: A macroscopic simulation model of freeway traffic. *Transportation Research Record*, *722*.
Pipes, L. A. (1953). An Operational Analysis of Traffic Dynamics. *Journal of Applied Physics*, *24*(3), 274–281. https://doi.org/10.1063/1.1721265
PTV AG. (2020). PTV Vissim 2020 User Manual. PTV AG, Karlsruhe, Germany.
Rainer Wiedemann. (1974). *Simulation des Strassenverkehrsflusses*. Institut Für Verkehrswesen Der Universität Karlsruhe.
Rakha, H., & Crowther, B. (2002). Comparison of Greenshields, Pipes, and Van Aerde Car-Following and Traffic Stream Models. *Transportation Research Record: Journal of the Transportation Research Board*, *1802*(1), 248–262. https://doi.org/10.3141/1802-28
Ratner, B. (2009). The correlation coefficient: Its values range between +1/−1, or do they? *Journal of Targeting, Measurement and Analysis for Marketing*, *17*(2), 139–142. https://doi.org/10.1057/jt.2009.5
Reiter, U. (1994). Empirical Studies as Basis for Traffic Flow Models. *Second International Symposium on Highway Capacity*, *2*, 493–502.
Rouse, M. (2019). *What is smart city?* IoT Agenda. https://internetofthingsagenda.techtarget.com/definition/smart-city
Rrecaj, A. A., & M.Bombol, K. (2015). Calibration and Validation of the VISSIM Parameters - State of the Art. *TEM Journa*, *4*(3), 255–269.
Saunier, N., & Sayed, T. (2006). A feature-based tracking algorithm for vehicles in intersections. *3rd Canadian Conference on Computer and Robot Vision*.
Silvano, A. P., Koutsopoulos, H. N., & Farah, H. (2020). Free flow speed estimation: A probabilistic, latent approach. Impact of speed limit changes and road characteristics. *Transportation Research Part A: Policy and Practice*, *138*(2020), 283–298. https://doi.org/10.1016/j.tra.2020.05.024
St-Aubin, P., Miranda-Moreno, L., & Saunier, N. (2013). An automated surrogate safety analysis at protected highway ramps using cross-sectional and before–after video data. *Transportation Research Part C: Emerging Technologies*, *36*, 284–295. https://doi.org/10.1016/j.trc.2013.08.015
Thiemann, C., Treiber, M., & Kesting, A. (2008). Estimating Acceleration and Lane-Changing Dynamics from Next Generation Simulation Trajectory Data. *Transportation Research Record: Journal of the Transportation Research Board*, *2088*(1), 90–101. https://doi.org/10.3141/2088-10
Underwood, R. T. (1961). Speed, volume, and density relationship: quality and theory of traffic flow. *Yale Bureau of Highway Traffic*, 141–188.
Van Aerde, M. (1995). A single regime speed-flow-density relationship for freeways and arterials. *74th TRB Annual Meeting*.
Van Aerde, M. (1996). *INTEGRATION: Overview of Simulation Features*. Queen's University.
Vogel, K. (2002). What characterizes a "free vehicle" in an urban area?. *Transportation Research Part F: Traffic Psychology and Behaviour*, *5*(1), 15–29. https://doi.org/10.1016/s1369-8478(02)00003-7

Wunderlich, K., Vasudevan, M., & Wang, P. (2019). *Traffic Analysis Toolbox Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software 2019 Update to the 2004 Version*. FHWA.

Xu, J., Hilker, N., Turchet, M., Al-Rijleh, M.-K., Tu, R., Wang, A., Fallahshorshani, M., Evans, G., & Hatzopoulou, M. (2018). Contrasting the direct use of data from traffic radars and video-cameras with traffic simulation in the estimation of road emissions and PM hotspot analysis. *Transportation Research Part D: Transport and Environment*, *62*, 90–101. https://doi.org/10.1016/j.trd.2018.02.010

Yang, X.-S. (2014). *Nature-inspired optimization algorithms*. Elsevier.

Yu, L., Yu, L., Chen, X., Wan, T., & Guo, J. (2006). Calibration of Vissim for Bus Rapid Transit Systems in Beijing Using GPS Data. *Journal of Public Transportation*, *9*(3), 239–257. https://doi.org/10.5038/2375-0901.9.3.13

Yu, M., & David) Fan, W. (2017). Calibration of microscopic traffic simulation models using metaheuristic algorithms. *International Journal of Transportation Science and Technology*, *6*(1), 63–77. https://doi.org/10.1016/j.ijtst.2017.05.001

Zhong, R. X., Fu, K. Y., Sumalee, A., Ngoduy, D., & Lam, W. H. K. (2016). A cross-entropy method and probabilistic sensitivity analysis framework for calibrating microscopic traffic models. *Transportation Research Part C: Emerging Technologies*, *63*, 147–169. https://doi.org/10.1016/j.trc.2015.12.006

Appendices

Appendix A – Python Scripts for Determining VISSIM Parameters

1: Desired Speed Distribution

```
import pandas as pd
import numpy as np
df = pd.read_csv("NGSIM_Peachtree_Vehicle_Trajectories.csv")
#only keep time headway greater than 6s (free flow state)
time_headway_6s=df[(df['Time_Headway'] >6) & (df['v_Vel'] >=0) & (df['v_Class'] ==2)]
speed0=time_headway_6s.groupby('Vehicle_ID', sort = False).max()
speed=speed0['v_Vel']
speed_sorted=speed.sort_values()
speed_number_of_rows=len(speed)
i=0
rank=[]
for i in range (0, speed_number_of_rows):
     rank.append(i+1)
speed = speed_sorted.tolist()
p=[]
i=0
for i in range(0, speed_number_of_rows):
    p.append(rank[i]/(len(rank)+1))

import matplotlib.pyplot as plt
fig=plt.plot(speed,p)
plt.xlabel('Speed (km/h)', fontsize=12)
plt.ylabel('Cumulative Probability', fontsize=12)
plt.ylim((0.0,1.0))
plt.grid(True)

import statsmodels.api as sm
from sklearn.metrics import r2_score
speed2 = sm.add_constant(speed)
est = sm.OLS(p, speed2)
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(speed,p, 1)
p_hat = np.poly1d(z)(speed)
plt.plot(speed,p_hat,"r--")
text = f"$y={z[0]:0.3f}\;x{z[1]:+0.3f}$\n$R^2 = {r2_score(p,p_hat):0.3f}$"
plt.gca().text(0.05, 0.95, text,transform=plt.gca().transAxes,
    fontsize=14, verticalalignment='top')
print(est2.summary())
```

2: Desired Acceleration

```
import pandas as pd
import numpy as np

df = pd.read_csv("NGSIM_Peachtree_Vehicle_Trajectories.csv")

#speed between 0, 10
df1=df[(df['v_Vel'] >0) & (df['v_Vel'] <10) & (df['v_Acc'] >0)& (df['v_Class'] ==2)]
a=df1['v_Acc']
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, 0.5,1,2,2.5,3,3.5,4]
plt.hist(a, bins= bin_edge)
plt.title('Vehicle Acceleration')
min_acc1 = df1.v_Acc.quantile(0.15)
mean_acc1 = df1.v_Acc.quantile(0.5)
max_acc1 = df1.v_Acc.quantile(0.85)

#speed between 10, 20
df1=df[(df['v_Vel'] >=10) & (df['v_Vel'] <20) & (df['v_Acc'] >0)& (df['v_Class'] ==2)]
a=df1['v_Acc']
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, 0.5,1,2,2.5,3,3.5,4]
plt.hist(a, bins= bin_edge)
plt.title('Vehicle Acceleration')
min_acc2 = df1.v_Acc.quantile(0.15)
mean_acc2 = df1.v_Acc.quantile(0.5)
max_acc2 = df1.v_Acc.quantile(0.85)

#speed between 20, 30
df1=df[(df['v_Vel'] >=20) & (df['v_Vel'] <30) & (df['v_Acc'] >0)& (df['v_Class'] ==2)]
a=df1['v_Acc']
```

```
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, 0.5,1,2,2.5,3,3.5,4]
plt.hist(a, bins= bin_edge)
plt.title('Vehicle Acceleration')
min_acc3 = df1.v_Acc.quantile(0.15)
mean_acc3 = df1.v_Acc.quantile(0.5)
max_acc3 = df1.v_Acc.quantile(0.85)

#speed between 30, 40
df1=df[(df['v_Vel'] >=30) & (df['v_Vel'] <40) & (df['v_Acc'] >0)& (df['v_Class'] ==2)]
a=df1['v_Acc']
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, 0.5,1,2,2.5,3,3.5,4]
plt.hist(a, bins= bin_edge)
plt.title('Vehicle Acceleration')
min_acc4 = df1.v_Acc.quantile(0.15)
mean_acc4 = df1.v_Acc.quantile(0.5)
max_acc4 = df1.v_Acc.quantile(0.85)

#speed between 40, 50
df1=df[(df['v_Vel'] >=40) & (df['v_Vel'] <50) & (df['v_Acc'] >0)& (df['v_Class'] ==2)]
a=df1['v_Acc']
import matplotlib.pyplot as plt
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, 0.5,1,2,2.5,3,3.5,4]
plt.hist(a, bins= bin_edge)
plt.title('Vehicle Acceleration')
```

```
min_acc5 = df1.v_Acc.quantile(0.15)
mean_acc5 = df1.v_Acc.quantile(0.5)
max_acc5 = df1.v_Acc.quantile(0.85)

#speed between 50, 60
df1=df[(df['v_Vel'] >=50) & (df['v_Vel'] <60) & (df['v_Acc'] >0)& (df['v_Class'] ==2)]
a=df1['v_Acc']
import matplotlib.pyplot as plt
plt.figure(5)
plt.xlabel('Acceleration (m/s^2)', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.ylim((0,20000))
plt.grid(True)
bin_edge= [0, 0.5,1,2,2.5,3,3.5,4]
plt.hist(a, bins= bin_edge)
plt.title('Vehicle Acceleration')
min_acc6 = df1.v_Acc.quantile(0.15)
mean_acc6 = df1.v_Acc.quantile(0.5)
max_acc6 = df1.v_Acc.quantile(0.85)

max_acc=[max_acc1, max_acc2, max_acc3, max_acc4, max_acc5, max_acc6]
med_acc=[mean_acc1, mean_acc2, mean_acc3, mean_acc4, mean_acc5, mean_acc6]
min_acc=[min_acc1, min_acc2, min_acc3, min_acc4, min_acc5, min_acc6]
x=[5, 15, 25, 35, 45, 55]

import matplotlib.pyplot as plt
fig=plt.figure()
ax1 = fig.add_subplot(111)
ax1.plot(x, max_acc, c='b', marker="s", label='Maximum')
ax1.plot(x, med_acc, c='r', marker="s", label='Median')
ax1.plot(x, min_acc, c='g', marker="s", label='Minimum')
plt.legend(bbox_to_anchor=(1.05, 1), loc='best');
plt.xlabel('Speed (km/h)', fontsize=12)
plt.ylabel('Acceleration (m/s2)', fontsize=12)
plt.title('Desired Acceleration')
plt.ylim((0.0,4.0))
plt.xlim((0,60))
ax1.yaxis.grid()

import statsmodels.api as sm
from sklearn.metrics import r2_score
```

```
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, 0.6, 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, 0.2, text,transform=plt.gca().transAxes,
    fontsize=14, verticalalignment='top')
print(est2.summary())
```

3: Desired Deceleration

```
import pandas as pd
import numpy as np

df = pd.read_csv("NGSIM_Peachtree_Vehicle_Trajectories.csv")

#speed between 0, 10
df1=df[(df['v_Vel'] >0) & (df['v_Vel'] <10) & (df['v_Acc'] <0)& (df['v_Class'] ==2)]
min_acc1 = df1.v_Acc.quantile(0.95)
mean_acc1 = df1.v_Acc.quantile(0.5)
max_acc1 = df1.v_Acc.quantile(0.05)

#speed between 10, 20
df1=df[(df['v_Vel'] >=10) & (df['v_Vel'] <20) & (df['v_Acc'] <0)& (df['v_Class'] ==2)]
min_acc2 = df1.v_Acc.quantile(0.95)
mean_acc2 = df1.v_Acc.quantile(0.5)
max_acc2 = df1.v_Acc.quantile(0.05)

#speed between 20, 30
df1=df[(df['v_Vel'] >=20) & (df['v_Vel'] <30) & (df['v_Acc'] <0)& (df['v_Class'] ==2)]
min_acc3 = df1.v_Acc.quantile(0.95)
mean_acc3 = df1.v_Acc.quantile(0.5)
max_acc3 = df1.v_Acc.quantile(0.05)

#speed between 30, 40
df1=df[(df['v_Vel'] >=30) & (df['v_Vel'] <40) & (df['v_Acc'] <0)& (df['v_Class'] ==2)]
min_acc4 = df1.v_Acc.quantile(0.95)
mean_acc4 = df1.v_Acc.quantile(0.5)
max_acc4 = df1.v_Acc.quantile(0.05)

#speed between 40, 50
df1=df[(df['v_Vel'] >=40) & (df['v_Vel'] <50) & (df['v_Acc'] <0)& (df['v_Class'] ==2)]
min_acc5 = df1.v_Acc.quantile(0.95)
mean_acc5 = df1.v_Acc.quantile(0.5)
max_acc5 = df1.v_Acc.quantile(0.05)

#speed between 50, 60
df1=df[(df['v_Vel'] >=50) & (df['v_Vel'] <60) & (df['v_Acc'] <0)& (df['v_Class'] ==2)]
min_acc6 = df1.v_Acc.quantile(0.95)
mean_acc6 = df1.v_Acc.quantile(0.5)
max_acc6 = df1.v_Acc.quantile(0.05)
```

```
max_acc=[abs(max_acc1), abs(max_acc2), abs(max_acc3), abs(max_acc4), abs(max_acc5), abs(max_acc6)]
med_acc=[abs(mean_acc1), abs(mean_acc2), abs(mean_acc3), abs(mean_acc4), abs(mean_acc5), abs(mean_acc6)]
min_acc=[abs(min_acc1), abs(min_acc2), abs(min_acc3), abs(min_acc4), abs(min_acc5), abs(min_acc6)]
x=[5, 15, 25, 35, 45, 55]

import matplotlib.pyplot as plt
fig=plt.figure()
ax1 = fig.add_subplot(111)
ax1.plot(x, max_acc, c='b', marker="s", label='Maximum')
ax1.plot(x, med_acc, c='r', marker="s", label='Median')
ax1.plot(x, min_acc, c='g', marker="s", label='Minimum')
plt.legend(bbox_to_anchor=(1.05, 1), loc='best');
plt.xlabel('Speed (km/h)', fontsize=12)
plt.ylabel('Deceleration (m/s2)', fontsize=12)
plt.title('Desired Deceleration')
plt.ylim((0.0,4.0))
plt.xlim((0,60))
ax1.yaxis.grid()

import statsmodels.api as sm
from sklearn.metrics import r2_score
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, 0.6, 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, 0.18, text,transform=plt.gca().transAxes,
    fontsize=14, verticalalignment='top')
print(est2.summary())
```

4: Standstill Distance

```
import pandas as pd
import numpy as np

df = pd.read_csv("NGSIM_Peachtree_Vehicle_Trajectories.csv")
df0 = df[(df['Space_Headway'] !=0) & (df['Preceding']!=0) & (df['v_Vel']==0) & (df['v_Acc']==0) ]

#Sort the database to generate preceding vehicle data for each vehicle record in the original database one by one
df1=df0.reset_index()
index = df1.index
number_of_rows = len(index)
i=0
df3= pd.read_csv("preceding_vehicle_records.csv")
df5= pd.read_csv("following_vehicle_records.csv")
for i in range (0, number_of_rows-1):
   vehicle_i = df1['Vehicle_ID'].values[i]
   preceding_vehicle_i = df1['Preceding'].values[i]
   timestep_i = df1['Global_Time'].values[i]
   direction = df1['Direction'].values[i]
   #df2 is a temporary df to store preceding vehicle data
   df2 = df[(df['Global_Time'] == timestep_i) & (df['Vehicle_ID'] == preceding_vehicle_i) & (df['Following'] ==
vehicle_i) & (df['Direction'] == direction) & (df['v_Vel'] == 0) & (df['v_Acc']==0)]
   if len(df2) != 1:
      df2 = []
   else:
   #df4 is a temporary df to store following vehicle data
      df4=df1.iloc[i]
      df3 = df3.append(df2) #preceding
      df5 = df5.append(df4) #following
      continue
   continue

df6=df3.reset_index() #preceding
df7=df5.reset_index() #following
headway=df7['Space_Headway']
v_length=df6['v_length']
distance=headway-v_length
Vehicle_ID= df7[‘Vehicle_ID’]
df8 = pd.DataFrame({'v_length’: v_length, 'distance': distance})
df9=df8.loc[df8.groupby('Vehicle_ID')['distance'].idxmin()]
```

```
df10 = df9[(df9['distance']>0) & (df9['distance']<= 10)]
distance=df10['distance']
average_standstill_distance=statistics. mean(distance)

import matplotlib.pyplot as plt
#fig=plt.scatter(speed_d,distance)
plt.xlabel('ΔX (m)', fontsize=12)
plt.ylabel('Number of Vehicles', fontsize=12)
#plt.ylim((0.0,10))
plt.grid(True)
bin_edge= [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10]
plt.hist(distance, bins= bin_edge)
```

5: Following Distance

```
df = pd.read_csv("NGSIM_Peachtree_Vehicle_Trajectories.csv")
df0 = df[(df['Space_Headway'] < 10) & (df['Space_Headway'] !=0) & (df['Preceding']!=0)& (df['v_Vel']==0)]
#Sort the database to generate preceding vehicle data for each vehicle record in the original database one by one
df1=df0.reset_index()
index = df1.index
number_of_rows = len(index)
i=0
df3= pd.read_csv("preceding_vehicle_records.csv")
for i in range (0, number_of_rows-1):
   #speed=df['v_Vel'].values[i]
   #speed_preceding=df['v_Vel'].values[i]
   vehicle_i = df1['Vehicle_ID'].values[i]
   preceding_vehicle_i = df1['Preceding'].values[i]
   timestep_i = df1['Global_Time'].values[i]
   direction = df1['Direction'].values[i]
   #df1 is a temporary df to store preceding vehicle data
   df2 = df[(df['Global_Time'] == timestep_i) & (df['Vehicle_ID'] == preceding_vehicle_i) & (df['Following'] ==
vehicle_i)&(df['Direction'] == direction)]
   if len(df2) != 1:
      update_df1 =df1.drop([i])
      df2 = []
   else:
      df3 = df3.append(df2)
      continue
   continue

#distance (head to bumper) between two vehicle
df4=df1.reset_index()
df5=df3.reset_index()
headway=df4['Space_Headway']
v_length=df5['v_length']
distance=headway-v_length
#speed difference
speed1=df4['v_Vel']
speed2=df5['v_Vel']
speed_d=speed1-speed2
df4.to_csv('following2.csv')
df5.to_csv('preceding2.csv')

import matplotlib.pyplot as plt
```

```
fig=plt.scatter(speed_d,distance)
plt.xlabel('ΔV', fontsize=12)
plt.ylabel('ΔX', fontsize=12)
plt.ylim((0.0,10))
plt.grid(True)
plt.hist(distance, bins=6)
```

```
import pandas as pd
import numpy as np
import statistics
df1 = pd.read_csv("following2.csv")
df2 = pd.read_csv("preceding2.csv")
#distance (head to bumper) between two vehicle
headway=df1['Space_Headway']
v_length=df2['v_length']
distance=headway-v_length
#speed difference
speed1=df1['v_Vel']
speed2=df2['v_Vel']
acc=df1['v_Acc']
dv=speed1-speed2
v_id=df1['Vehicle_ID']
data = {'Vehicle_ID':v_id, 'speed_follow':speed1, 'speed_preced':speed2, 'speed_diff':dv, 'distance':distance,
'acceleration': acc}
df = pd.DataFrame(data)
df=df[(df['distance'] > 2.8) & (df['distance'] < 100)]
dfmin=df.loc[df.groupby('Vehicle_ID')['distance'].idxmin()]
df['pctile']=df.groupby('Vehicle_ID')['distance'].rank(pct=True)
df['difference']=df['pctile']-0.1
df['difference']=df['difference'].abs()
dfmin=df.loc[df.groupby('Vehicle_ID')['difference'].idxmin()]
dfmin=dfmin[(dfmin['difference'] <0.2) & (dfmin['distance'] < 10)]
v=dfmin['speed_follow']/3.6
d=dfmin['distance']
ax=2.8
parameter=(d-ax)/np.sqrt(v)
mean=statistics. mean(parameter)
std=statistics. stdev(parameter)
#linear programing
# import the library pulp as p
import pulp as p

# Create a LP Minimization problem
Lp_prob = p.LpProblem('Problem', p.LpMinimize)

# Create problem Variables
x = p.LpVariable("x", lowBound = 1, upBound=5)   # Create a variable x >= 0 bxadd
y = p.LpVariable("y", lowBound = 1, upBound=6)   # Create a variable y >= 0 bxmult
```

```
# Objective Function
Lp_prob += std - 0.15 * y

# Constraints:
Lp_prob += x + 0.5 * y == mean
Lp_prob += x >= 1
Lp_prob += x <= 5
Lp_prob += y >= 1
Lp_prob += y <= 6
# Display the problem
print(Lp_prob)

status = Lp_prob.solve()   # Solver
print(p.LpStatus[status])   # The solution status

# Printing the final solution
print(p.value(x), p.value(y), p.value(Lp_prob.objective))
```

6: Lane Changing Vehicle Parameters

```
import pandas as pd
df = pd.read_csv("NGSIM_Peachtree_Vehicle_Trajectories.csv")
# Remove data with lane ID 0 or greater than 4, and no following vehicle
df1 = df[(df['Lane_ID'] > 0) & (df['Lane_ID'] <= 4)& (df['Following'] > 0)]
index = df1.index
number_of_rows = len(index)
i=0
#lane changing vehicle record
df2= pd.read_csv("lane_changing_vehicle_records.csv")
#lane trailing vehicle record
df3= pd.read_csv("trailing_vehicle_records.csv")
#Get lane change information
for i in range (0, number_of_rows-1):
#information of vehicle i
   vehicle_i = df1['Vehicle_ID'].values[i]
   lane_i = df1['Lane_ID'].values[i]
   i_new=i+1
#information of vehicle i_new
   vehicle_i_new=df1['Vehicle_ID'].values[i_new]
   lane_i_new = df1['Lane_ID'].values[i_new]
   if(vehicle_i != vehicle_i_new):
      continue
```

```
    else:
        if(lane_i == lane_i_new):
            continue
        else: df2.loc[df1.index[i_new]] = df1.iloc[i_new]
        continue
#get lane change data for 4s interval
df4= pd.read_csv("lane_changing_vehicle_records_4s.csv")
#append lane change data for 4s interval from df1 to df4 base on timestep and vehicle ID in df2
df2_number_of_rows=len(df2)
i=0
for i in range (0, df2_number_of_rows-1):
    vehicle_i = df2['Vehicle_ID'].values[i]
    timestep_i = df2['Global_Time'].values[i]
    #df5 is a temporary df to store data for each lane change movement
    df5 = df1[(df1['Global_Time'] <= timestep_i+1900) & (df1['Global_Time'] >= timestep_i-2000) &
(df1['Vehicle_ID'] == vehicle_i)]
    df5_number_of_rows=len(df5)
    if df5_number_of_rows ==40: #check if it has 40 frames (4s) data
        df4=df4.append(df5)
    continue
#Determine Max and desired deceleration
#Remove positive acc
df4_filtered = df4[df4['v_Acc'] <-0.5]
desired_dec = df4_filtered.v_Acc.quantile(0.85)
Max_dec = df4_filtered.v_Acc.quantile(0.15)
a=df4_filtered['v_Acc']
import matplotlib.pyplot as plt
plt.figure(0)
plt.xlabel('Deceleration (m/s^2)', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.ylim((0,1800))
plt.grid(True)
bin_edge= [-4, -3.5,-3,-2.5,-2,-1.5,-1,-0.5,0]
plt.hist(a, bins= bin_edge)
plt.title('Lane Changing Vehicle Deceleration')
```

7: Trailing Vehicle Parameters

```
import pandas as pd
import numpy as np
#Master database
df = pd.read_csv("NGSIM_Peachtree_Vehicle_Trajectories.csv")
```

```
# Remove data with lane ID 0 or greater than 4
df1 = df[(df['Lane_ID'] > 0) & (df['Lane_ID'] <= 4)]
index = df1.index
number_of_rows = len(index)
i=0
#lane changing vehicle record
df2= pd.read_csv("lane_changing_vehicle_records.csv")
#trailing vehicle record
df3= pd.read_csv("trailing_vehicle_records.csv")
#Get lane change information
for i in range (0, number_of_rows-1):
#information of vehicle i
   vehicle_i = df1['Vehicle_ID'].values[i]
   lane_i = df1['Lane_ID'].values[i]
   i_new=i+1
#information of vehicle i_new
   vehicle_i_new=df1['Vehicle_ID'].values[i_new]
   lane_i_new = df1['Lane_ID'].values[i_new]
   if(vehicle_i != vehicle_i_new):
      continue
   else:
      if(lane_i == lane_i_new):
        continue
      else: df2.loc[df1.index[i_new]] = df1.iloc[i_new]
      continue


#append trailing vehicle data from df1 to df6 base on timestep and vehicle ID in df2
df2_number_of_rows=len(df2)
i=0
for i in range (0, df2_number_of_rows-1):
   vehicle_i = df2['Vehicle_ID'].values[i]
   timestep_i = df2['Global_Time'].values[i]
   #df5 is a temporary df to store data for each lane change movement
   df5 = df1[(df1['Global_Time'] == timestep_i) & (df1['Preceding'] == vehicle_i)]
   df3 = df3.append(df5)
   continue
#get lane change data for 4s interval (trailing vehicle)
df6= pd.read_csv("trailing_vehicle_records_4s.csv")
#append trailing vehicle data for 4s interval from df1 to df6 base on timestep and vehicle ID in df2
df3_number_of_rows=len(df3)
i=0
```

```
for i in range (0, df3_number_of_rows-1):
    vehicle_i = df3['Vehicle_ID'].values[i]
    timestep_i = df3['Global_Time'].values[i]
    #df5 is a temporary df to store data for each lane change movement
    df5 = df1[(df1['Global_Time'] <= timestep_i+1900) & (df1['Global_Time'] >= timestep_i-2000) &
(df1['Vehicle_ID'] == vehicle_i)]
    df5_number_of_rows=len(df5)
    if df5_number_of_rows ==40: #check if it has 40 frames (4s) data
      df6=df6.append(df5)
    continue

# Minimum headway
df7=df6.iloc[39::40, :]
#lane changing vehicle record
df8= pd.read_csv("vehicle length.csv")
#Get headway
df7['Preceding_v_length'] = df7['Preceding'].map(df8.set_index('Preceding')['Preceding_v_length'])
df7['new_headway']=df7['Space_Headway']-df7['Preceding_v_length']
df7_filtered = df7[df7['new_headway'] >0]
min_headway=df7_filtered.new_headway.min()

# Safety distance reduction factor
#get lane change data for 2s interval (trailing vehicle)
df61= pd.read_csv("trailing_vehicle_records_4s.csv")
#append trailing vehicle data for 4s interval from df1 to df6 base on timestep and vehicle ID in df2
df3_number_of_rows=len(df3)
i=0
for i in range (0, df3_number_of_rows-1):
    vehicle_i = df3['Vehicle_ID'].values[i]
    timestep_i = df3['Global_Time'].values[i]
    #df5 is a temporary df to store data for each lane change movement
    df51 = df1[(df1['Global_Time'] < timestep_i) & (df1['Global_Time'] >= timestep_i-2000) & (df1['Vehicle_ID'] ==
vehicle_i)]
    df51_number_of_rows=len(df51)
    if df51_number_of_rows ==20: #check if it has 40 frames (4s) data
      df61=df61.append(df51)
    continue
df9=df61
#Get headway
df9['Preceding_v_length'] = df9['Preceding'].map(df8.set_index('Preceding')['Preceding_v_length'])
df9['new_headway']=df9['Space_Headway']-df9['Preceding_v_length']
```

```
n=20
df10=df9.groupby(np.arange(len(df9))//n)['new_headway'].max()
df11=df9.groupby(np.arange(len(df9))//n)['new_headway'].min()
df12=pd.concat([df10, df11], axis=1)
df12.columns = ['max', 'min']
df12=df12[(df12['max']>0) & (df12['max']<10) & (df12['min']>0)& (df12['min']<10)]
df12['ratio']=df12['min']/df12['max']
factor=df12['ratio'].mean()
#Determine Max and desired deceleration
#Remove positive acc
df6_filtered = df6[df6['v_Acc'] <-0.5]
desired_dec = df6_filtered.v_Acc.quantile(0.9)
Max_dec = df6_filtered.v_Acc.quantile(0.1)

a=df6_filtered['v_Acc']
import matplotlib.pyplot as plt
plt.figure(0)
plt.xlabel('Deceleration (m/s^2)', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.ylim((0,1800))
plt.grid(True)
bin_edge= [-4, -3.5,-3,-2.5,-2,-1.5,-1,-0.5,0]
plt.hist(a, bins= bin_edge)
plt.title('Trailing Vehicle Deceleration')

b=df7_filtered['new_headway']
plt.figure(1)
plt.xlabel('Headway after Lane Change', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.ylim((0,10))
plt.grid(True)
bin_edge= [0,1,2,3,4,5,6,7,8,9,10]
plt.hist(b, bins= bin_edge)
```

Appendix B – Python Scripts for Run VISSIM in COM Interface

```
import win32com.client as com
Vissim = com.Dispatch("Vissim.Vissim-64.200")
Filename = 'E:\\Final thesis\\NGSIM\\VISSIM\\Peachtree st default.inpx'
Vissim.loadNet(Filename)
Vissim.Graphics.CurrentNetworkWindow.SetAttValue('QuickMode',1)
# import format:
import csv
with open('inputs2.csv', 'r') as csvfile:
  driving_behavior_list = Vissim.Net.DrivingBehaviors.GetAll()
  myfile = csv.reader(csvfile, delimiter=',')
  input_variable_names = next(myfile)
  for k in range(800):
    this_line = next(myfile)
    Vissim.Net.DrivingBehaviors.GetAll()
    driving_behavior_list[0].SetAttValue("LookAheadDistMin", this_line[0])
    driving_behavior_list[0].SetAttValue("LookAheadDistMax", this_line[1])
    driving_behavior_list[0].SetAttValue("NumInteractObj", this_line[2])
    driving_behavior_list[0].SetAttValue("DecelRedDistOwn", this_line[3])
    driving_behavior_list[0].SetAttValue("DecelRedDistTrail", this_line[4])
    driving_behavior_list[0].SetAttValue("DiffusTm", this_line[5])
    driving_behavior_list[0].SetAttValue("W74ax", this_line[6])
    driving_behavior_list[0].SetAttValue("W74bxAdd", this_line[7])
    driving_behavior_list[0].SetAttValue("W74bxMult", this_line[8])
    driving_behavior_list[0].SetAttValue("MaxDecelOwn", this_line[9])
    driving_behavior_list[0].SetAttValue("MaxDecelTrail", this_line[10])
    driving_behavior_list[0].SetAttValue("AccDecelOwn", this_line[11])
    driving_behavior_list[0].SetAttValue("AccDecelTrail", this_line[12])
    driving_behavior_list[0].SetAttValue("MinFrontRearClear", this_line[13])
    driving_behavior_list[0].SetAttValue("SafDistFactLnChg", this_line[14])
    Vissim.Simulation.RunContinuous()
```

Appendix C – Python Scripts for Neural Network Calibration

```
# Create Dataset
import pandas as pd
inputs = pd.read_csv('inputs.csv', decimal=',', sep=',')
times = pd.read_excel('results.xlsx', decimal=',', sep=',', sheet_name='Time')
speeds = pd.read_excel('results.xlsx', decimal=',', sep=',', sheet_name='Speed')
times = times.pivot(index='SimRun', columns='VehicleTravelTimeMeasurement')
times = times.reset_index()
speeds = speeds.pivot(index='SimRun', columns='DataCollectionMeasurement')
speeds = speeds.reset_index()
dataset = pd.concat([times, speeds, inputs], axis=1)
dataset.columns = dataset.columns.map(str)
dataset.rename(columns='_'.join, inplace=True)
dataset.columns = dataset.columns.str.replace('[^a-zA-Z0-9]', '')
dataset = dataset.drop(columns='SimRun')
dataset = dataset.dropna()
dataset.to_csv('dataset_1.csv', decimal='.', sep=',', index=False)


# Neural Network Training
import matplotlib.pyplot as plt
import keras
import numpy as np
import pandas as pd
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
import pickle

layer_width = [50]

n_inputs = 10
n_outputs = 15

config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth = True
tf.compat.v1.keras.backend.set_session(tf.compat.v1.Session(config=config))

dataset = pd.read_csv('dataset_1.csv')

train_dataset = dataset.sample(frac=0.8, random_state=0)
test_dataset = dataset.drop(train_dataset.index)

train_stats = train_dataset.describe()
train_stats = train_stats.transpose()
train_stats.to_csv('train_stats1.csv')

def norm(x):
  return (x - train_stats['mean']) / train_stats['std']

normed_train_data = norm(train_dataset)
normed_test_data = norm(test_dataset)

train_data_as_numpy = normed_train_data.values
test_data_as_numpy = normed_test_data.values
```

```
x_train = train_data_as_numpy[:, 0:n_inputs]
y_train = train_data_as_numpy[:, n_inputs:]
x_test = test_data_as_numpy[:, 0:n_inputs]
y_test = test_data_as_numpy[:, n_inputs:]

for width in layer_width:
  model = Sequential()
  model.add(Dense(units=width, activation='sigmoid', input_dim=n_inputs))
  model.add(Dense(units=width, activation='sigmoid'))
  model.add(Dense(units=n_outputs, activation='sigmoid'))

  model.compile(loss='mean_squared_error', optimizer='Nadam', metrics=['mean_squared_error']
  early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=2)
  history = model.fit(x_train, y_train, validation_split=0.2, epochs=50000,callbacks=[early_stop], verbose=1)

  def plot_history(history):
    hist = pd.DataFrame(history.history)
    hist['epoch'] = history.epoch
    plt.figure()
    plt.xlabel('Epoch')
    plt.ylabel('Mean Square Error')
    plt.plot(hist['epoch'], hist['mean_squared_error'], label='Train Error')
    plt.plot(hist['epoch'], hist['val_mean_squared_error'], label='Val Error')
    plt.ylim([0, 1])
    plt.legend()
    plt.show()

  plot_history(history)
  hist = pd.DataFrame(history.history)
  with open('history.txt', 'wb') as file:
     pickle.dump(history.history, file)
  loss, mse = model.evaluate(x_test, y_test, verbose=0)
  outputs = model.predict(x_test, verbose=0)
  correlations = np.zeros(outputs.shape[1])
  for i in range(len(correlations)):
     correlations[i] = np.corrcoef(y_test[:, i], outputs[:, i])[0, 1]

  print(normed_test_data.columns.values[n_inputs:])
  print(correlations)
  model.save("calibration_nn.h5")
  del model

# Load NN
from keras.models import load_model
import keras
import pandas as pd
import numpy as np

new_inputs = pd.read_csv('new_inputs.csv', sep=',', decimal='.')
model = load_model('calibration12_nn.h5')
train_stats = pd.read_csv('train_stats12.csv', index_col=0, sep=',', decimal='.')
input_stats = train_stats.drop(index=['W74ax', 'W74bxAdd', 'W74bxMult', 'MaxDecelOwn', 'MaxDecelTrail',
'AccDecelOwn', 'AccDecelTrail', 'MinFrontRearClear', 'SafDistFactLnChg','LookAheadDistMin','LookAheadDistMax',
'NumInteractObj',  'DecelRedDistOwn', 'DecelRedDistTrail','DiffusTm'])
#input_stats = train_stats.drop(index=['Speedlowerbound', 'Speedhigherbound','W74ax', 'W74bxAdd', 'W74bxMult',
'MaxDecelOwn', 'MaxDecelTrail', 'AccDecelOwn', 'AccDecelTrail', 'MinFrontRearClear',
'SafDistFactLnChg','LookAheadDistMin','LookAheadDistMax',  'NumInteractObj',  'DecelRedDistOwn',
'DecelRedDistTrail','DiffusTm'])
#input_stats = train_stats.drop(index=['LookAheadDistMin','LookAheadDistMax',  'NumInteractObj',
'DecelRedDistOwn', 'DecelRedDistTrail','DiffusTm'])
def norm(x):
  return (x - input_stats['mean']) / input_stats['std']
```

```
normed_new_inputs = norm(new_inputs)
x_numpy = normed_new_inputs.values
y_numpy = model.predict(x_numpy)
new_outputs = pd.DataFrame(y_numpy, columns=['W74ax', 'W74bxAdd', 'W74bxMult', 'MaxDecelOwn',
'MaxDecelTrail', 'AccDecelOwn', 'AccDecelTrail', 'MinFrontRearClear',
'SafDistFactLnChg','LookAheadDistMin','LookAheadDistMax',  'NumInteractObj',  'DecelRedDistOwn',
'DecelRedDistTrail','DiffusTm'])
output_stats = train_stats.loc[['W74ax', 'W74bxAdd', 'W74bxMult', 'MaxDecelOwn', 'MaxDecelTrail', 'AccDecelOwn',
'AccDecelTrail', 'MinFrontRearClear', 'SafDistFactLnChg','LookAheadDistMin','LookAheadDistMax',
'NumInteractObj',  'DecelRedDistOwn', 'DecelRedDistTrail','DiffusTm']]
#new_outputs = pd.DataFrame(y_numpy, columns=['Speedlowerbound', 'Speedhigherbound','W74ax', 'W74bxAdd',
'W74bxMult', 'MaxDecelOwn', 'MaxDecelTrail', 'AccDecelOwn', 'AccDecelTrail', 'MinFrontRearClear',
'SafDistFactLnChg','LookAheadDistMin','LookAheadDistMax',  'NumInteractObj',  'DecelRedDistOwn',
'DecelRedDistTrail','DiffusTm'])
#output_stats = train_stats.loc[['Speedlowerbound', 'Speedhigherbound','W74ax', 'W74bxAdd', 'W74bxMult',
'MaxDecelOwn', 'MaxDecelTrail', 'AccDecelOwn', 'AccDecelTrail', 'MinFrontRearClear',
'SafDistFactLnChg','LookAheadDistMin','LookAheadDistMax',  'NumInteractObj',  'DecelRedDistOwn',
'DecelRedDistTrail','DiffusTm']]
#new_outputs = pd.DataFrame(y_numpy, columns=['LookAheadDistMin','LookAheadDistMax',  'NumInteractObj',
'DecelRedDistOwn', 'DecelRedDistTrail','DiffusTm'])
#output_stats = train_stats.loc[['LookAheadDistMin','LookAheadDistMax',  'NumInteractObj',  'DecelRedDistOwn',
'DecelRedDistTrail','DiffusTm']]
def denorm(y):
  return (y * output_stats['std']) + output_stats['mean']

new_outputs = denorm(new_outputs)
new_outputs.to_csv("new_outputs12.csv", index=False)

# Re-run VISSIM with Calibrated Parameters

import win32com.client as com
Vissim = com.Dispatch("Vissim.Vissim-64.200")
Filename = 'E:\\Final thesis\\NGSIM\\VISSIM\\Peachtree st default.inpx'
Vissim.loadNet(Filename)
import csv
with open('new_outputs4.csv', 'r') as csvfile:
  driving_behavior_list = Vissim.Net.DrivingBehaviors.GetAll()
  myfile = csv.reader(csvfile, delimiter=',')
  input_variable_names = next(myfile)
  for k in range(1):
    this_line = next(myfile)

    driving_behavior_list[0].SetAttValue("W74ax", this_line[0])
    driving_behavior_list[0].SetAttValue("W74bxAdd", this_line[1])
    driving_behavior_list[0].SetAttValue("W74bxMult", this_line[2])
    driving_behavior_list[0].SetAttValue("MaxDecelOwn", this_line[3])
    driving_behavior_list[0].SetAttValue("MaxDecelTrail", this_line[4])
    driving_behavior_list[0].SetAttValue("AccDecelOwn", this_line[5])
    driving_behavior_list[0].SetAttValue("AccDecelTrail", this_line[6])
    driving_behavior_list[0].SetAttValue("MinFrontRearClear", this_line[7])
    driving_behavior_list[0].SetAttValue("SafDistFactLnChg", this_line[8])
    driving_behavior_list[0].SetAttValue("LookAheadDistMin", this_line[9])
    driving_behavior_list[0].SetAttValue("LookAheadDistMax", this_line[10])
    driving_behavior_list[0].SetAttValue("NumInteractObj", this_line[11])
    driving_behavior_list[0].SetAttValue("DecelRedDistOwn", this_line[12])
    driving_behavior_list[0].SetAttValue("DecelRedDistTrail", this_line[13])
    driving_behavior_list[0].SetAttValue("DiffusTm", this_line[14])
    Vissim.Simulation.RunContinuous()
```