C2SMART
CONNECTED CITIES WITH SMART TRANSPORTATION

A USDOT University Transportation Center

New York University

Rutgers University

University of Washington

The University of Texas at El Paso

City College of New York

A Multiscale Simulation Platform for Connected and Automated Transportation Systems

August 2022

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2.Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle A Multiscale Simulation Platform for Connected and Automated Transportation Systems		5. Report Date August 2022	
		6. Performing Organization Code	
7. Author(s) Jeff Ban, Ohay Angah, Yiran Zhang, Qiangqiang Guo		8. Performing Organization Report No.	
9. Performing Organization Name and Address Connected Cities for Smart Mobility towards Accessible and Resilient Transportation Center (C2SMART), 6 Metrotech Center, 4th Floor, NYU Tandon School of Engineering, Brooklyn, NY, 11201, United States		10. Work Unit No.	
		11. Contract or Grant No. 69A3551747119	
12. Sponsoring Agency Name and Address Office of the Assistant Secretary for Research and Technology University Transportation Centers Program U.S. Department of Transportation Washington, DC 20590		13. Type of Report and Period Covered Final Report, 3/1/2021 - 5/31/2022	
		14. Sponsoring Agency Code	
15. Supplementary Notes			
16. Abstract Traffic simulation is an important tool that can assist researchers, analysts, and policymakers in testing vehicle/traffic control algorithms, gaining insights into micro/macro traffic dynamics, and designing traffic management strategies. However, different implementations require different simulation scales, and no multiscale simulation platform satisfies all requirements. In this project, we proposed to establish a multiscale vehicle-traffic-demand (VTD) simulation platform for connected and automated transportation systems (CATS). This is particularly meant for the control and management of CATS with varying penetration rates of connected and automated vehicles (CAVs). We built a microscopic vehicle-in-the-loop (VIL) simulation platform, which used Unity 3D to simulate/visualize vehicle operations/dynamics and Simulation of Urban Mobility (SUMO) to simulate traffic flow dynamics.			
17. Key Words	18. Distribution Statement Public Access		
19. Security Classif (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No of Pages 69	22. Price

Form DOT F 1700.7 (8-69)

A Multiscale Simulation Platform for Connected and Automated Transportation Systems

Principal Investigator: Jeff Ban
University of Washington
0000-0003-3605-971X

Ohay Angah
University of Washington
0000-0002-8937-5022

Yiran Zhang
University of Washington
0000-0002-7392-8841

Qiangqiang Guo
University of Washington
0000-0002-6461-5886

C2SMART Center is a USDOT Tier 1 University Transportation Center taking on some of today's most pressing urban mobility challenges. Some of the areas C2SMART focuses on include:

Urban Mobility and Connected Citizens

Urban Analytics for Smart Cities

Resilient, Smart, & Secure Infrastructure

Disruptive Technologies and their impacts on transportation systems. Our aim is to develop innovative solutions to accelerate technology transfer from the research phase to the real world.

Unconventional Big Data Applications from field tests and non-traditional sensing technologies for decision-makers to address a wide range of urban mobility problems with the best information available.

Impactful Engagement overcoming institutional barriers to innovation to hear and meet the needs of city and state stakeholders, including government agencies, policy makers, the private sector, non-profit organizations, and entrepreneurs.

Forward-thinking Training and Development dedicated to training the workforce of tomorrow to deal with new mobility problems in ways that are not covered in existing transportation curricula.

Led by New York University's Tandon School of Engineering, **C2SMART** is a consortium of leading research universities, including Rutgersersity, University of Washington, the University of Texas at El Paso, and The City College of NY.

Visit c2smart.engineering.nyu.edu to learn more

C2SMART CONNECTED CITIES WITH SMART TRANSPORTATION

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

Acknowledgments

The project team would like to thank the C2SMART UTC for financial and administrative support.

Executive Summary

Traffic simulation is an important tool that can assist researchers, analysts, and policymakers in testing vehicle/traffic control algorithms, gaining insights into micro/macro traffic dynamics, and designing traffic management strategies. However, different implementations require different simulation scales, and no multiscale simulation platform satisfies all requirements. In this project, we proposed to establish a multiscale vehicle-traffic-demand (VTD) simulation platform for connected and automated transportation systems (CATS). This is particularly meant for the control and management of CATS with varying penetration rates of connected and automated vehicles (CAVs). We built a microscopic vehicle-in-the-loop (VIL) simulation platform, which used Unity 3D to simulate/visualize vehicle operations/dynamics and Simulation of Urban Mobility (SUMO) to simulate traffic flow dynamics [1].

The development of the multiscale VTD leveraged the existing platform and extended the platform to also simulate large-scale traffic flows. The VTD platform in this project added Multiagent Transport Simulation (MATSim) to the VIL platform. This addition led to tasks on integrating traffic simulation models at different simulating scales (i.e., MATSim, SUMO, and Unity); communicating among MATSim, SUMO, Unity, and the Amazon Web Services (AWS) DeepRacer; smoothing all processes in the platform; and efficiently building a larger traffic environment in Unity. In this VTD platform, MATSim was expected to model traffic flows moving across traffic links and therefore was assigned a larger traffic region than SUMO, followed by Unity. In this regard, MATSim was given a network with a larger range than that given to SUMO. More specifically, we chose Greater Seattle to be MATSim's study area, while Downtown Seattle was the study area in SUMO. These two network systems were obtained from different sources; we therefore calibrated them individually.

This report describes our proposed approach to these tasks. Sections in this report address (i) network set-up and network calibrations for the two obtained networks; (ii) design of the multiscale simulation platform VTD; (iii) development and implementation of the multiscale simulation platform VTD; and (iv) discussion, conclusions, and possible directions for future work. In the first task, we collected the network data sets, traffic speed data, and traffic volume data from the City of Seattle, the City of Bellevue, INRIX, and the Puget Sound Regional Council (PSRC). The collected networks were then calibrated based on observed speeds and volumes. The second task focused on the integration of MATSim and SUMO, as well as communication among SUMO, Unity, and DeepRacer. This task was further broken down into integration between SUMO and MATSim, and communication among SUMO, Unity, and DeepRacer. The third task was to complete integration and communication based on the design and to ensure that the MATSim-SUMO integration was compatible. We then implemented the platform for the Greater Seattle area.

After platform development and implementation, we also proposed several research tasks to utilize the platform to better understand traffic dynamics and improve traffic performance. These included traffic dynamics learning and traffic signal and vehicle control.

Table of Contents

List of Figures

List of Tables

Section 1 Research Background

Traffic simulation is becoming an increasingly important tool for assisting researchers, analysts, and policymakers in testing vehicle/traffic control algorithms, gaining insights into micro/macro traffic dynamics, and designing traffic management strategies. Specific traffic/transportation applications may require a proper simulation at the scale of a corridor, a small network, a city/region network, etc. Although any of the specific scales may lack perspective on the behaviors of the overall transportation system, there are currently few multiscale simulation platforms that can satisfy the requirements of all the scales. In this project, we proposed a framework for establishing a multiscale vehicle-traffic-demand (VTD) simulation platform. This platform is general and can be particularly tailored for the control and management of connected and automated transportation systems (CATS) with varying penetration rates of connected and automated vehicles (CAVs).

There are currently multiple traffic simulation software tools for transportation-related studies. Here we briefly discuss some of the latest and most widely used simulation platforms, including a popular macroscopic simulation model called Multi-Agent Transport Simulation (MATSim) [2], an emerging mesoscopic simulation model called DTALite [3], the popular microscopic simulation model called Vissim [4], a novel microscopic simulation model called A/B Street [5], and another popular open-source simulation model called Simulation of Urban MObility (SUMO) [6]. MATSim is an agent-based platform for large-scale traffic simulation. It is able to simulate daily traffic for a large region, but it cannot capture detailed, individual vehicle behaviors such as turning movements and accelerations. DTALite is an open-source, mesoscopic, dynamic traffic assignment (DTA) simulation package that provides a theoretically rigorous and computationally efficient traffic network modeling tool. However, users need a background in DTA and traffic flow theory to use the model, and the network visualization of DTALite is simple in comparison to that of Vissim, A/B Street, or SUMO. Vissim is a widely used, multimodal simulator that allows users to define different vehicle types, although it is not open source. A/B Street is an open-source, microscopic traffic simulation platform released last year. The platform has an impressive user interface for traffic networks. Nevertheless, the underlying traffic models in A/B Street may be limited. SUMO is an open-source software platform and has been widely used for transportation studies recently. However, users need to define many of the detailed traffic behaviors and dynamics, which requires deeper knowledge about traffic/transportation.

None of the above-discussed simulation platforms is multiscale. To explore ways to simulate traffic networks on different scales, some researchers [6, 7] have attempted to build a hybrid model that integrates different levels of detail. However, developing a new multiscale simulation platform from scratch is time- and resource-intensive, and it is often hard to capture both the macroscopic and microscopic features of traffic flow or demand patterns. Although other researchers have coupled different traffic simulation tools representing different levels of detail for multiscale simulation, such as

mesoscopic vs. microscopic [8], making multiple platforms compatible and implementing a communication system among multiple models have remained a challenging task. Additionally, a limited number of studies have explored traffic networks from a single vehicular level to a network level.

The research team built a microscopic vehicle-in-the-loop (VIL) simulation platform in 2020 [1], which uses Unity 3D to simulate/visualize vehicle operations/dynamics and SUMO to simulate traffic flow dynamics. Figure 1 shows an overview of the VIL simulation model. The VIL model applies the TCP/IP communication protocol to communicate and coordinate between vehicle simulation (Unity) and traffic simulation (SUMO), which can also help transfer data and information between the two. The model has been shown to be a useful virtual testbed for research, testing, and validation of vehicle control (e.g., eco-driving), traffic control, and coupled traffic-vehicle control [1]. The VTD simulation platform implemented in this project expands the VIL platform, and is expected to enable the team and other researchers to investigate and test/validate more integrated vehicle-traffic control models on larger areas. The VTD will also be able to facilitate concerted management strategies that can be simultaneously developed for and applied to a multiscale transportation system.

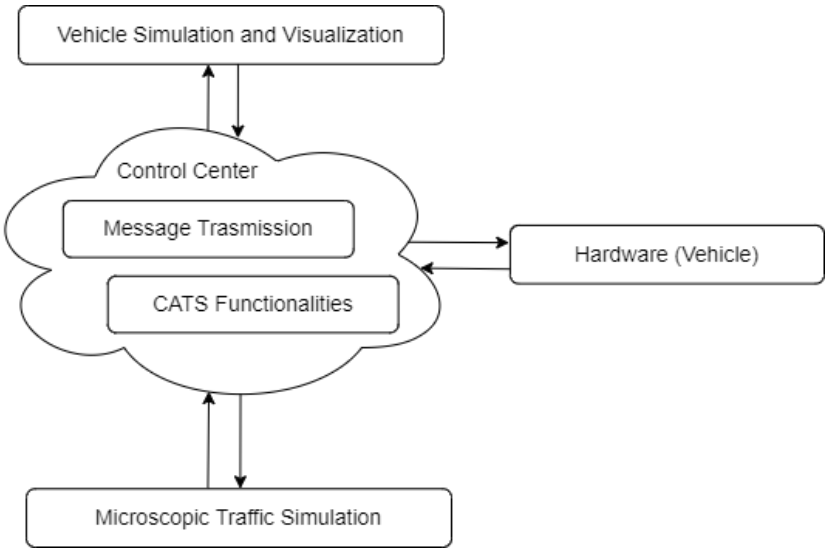


Figure 1: Overview of the Vehicle in the Loop (VIL) Simulation

Section 2 The Framework for the VTD Platform

The vehicle-traffic-demand (VTD) platform integrates MATSim, SUMO, and Unity. The overall framework is shown in Figure 2. The integration is divided into MATSim-SUMO and SUMO-Unity because of the distinct simulation frameworks inherent in the three models. MATSim models the 24-hour trip activities of all agents and demand at once. In contrast, SUMO models individual moving behaviors at each time step; Unity projects the moving behaviors at each time step on the corresponding objects in its environment. An individual agent's behavior can therefore be obtained at each time step in SUMO and Unity. The trip activities and demand in MATSim, on the other hand, can be yielded only after MATSim completes all the activity modeling. Therefore, integrations of MATSim-SUMO and SUMO-Unity have different designs, which are described as follows.

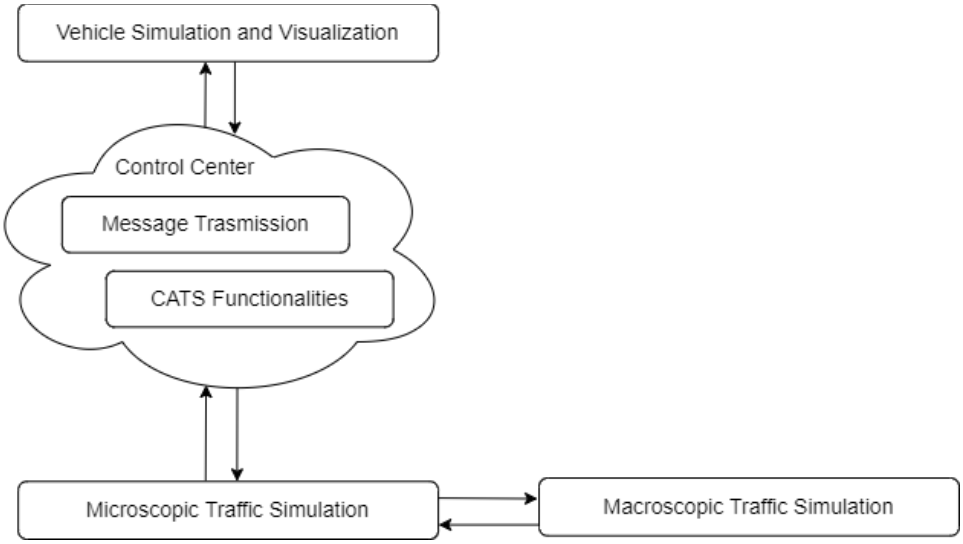


Figure 2: Overview of the VTD Simulation

Subsection 2.1: MATSim-SUMO Integration

The concept of MATSim-SUMO integration is to provide coarse information (e.g., demand) about the larger region while presenting traffic details (e.g., individual movements) from the local areas that are considered important. MATSim is designed to handle the former and SUMO the latter. At the MATSim-SUMO boundaries, the trip of each agent and the traffic network representation should be consistent. In this regard, we designed a trip integration process for trip consistency and a potential strategy for network connection at boundaries.

Subsection 2.1.1: Design of Trip Integration

Trip integration follows the concept discussed above in that we first gain a coarse view of traffic dynamics (e.g., traffic flow, link departure/arrival) and then observe the traffic details in several key local areas. In the design, MATSim provides the large-scale traffic dynamics because it is capable of collecting the link departure and arrival locations and times of each agent. The departure/arrival information allows us to extract when and where each agent entered and exited the local areas. According to the extracted information, each trip is divided into multiple trip segments categorized by traveling within the larger region or in the local areas. The trip segments assigned to the local areas then act as the travel input for the microscopic simulation by SUMO. To guarantee consistency of the trip scenarios in both simulation models, the departure times of the ongoing trips within the larger region for those agents just exiting local areas are updated and then become the input to MATSim for the final demand simulation.

The number of divided trips depends on the number of times an agent travels into the local areas. All the cases of trip division can be found in Table 1, in which we define the larger and the local areas by the respective abbreviations for MATSim ("M") and SUMO ("S"). Cases (i) and (ii) consider trips only in S and M, respectively. Other cases, in contrast, consider trips across both M and S. Suppose an agent travels into S *n* times, then the agent's trip in M will be divided into *n* trip segments if the origin and the destination include both M and S, i.e., cases (iii) and (vi). If the origin and the destination are only in M, i.e. case (iv), then the agent's trip in M will be divided into *n+1* trip segments. The divided trip segments in M, i.e. case (v) will be *n-1* if the agent's origin and the destination are both in S.

Table 1: Cases for Trip Division

Cases	Traveling Cases	The Number of Trips in M	The Number of Trips in S
1	S	0	1
2	M	1	0
3	M→S→M→…→S	*n*	*n*
4	M→S→M→…→M	*n+1*	*n*
5	S→M→S→…→S	*n-1*	*n*
6	S→M→S→…→M	*n*	*n*

In summary, the process discussed above requires two MATSim simulations and one SUMO simulation in between. Since the two simulations are different, outputs from the two MATSim simulations may also be different. Here we provide a simple example as follows. After running the first MATSim simulation, the result shows an agent travels from M (at 8:10 am) through S (8:20 to 8:30 am) and finishes the overall trip in M (at 8:40 am), i.e., case (iv) in Table 1 with n = 1. The overall trip will consequently be divided into two trips in MATSim and one trip in SUMO, as illustrated in Figure 3, in the following order: (i) trip #1 traveling in M (8:10 to 8:20 am), (ii) trip of traveling in S (8:20 to 8:30 am), and (iii) trip #2 traveling in M (8:30 to 8:40 am). These trips are later converted into agent plans as input for MATSim and a route demand as input for SUMO. At the moment, MATSim estimates the arrival time given to the route demand in SUMO is 8:30 am and the travel time in SUMO area is 10 min. The agent's movement will then be simulated in SUMO, and the travel time of the agent may be different from 10 min since SUMO considers more detailed traffic dynamics and congestion patterns. Assume SUMO simulation generates 12 min travel time for the agent, implying that the agent will finish the trip in SUMO and return to MATSim at 8:32 am. That is, in the second MATSim simulation, trip #2 in MATSim will start at 8:32 am instead of 8:30 am as MATSim initially estimated. In this case, the first MATSim simulation is just used to generate initial agent schedules to divide their trips if necessary. Results of the second MATSim simulation are used for analysis.

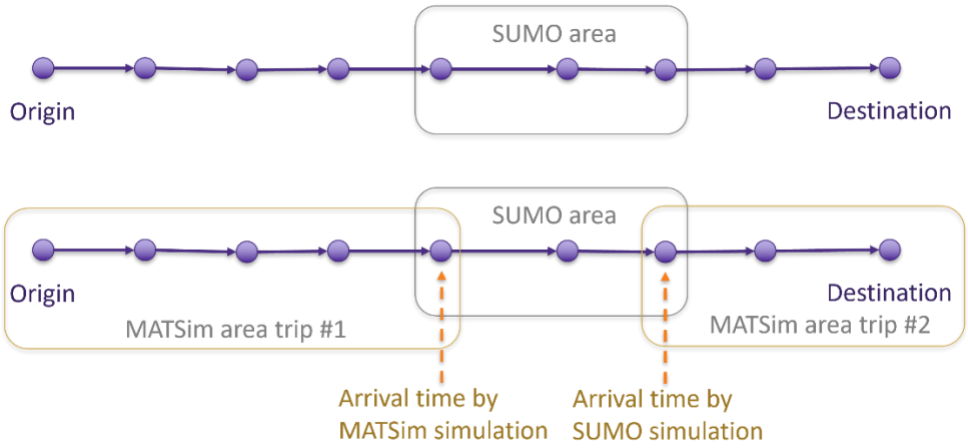


Figure 3: Design of Trip Integration

Subsection 2.1.2: Design of the Integration of Network Representations

The difficulty of making the networks used in MATSim and SUMO compatible varies depending on the complexity and sources of the networks. Indeed, the difficulty should be relatively low if one uses the same network in both simulation models. The number of links that SUMO covers, in this case, will be less than or equal to the number of links that MATSim covers. Because the network representations in

both simulation models are already compatible, integration of the network representations is no longer required no matter how complex the network is. However, the step is necessary if the networks used for the two simulation models are from different sources. One will have to compare the links of one network to the other. The comparison can be time-consuming if the networks are complex and, in particular, have many differences. In this case, matching the links at the boundaries of the two simulation models, whose networks have been calibrated, is a potential strategy for making the two networks compatible. The matching approach in our design is affected by the networks used in this project (see Section 3: MATSim Calibration), in which a road is usually represented by a link in SUMO but by a series of shorter links in MATSim. As illustrated in Figure 4, we split each of the boundary links in SUMO into multiple segments and compute the distance between each link segment and the center of the MATSim link. The least distance is then considered to be the distance between the SUMO link and the MATSim link. After distances between the MATSim link and all the SUMO links have been computed, e.g., the green and the yellow illustrated in this figure, we record the SUMO link that has the least distance as the one matched to the MATSim link. Those matched link pairs are later summarized in a matching list, which is used for simulating agents traveling from the SUMO to the MATSim areas and vice versa. Nevertheless, this approach may successfully match only a limited proportion of the links at the boundaries if the two networks are very different. In other words, manually matching the rest of the links and ensuring the validity of the matched link pairs may still be required.

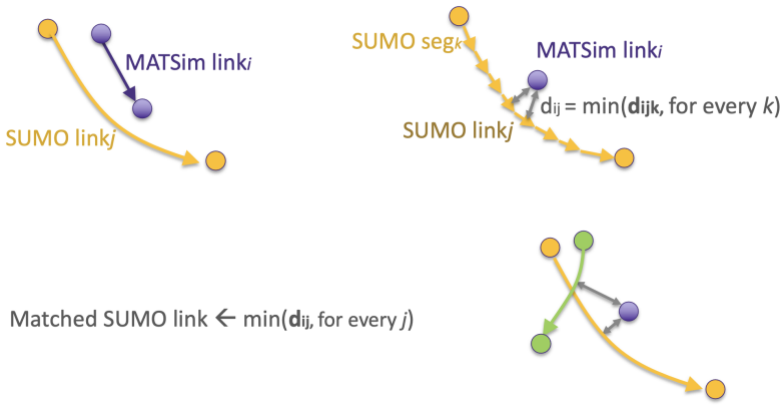


Figure 4: Link Matching Approach

Subsection 2.1.3: Development of the MATSim-SUMO Integration

The design of the MATSim-SUMO integration aims to achieve consistency in network representation and consistency in trips between the two models. For consistency in network representation, despite the potential strategy discussed in the previous subsection, one may encounter a series of issues related to different traffic network resolution levels between MATSim and SUMO. Because MATSim simulation is

usually used to observe traffic flow movements whereas SUMO simulation is used to model interactions of individual vehicles in detail, it is rational to only include major roads in MATSIM but also include roads at lower hierarchies (e.g., collector roads and minor roads) in the SUMO network. Such resolution differences can result in road disconnection betwen the SUMO areas and the MATSim area. Moreover, the differences can also create issues such as distributing traffic volumes from the MATSim area to the SUMO areas. Trip consistency requires ensuring consistency in the number of agents and departure/arrival times between the trips in the two simulation models. The number of agents from the MATSim area to the boundaries of each SUMO area should be equal to the number of agents traveling in the SUMO area. The time every agent from the MATSim area arrives at the boundaries of a SUMO area should be consistent with the agent's departure time traveling within the SUMO area. Likewise, the time every agent traveling within a SUMO area arrives at the boundary of the MATSim area should be consistent with the agent's departure time for traveling within the MATSim area. Given the above factors, we proposed the development framework shown in Figure 5 for MATSim-SUMO integration.

To begin with, as displayed in Figure 5, a road network of the larger region, agents, and trips of all agents are prepared for MATSim; the road networks of the local areas and the schedules of traffic signals are the inputs for SUMO. As mentioned above, the network prepared for MATSim considers only the top-hierarchy road system, i.e., Interstate freeways, state highways, major arterial roads, etc.; each of the networks for SUMO additionally considers collector roads and minor roads. All the networks used in the two models are already calibrated. According to the prepared networks, a matching list is then generated through the process of link matching. Meanwhile, the initial MATSim simulation can be launched, given the prepared network, agents, and their trip plans. After the simulation, each simulated trip is divided into multiple trips, if the agent entered the SUMO areas, based on the retrieved link entry times and the link matching list; see Subsection 2.1.1. The divided trips are subsequently sent to the next processor to check whether each trip in SUMO is valid. The route validity check is necessary if the SUMO network is disconnected at certain locations. An example can be found in Figure 6; the Downtown Seattle network is extracted from a larger area in OpenStreetMap (OSM). Given an origin-destination (OD) pair from node 1 to node 6, an agent could go through the path 1-2-3-4-5-6 in the MATSim network. However, in the Downtown Seattle area (the SUMO network), the agent cannot find a feasible path for the given OD pair since part of the MATSim path is not included in the SUMO network. Links that have this issue are mainly at the boundary of the SUMO network, i.e., in the highlighted red blocks on the left-hand side of the figure. In this case, before the SUMO simulation, the route validity check examines whether the OD pair of a trip is connected in SUMO. Trips with valid OD pairs are retained as the inputs for the SUMO simulation. The subsequent MATSim departure times are then updated on the basis of the arrival times simulated by SUMO. The updated trips and the trips without being split, i.e., those traveling only in the MATSim area, are combined as inputs into MATSim. Lastly, overall demand is finalized by running the MATSim simulation again with the updated plans.

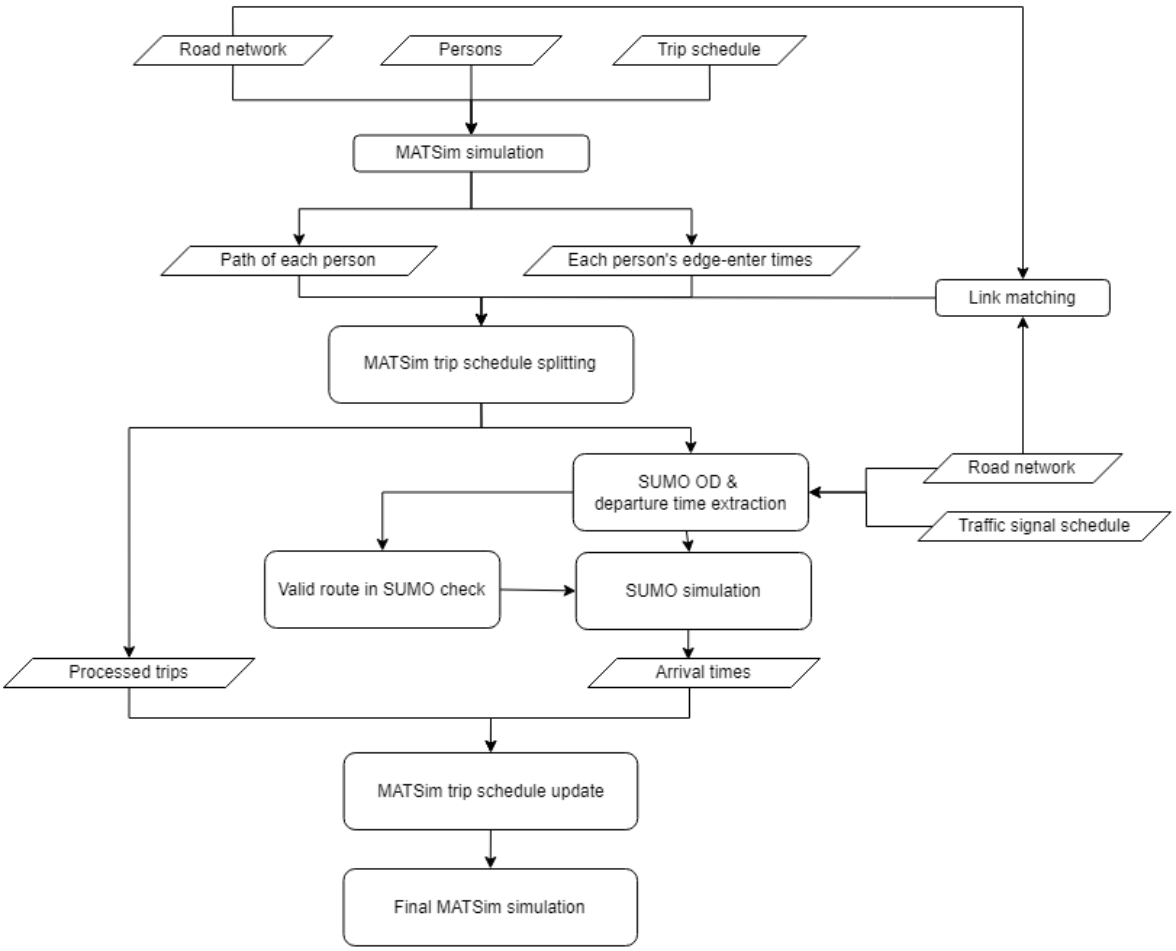


Figure 5: Development Pipeline of MATSim-SUMO Integration

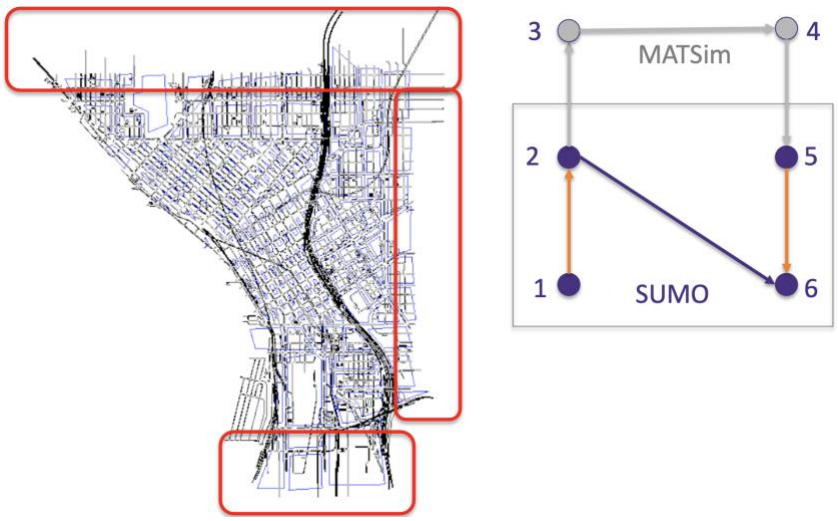


Figure 6: Disconnected Network

Subsection 2.2: SUMO-Unity Integration

As shown in Figure 1, the VIL simulation platform integrates a vehicle visualization model (Unity), a microscopic traffic simulation model (SUMO), and hardware. Message transfer between SUMO, Unity, and the hardware is managed by a control center, which also generates traffic control commands based on data gathered from the two models and the hardware. The SUMO-Unity integration in our VTD platform inherits the functionalities of the control center. Note that the hardware is not the main focus of this project and thus is omitted hereafter in this report.

Subsection 2.2.1: Design of the SUMO-Unity Integration

As just discussed, the design of communication between SUMO and Unity in our platform has the same design as that of the VIL platform [1]. The design is summarized in Figure 7. SUMO and Unity share their real-time information. On the one hand, the messages that are transmitted from SUMO to Unity can be categorized into static information and dynamic information. Static information is information determined before the SUMO simulation has been run, including the number of traffic signals, locations of traffic signals, and traffic network configuration. Dynamic information is updated during the simulation and contains traffic signal phases, ego-vehicle state, states of surrounding connected automated vehicles, states of surrounding human-driven vehicles, and states of other surrounding travelers. On the other hand, Unity sends information to remind SUMO whether the current information has been processed and whether it is up for processing the next time-step information.

The information transferred between SUMO and Unity is managed by the message transmission processor in the control center. Once the overall system has been launched, SUMO first waits for Unity

to be initialized until Unity sends an idle message to SUMO. SUMO then sends the traffic states—including traffic signal states, network information, and current states of the ego vehicle and its surrounding vehicles—to Unity. After receiving information from SUMO, Unity continues sending the message "cook" to SUMO when it is processing the visualization. Meanwhile, Unity simulates the ego vehicle sensing (e.g., camera and LiDAR) and sends the current sensed results to the control center for traffic control (e.g., signal control and vehicle-traffic control) at the incoming time step. Commands of signal control are then sent to SUMO to update the signal phases. Commands of vehicle-traffic control are also determined on the basis of the traffic states collected from SUMO. After traffic visualization and simulation have been completed, Unity then sends a message to SUMO to declare it is ready to process the updated traffic states. Next, SUMO updates the signal phases on the basis of the commands from the control center and surrounding vehicles' states from its simulation. The overall steps then iterate until the end of the simulation.

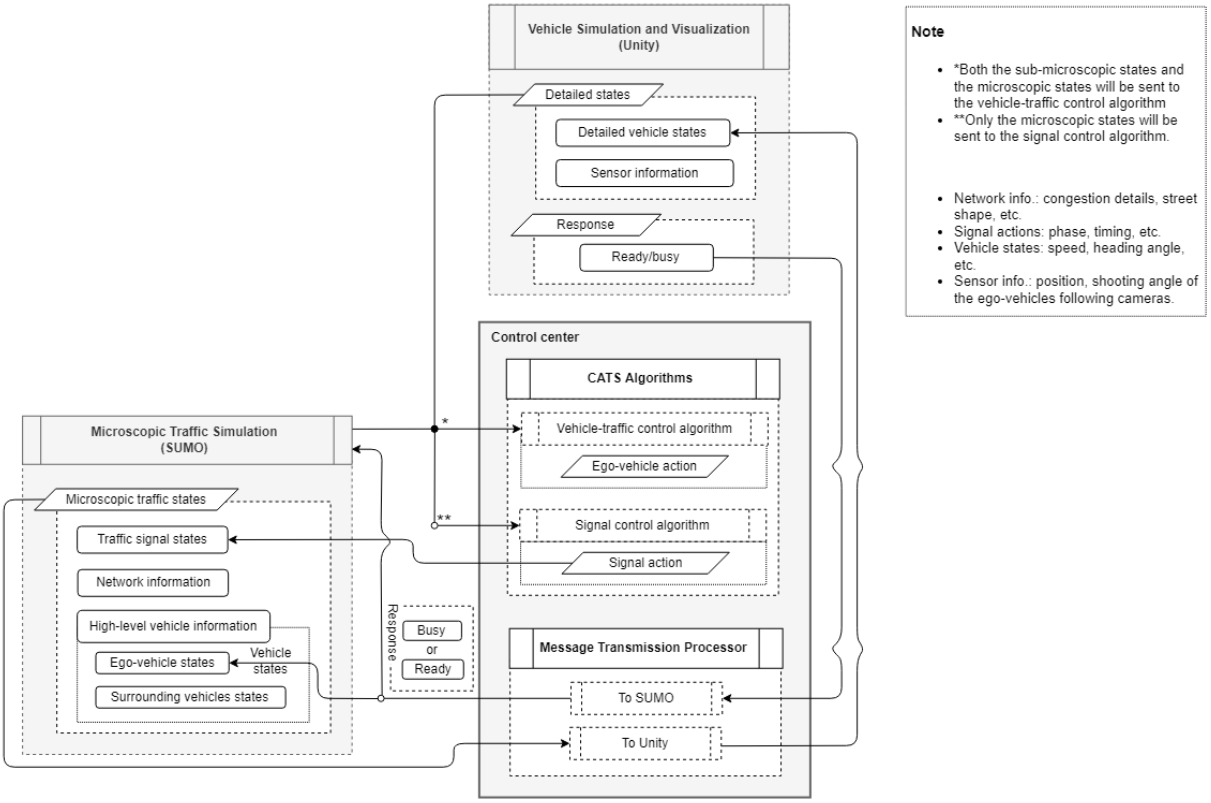


Figure 7: Detailed Design of SUMO-Unity Integration

Subsection 2.2.2: Development of the SUMO-Unity Integration

Different from the development process proposed Ban et al. [1], the environment in Unity in the VTD platform is predetermined. In the VIL platform [1], Unity changes its visualization environment as ego vehicles move forward. When an ego vehicle meets a new road section, Unity adds a new road section to the visualization environment. Meanwhile, Unity removes the most aged road section if the capacity for accommodating road objects is full to avoid out-of-memory issues. However, this strategy has two main drawbacks. First, the overall platform can run slowly while waiting for Unity to generate road sections. Second, for a study area that has many special road connections and road types, it can be time-consuming to design all the road modules and road connection types in advance. Therefore, we take a different strategy in this research when generating a traffic environment in the VTD platform. This strategy is to generate the environment before launching the platform.

The environment is generated by the following steps: (i) target a traffic area, (ii) divide the area into multiple sub-areas if the area is too big, (iii) generate and compress each sub-area's traffic environment, and (iv) combine all the areas into a base map. The purpose of dividing an area into multiple parts and generating a traffic environment one by one is to reduce memory usage while creating a traffic environment in each area. Compressing the generated traffic environment makes sure that the combined traffic environment will not consume too much memory. To generate a traffic environment, we apply the two packages RoadArchitect and CityGen3D in Unity. RoadArchitect is a package for creating road systems. It has a traffic signal system module and a road system module. The former includes options for traffic light poles, streetlights, and/or traffic lights coordinating with a controller to dynamically change traffic signal phases. The latter has road markings, right or left-turn markings, and one- to three-lane types. Nevertheless, the road system options are still limited to the traffic environment that has a complex road design, such as Downtown Seattle. This is where the CityGen3D package comes in. CityGen3D is a city generator that creates terrains, road systems, and amenities. It generates terrains and road systems directly from an OSM file, although it does not include traffic signal modules. Based on their capabilities, we create traffic signals using RoadArchitect while creating the basic terrains for vehicles and the road systems with CityGen3D. Figure 8 and Figure 9 show examples of generating an environment for the northern region of Downtown Seattle. The region is divided into four parts along Harrison Street and Westlake Avenue N. We generate each part of the environment in Unity and combine all the parts, resulting in the traffic environment displayed in Figure 9 (c).

Figure 8: An Example: Target Area in Downtown Seattle

(a) OSM (b) Environment in SUMO (c) Environment in Unity

Figure 9: Environments in OSM, SUMO, and Unity

Because the Unity environment in the VTD platform is created before simulations have been run, we propose an approach for Unity to process the received messages. As shown in Figure 10, after initialization, Unity sends a message to claim that it is ready to take a new message. Then it begins to receive messages. The message sent from SUMO has two information pieces: information about the current signal phases and information about current vehicles. Unity extracts and processes the information pieces one by one. To process signal phase information, Unity sorts each traffic signal in a

clockwise direction starting from from the north. According to the sorting outcome, Unity then assigns each phase to the corresponding traffic light. To update vehicle states, the vehicle information piece is further decomposed into existing vehicles, departed vehicles, and newly arrived vehicles. Unity deletes vehicles that have departed the traffic system, generates new vehicle objects for those that have just joined the traffic system and moves them to their corresponding locations, and in the meantime moves existing vehicles to their updated locations. Note that vehicles are not just moved to their updated locations but are steered toward the updated directions, and updated steering angles are also extracted from the vehicle information. In the VTD platform, we retain the camera tracking feature that always stays a certain distance away from the ego vehicle, i.e., a half vehicle-length away from the ego vehicle with a half vehicle-height above ground at ten degrees downward.

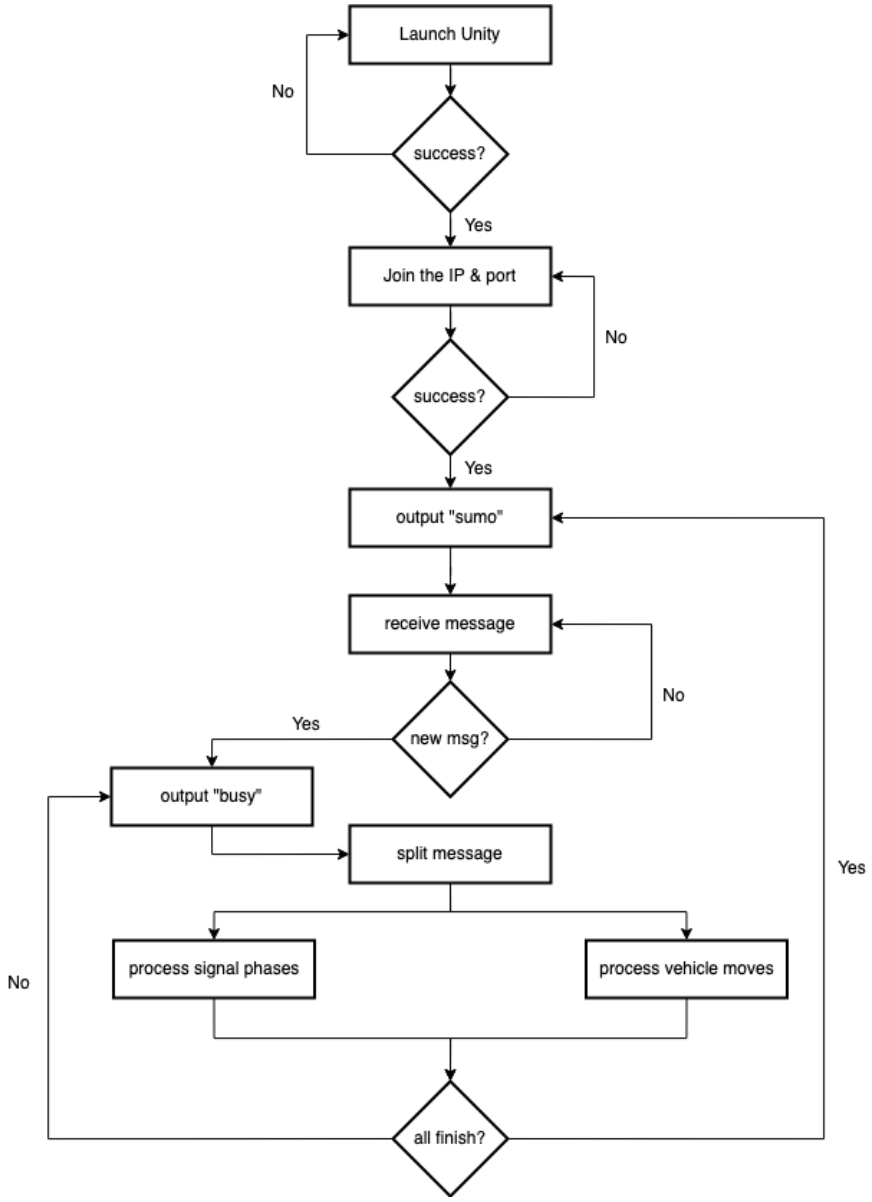


Figure 10: Diagram of the Used Processor in Unity

Figure 11 shows the implementation of the VTD platform environment. From left to right, it shows environments of the macroscopic traffic simulation (MATSim), the microscopic traffic simulation (SUMO), and the vehicle simulation and visualization (Unity). The Unity environment simulates a certain part of the SUMO environment, while SUMO simulates the Downtown Seattle region, which is a part of the MATSim environment. Following the implementation shown in Figure 11, Figure 12 shows five vehicles driving from the north to the Downtown Seattle region (SUMO area). SUMO simulates their movements in the Downtown Seattle area, as shown on the right-hand side. Two codes are presented in the SUMO simulation window shown in Figure 12. The bottom code refers to each vehicle's ID, while the

code (“1.00” or “0.00”) right above the vehicle’s ID indicates whether this vehicle is from the MATSim environment. “1.00” indicates that the vehicle comes from the MATSim region, “0.00” otherwise. Since vehicles tagged “0.00” are not from the MATSim region, we may treat them as background vehicles. Figure 13 displays a case in which several vehicles move from the MATSim environment to the SUMO environment, and then return to the MATSim environment again. We use an underscore and a number to count how many times the vehicle goes through the SUMO region. In this case, both the highlighted vehicles go through the SUMO region once.

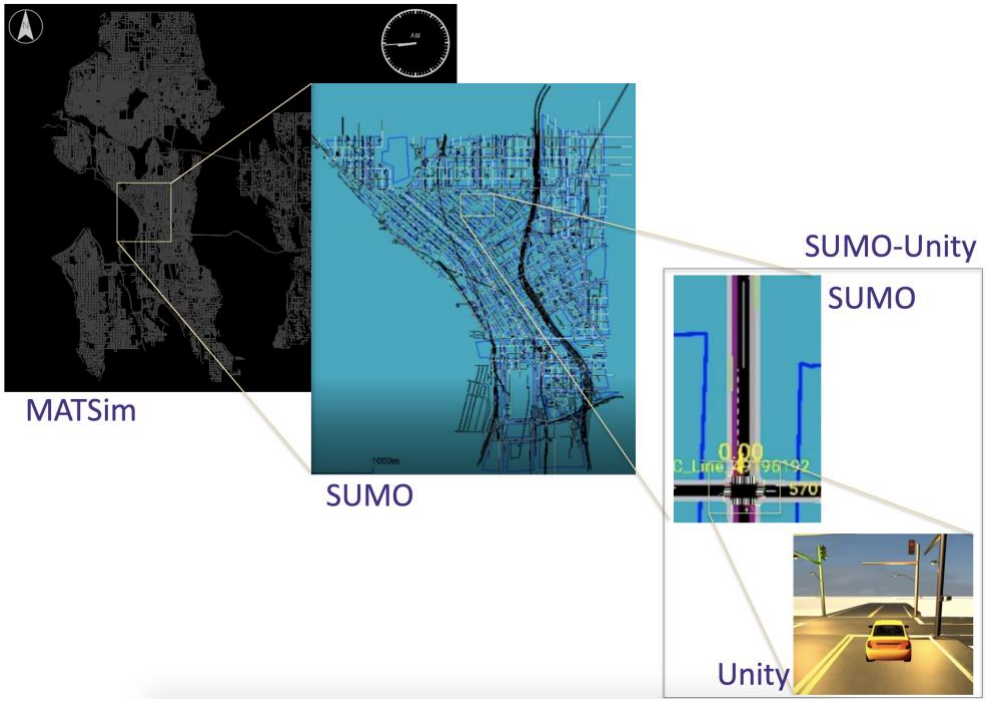


Figure 11: VTD Platform Environment

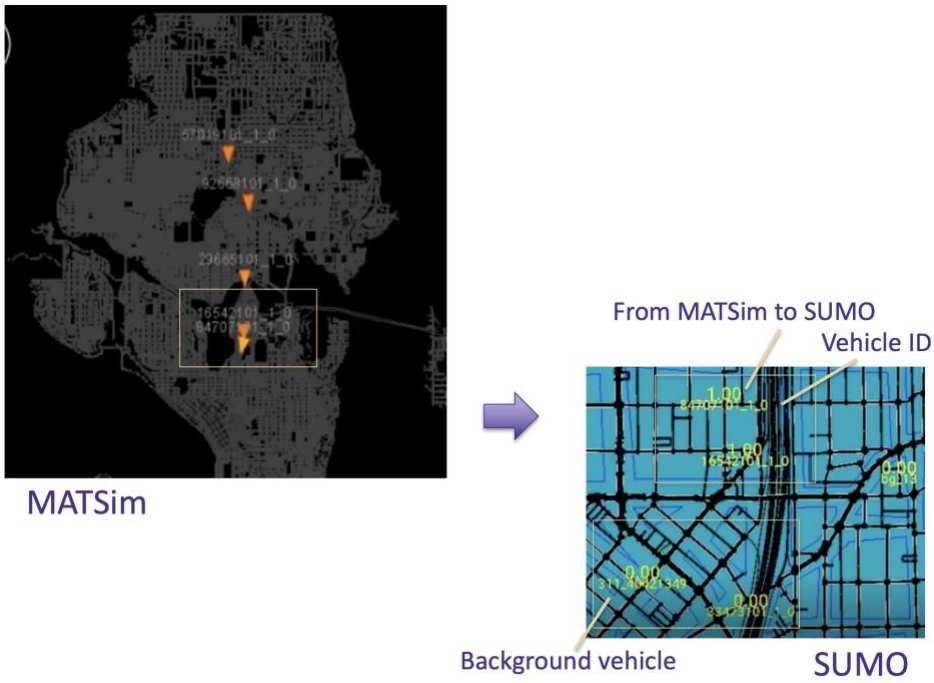


Figure 12: VTD Platform Implementation

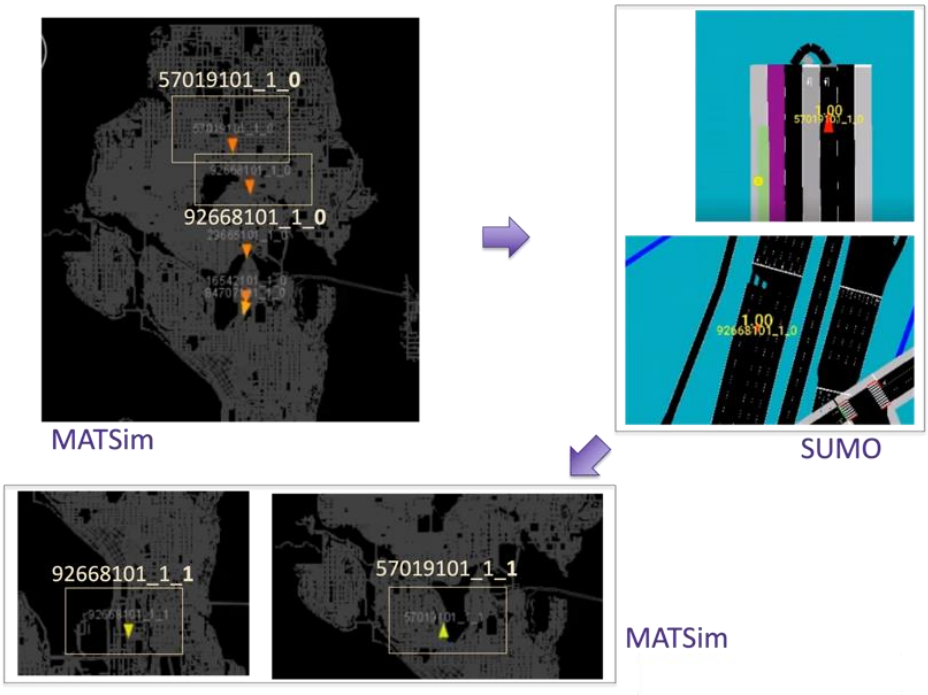


Figure 13: VTD Platform Implementation (Cont'd)

Section 3: MATSim Calibration

Subsection 3.1: Network Set-up

Our simulation area in MATSim contains Seattle and Bellevue, which are connected by State Route 520 (SR 520) and the Interstate 90 (I-90). We collected the network datasets, as shown in Figure 14, from the Seattle Department of Transportation (SDOT) and the Bellevue Department of Transportation, respectively. The two network datasets categorize road types in different ways, as listed in Table 2. We simplified the road types from their definitions into five categories: motorway, primary arterial, secondary arterial, tertiary arterial, and unclassified. The motorway contains the highway systems: the Interstate freeway and the state route freeway (Seattle), and the highway and highway ramp (Bellevue). The primary arterial contains the major road types of the two network systems: the principle arterial (Seattle) and the major arterial (Bellevue). The secondary arterial contains the collector and the minor arterial types in both network systems. The tertiary arterial included the country arterial in the Seattle road system and the *other* arterial in the Bellevue road system. Lastly, the unclassified category had the undesignated (Seattle) and the local (Bellevue). Other road types are out of interest in this report.

The road types of the two networks were rearranged on the basis of the corresponding road categories discussed above. The networks were then connected by merging their SR 520 and the I-90 road links. We named the combined network the "Greater Seattle road network." The Greater Seattle road network had 107,253 nodes and 223,800 one-way road edges, as shown in Figure 15. However, the network appeared too detailed for a macroscopic simulation, which might lead to excessive computation time for calibration and simulation. We therefore generated a new road network containing only the top-hierarchy roads, i.e., the motorways and the primary arterials. Roads that were not classified as the motorway or the primary arterial were mostly removed, but those connecting the top-hierarchy roads were retained. This was done with a two-step procedure. First, after retaining only the top-hierarchy roads, we located the broken roads by checking the network connectivity. The method we utilized for connectivity check was the Breath-First Search Traversal [9] (see Algorithm 1). Second, we manually connected the unreached top-hierarchy roads by putting back the lower-level road link. The procedure needed to be repeated multiple times until the overall network was connected. The Greater Seattle road network now have 34,631 road links, as shown in Figure 16.

(a) Seattle (b) Bellevue

Figure 14: Collected Network Datasets

Table 2: Road Types in the Combined Greater Seattle Network

	Seattle Network	Bellevue Network	Combined Network
Freeway	Interstate freeway	Highway	Motorway
	State Route freeway	Highway Ramp	
Arterial	Principle arterial	Major arterial	Primary arterial
	Collector arterial		Secondary arterial
	Minor arterial		
	County arterial	Other arterial	Tertiary arterial
Others	Not designated	Local	Unclassified
	-	Pedestrian corridor	-

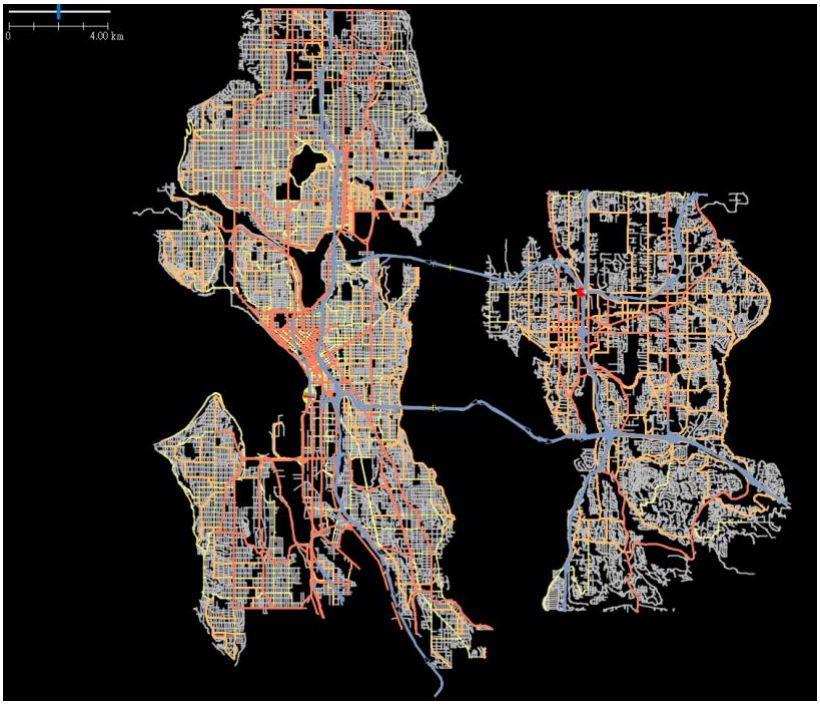


Figure 15: Initial Network for the Greater Seattle Area

Algorithm 1: Breath-First Search

```
Input G graph, s any node in G
function BFS(G, s):
      Initialize Q be a queue and mark s as visited
      Q.enqueue(s)
      while Q is not empty:
          v = Q.dequeue()
          for all edges w in G.adjacentEdges(v):
              if w is not marked as visited:
                  Q.enqueue(w)
                  mark w as visited
```

Next, we added bus routes to the Greater Seattle road network. The added bus routes were the 25 percent most productive routes evaluated by King County Metro (KCM) in 2014 [10]. The evaluation considered the measures of "rides per platform hour" and "passenger miles per platform mile in all time periods served," which are clearly described in the 2014 Service Guidelines Report [10]. The high-productivity routes in the Greater Seattle area include Routes 316, 41, 49, 71, 72, 73, 76, 77, B Line, D Line, E Line, 15EX, and 74EX. Extracting the route data from the General Transit Feed Specification

(GTFS) route dataset and the King County Metro (KCM) October schedule dataset of the same year, we considered just the former eleven routes because the last two routes were not included in the October schedule dataset, as displayed in Figure 17. More bus routes could be integrated into the MATSim model if needed, with proper calibrations.

Although GTFS route data and bus schedules provide information that included directions, service locations, service schedules, and corresponding route IDs, the route data could not be directly used by the MATSim model, as they were neither in the road map format nor in the format of the MATSim schedule. Each bus route with its schedules needed to be mapped onto the MATSim network and MATSim schedules. Therefore, we followed the approach proposed by Poletti [11] to map the chosen bus routes onto the Greater Seattle road network in MATSim.

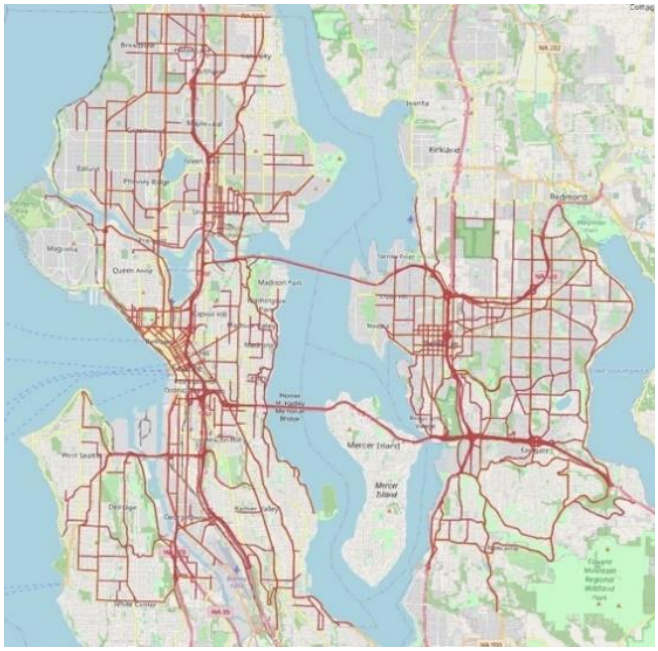


Figure 16: Major Roads in Greater Seattle

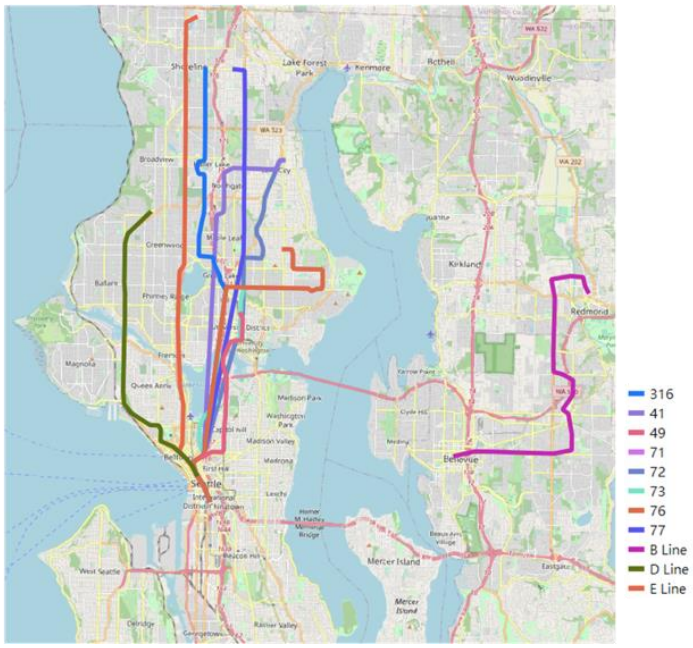
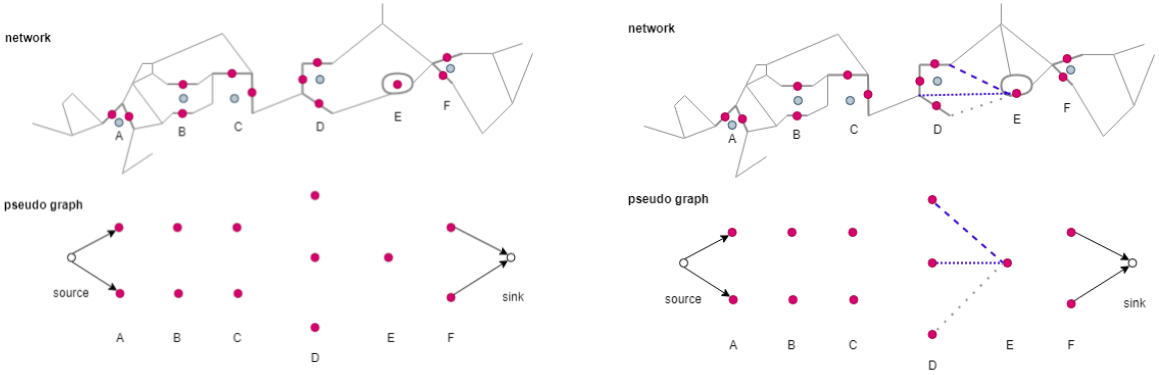


Figure 17: Picked Bus Routes

The mapping approach has five steps [11]: (i) sorting bus stops by route schedules, (ii) looking for link candidates that are closest to bus stops and recording the center of each link, (iii) generating a pseudo graph based on the recorded centers (see Figure 18 (a)), (iv) given travel distances, locating a pseudo path that has the least cost between each link pair (see Figure 18 (b)), and (v) creating a link sequence from the pseudo paths and adding an artificial link if any two consecutive bus stop pair had no link connection. After including the selected bus routes in the network, the final Greater Seattle network had 36,295 road edges, as shown in Figure 19. We next calibrated the network given travel plans (see Subsection 3.2: Network Calibration).



(a) Pseudo graph from the recorded centers

(b) Pseudo paths of each link pair

Figure 18: Approach of Mapping Bus Routes on a Network [11]

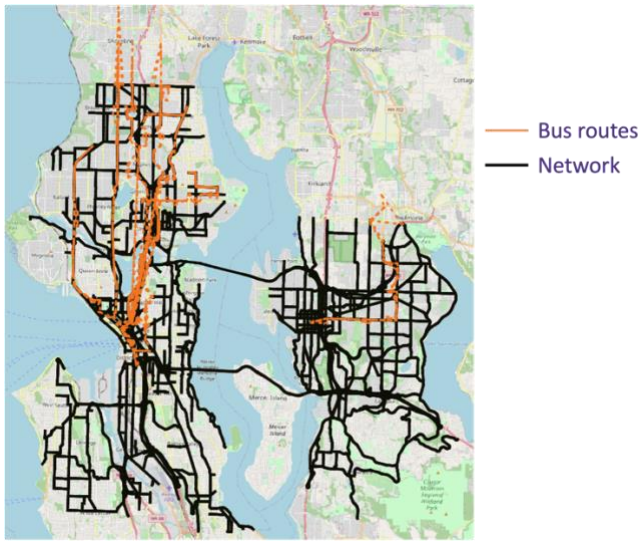


Figure 19: Finalized Network for the Greater Seattle Area

Subsection 3.2: Network Calibration

We collected the demand data from the 2018 travel data provided by SDOT, the full-day travel plan from the Puget Sound Regional Council's (PSRC) 2014 Activity-Based Travel Estimation, the October 2014 KCM schedule and 2014 GTFS data for bus routes, and the PSRC 2010 Traffic Analysis Zone (TAZ) data that contains the shapefiles describing the geometries of the TAZs. The MATSim calibration procedure is summarized in Figure 20, which is similar to the approach proposed by He et al. [12]. In this procedure, speeds and capacities were calibrated separately.

Subsection 3.2.1: Travel Plan Setup

We extracted home-based work trips that were self-driven or on buses for a typical day from the PSRC 2014 activity-based trip data. We further retrieved populations in the following cases: (i) both trip origin and destination were in our study area, (ii) only the trip origin was in our study area, and (iii) only the trip destination was in our study area. We treated trips in the former two cases as resident trips and trips in the latter case as non-resident trips. The number of driving resident trips was about ten times the number of driving non-resident trips; the number of bus-riding resident trips was about three times the number of bus-riding non-resident trips. In previous studies [12, 13, 14, 15, 16], a macroscopic simulation typically had used a scaled population sample instead of the total population because of the cost of computation. Table 3 lists their simulated populations and scale factors.

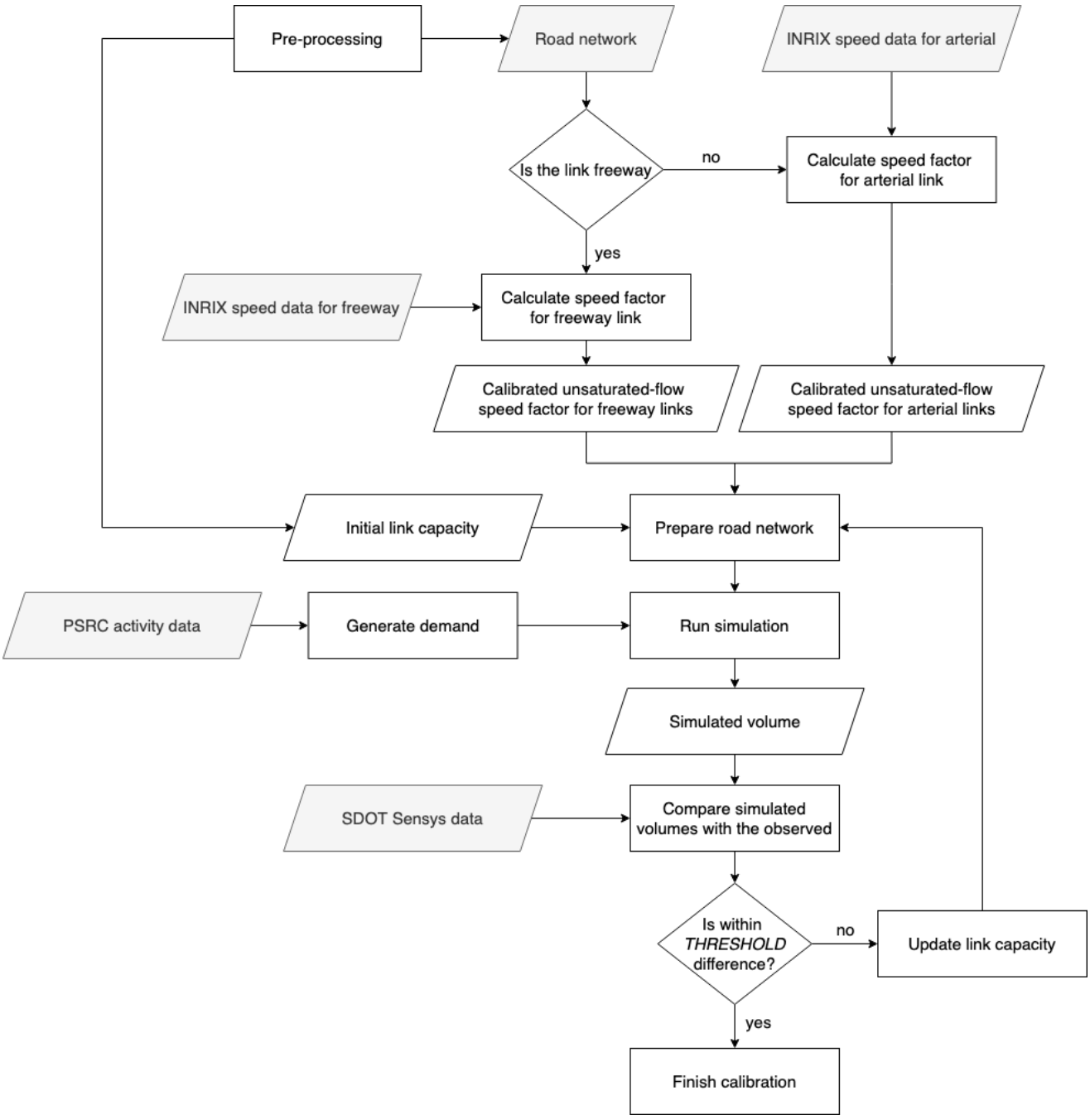


Figure 20: Calibration Process

Table 3: Scale Factors and Simulated Populations

City	Scale factor	Simulated population	Source	Year
Zurich	10%	181,693	[17]	2008
Zurich	10%	68,000	[19]	2011
Berlin	10%	-	[16]	2019
Paris	10%	-	[15]	2019
New York	4%	~350,000	[12]	2020

According to the 2020 Census Redistricting Data released by the U.S. Census Bureau on August 12, 2020, Seattle had a population of 761,100. On the basis of the latest investigation by the City of Bellevue, Bellevue had a population of 145,300 in 2019. In total, the population of the study area was 906,400. We then used 8 percent as the scaled population factor and sampled a 72,512 population for calibration. Note that in the MATSim simulation, we deployed several TAZ gates around the Greater Seattle area to simulate those agents whose origins or destinations were located outside. The TAZ gates wee deployed on the basis of the TAZ, as displayed in Figure 21. We selected 40 TAZs from the 2010 King County TAZ data around the Greater Seattle area as the gates, as listed in Table 4 and shown in Figure 21.

Table 4 TAZ Gates

#	TAZ ID	#	TAZ ID	#	TAZ ID	#	TAZ ID	#	TAZ ID
1	1	**9**	17	**17**	832	**25**	856	**33**	1640
2	2	**10**	19	**18**	837	**26**	857	**34**	1641
3	3	**11**	824	**19**	838	**27**	858	**35**	1642
4	6	**12**	825	**20**	839	**28**	861	**36**	1643
5	9	**13**	826	**21**	840	**29**	1408	**37**	1644
6	10	**14**	827	**22**	841	**30**	1450	**38**	1645
7	13	**15**	829	**23**	854	**31**	1638	**39**	1646
8	15	**16**	831	**24**	855	**32**	1639	**40**	1647

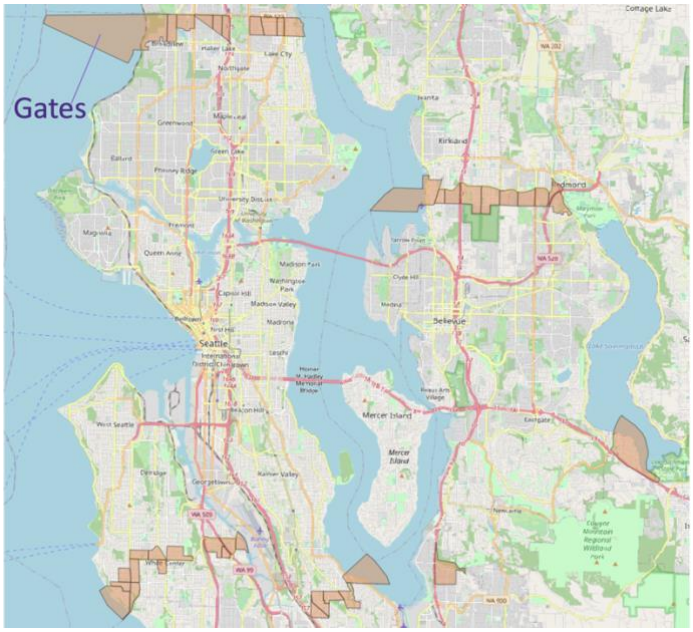


Figure 21: TAZ Gates around the Study Area

Subsection 3.2.2: Network Calibration

As discussed above, speed limits and capacities of network links were calibrated via the procedure shown in Figure 20. We split the INRIX data into speeds on arterials and speeds on highways. In the INRIX data, speeds from May 8th to 17th (weekdays) were used as the calibration references to compute speed factors for arterials and highways, respectively. The factors were then used for calibrating link capacities. In the capacity calibration, OD demands were prepared by considering only home-based work trips via the modes of driving or taking a bus. We extracted demands from the 2014 PSRC-generated activity data. The demands were input for simulation. Next, the link volume outcomes from the simulation were compared with the observed hourly weekday volumes in 2018 collected from Sensys. If the average volume difference was within a certain range, the calibration was done; otherwise, we used the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm [13, 14, 15] to update the link capacities. Note that for calibration, it is often preferable to collect observed data (volume, travel time, OD) for the same period of time (e.g., May 8 to 17, 2018). This was not possible for this large network because of limits in data availability. The major assumption here is that the OD demands in the area did not change much from 2014 to 2018.

The INRIX links are shown in Figure 22. We extracted typical weekday hourly speed data in 2018 from May 8th to May 17th and took the average to compute average hourly link speeds. The hourly speeds were split into six time periods: 6:00 to 9:00 am, 9:00 am to 12:00 pm, 12:00 to 3:00 pm, 3:00 to 6:00

pm, 6:00 to 9:00 pm, and 9:00 pm to 6:00 am. Then we computed the speed factors during each time period for arterial links and highway links, respectively. The observed speeds for the six time periods are listed in Table 5. The network was then calibrated on the basis of the link speeds. The speed factors were computed with equation (1), where $v^{\text{obs}}_{\text{type,time}}$ is the average observed speeds for a certain link type in a certain time period, $v^{\text{sim}}_{\text{type}}$ refers to the simulation link speed for a certain link type in a certain time period, and $f^{v}_{\text{type,time}}$ is the speed factor to calibrate the simulation speed to the observed speed.

$$v^{\text{obs}}_{\text{type,time}} = f^{v}_{\text{type,time}} v^{\text{sim}}_{\text{type}} \text{ , type} \in \{\text{arterial, highway}\} \text{ , time} \in \{1,2,3,4,5,6\} \quad (1)$$

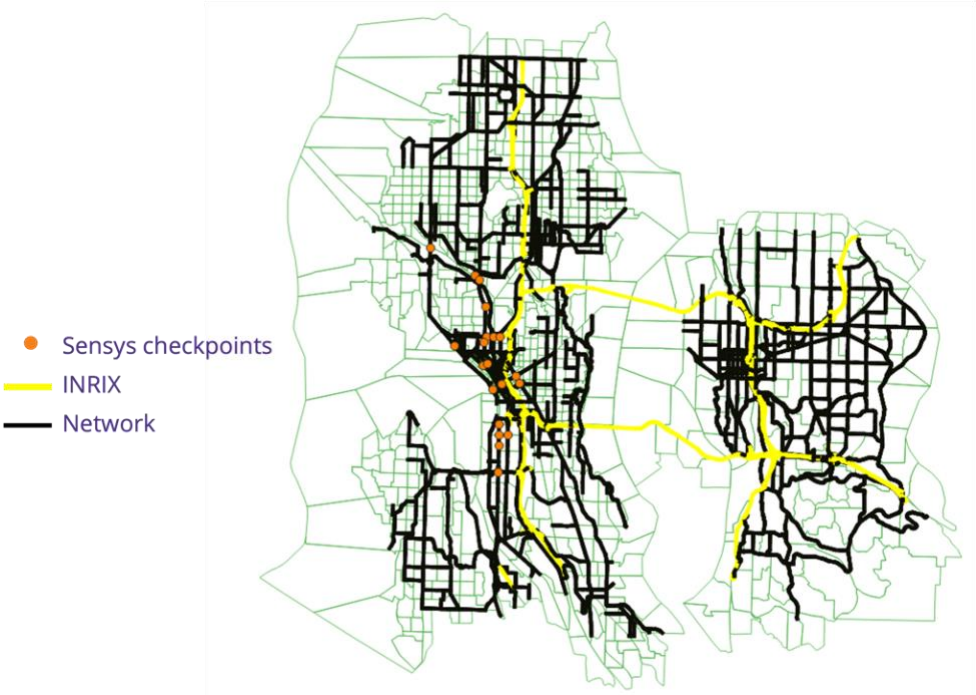


Figure 22: INRIX Links and Volume Checkpoints

Table 5: Observed Speed during Each Time Period and Link Types (mph)

	6 - 9 a.m.	9 a.m. - 12 p.m.	12 - 3 p.m.	3 - 6 p.m.	6 - 9 p.m.	9 p.m. - 6 a.m.
Highway	43.32	52.81	51.68	41.12	59.00	58.88

	6 - 9 a.m.	9 a.m. - 12 p.m.	12 - 3 p.m.	3 - 6 p.m.	6 - 9 p.m.	9 p.m. - 6 a.m.
Arterials	26.28	28.82	28.57	22.78	29.62	34.62

We used the 2018 Sensys traffic volume data to calibrate link capacities. To compare the observed traffic volumes with the simulated volumes, we first selected checkpoints that were intersected by major streets in Downtown Seattle, as shown in Figure 22 and listed in Table 6. Then the traffic volume data were processed into average observed hourly traffic volumes. These traffic volume data were classified into six groups by the six time periods discussed above. We used equation (2) to compute capacity factors. In this equation, $c^{\text{obs}}_{\text{type,time}}$ is the average observed volume for a certain link type in a certain time period, $c^{\text{sim}}_{\text{type}}$ is the simulation link volume for a certain link type in a certain time period, and $f^{c}_{\text{type,time}}$ is the capacity factor to calibrate the simulation speed to the observed speed.

$$c^{\text{obs}}_{\text{type,time}} = f^{c}_{\text{type,time}} c^{\text{sim}}_{\text{type}}, \text{type} \in \{\text{arterial}, \text{highway}\}, \text{time} \in \{1,2,3,4,5,6\} \quad (2)$$

Table 6: Traffic Volume Checkpoints

ID	Checkpoint	ID	Checkpoint	ID	Checkpoint	ID	Checkpoint
1	Ballard Bridge	**7**	1st & Edgar Martinez	**13**	Western Ave & Elliott Ave	**19**	Aurora & Harrison
2	Boren Ave & Madison St	**8**	4th & Madison	**14**	4th Ave S & Holgate St	**20**	1st Ave S & S Spokane St
3	Boren Ave & James St	**9**	Aurora Ave & Howe St	**15**	Westlake Ave N & Mercer St	**21**	Aurora Bridge
4	2nd Ave & Blanchard St	**10**	Dexter Ave N & Mercer St	**16**	Fairview Ave N & Mercer St		
5	4th Ave & Lenora St	**11**	1st Ave S & Holgate St	**17**	1st Ave S & S Stacy St		
6	Alaskan & Madison	**12**	Spokane St Viaduct	**18**	Fremont Bridge		

Then we adjusted the capacity factors several times until the average calibrated volume was close enough to the average observed traffic volume. The adjustment used the SPSA algorithm, as shown in Algorithm 2. In this algorithm, we defined capacity factors as *theta_initial*. Given the SPSA coefficients

beta = 0.602, *gamma* = 0.101, *A* = 3500, the gain sequence of step size a = 0.16, and the decreasing sequence of positive number c = 0.05, we did the following: (i) computed the Bernoulli variables *delta*, (ii) updated a and c, (iii) approximated the gradient at the current capacity factor, (iv) checked whether the capacity factor converged, and (v) quit if it converged; otherwise, went to step (i). We ran this algorithm multiple times until the difference between the average calibrated capacity and the average observed was below 12 percent (see Figure 23).

Algorithm 2: SPSA Algorithm [13, 14, 15]

```
Input initial link capacities as a list theta_initial
Input constant beta, gamma, a, A, c, THRESHOLD, MAXITER
Initialize iter = 0; diff = inf
Initialize theta(0) = theta_initial
Initialize delta = in p-dimension filled with zeros
function SPSA(beta, gamma, a, A, c, f_initial):
      while diff > THRESHOLD and iter <= MAXITER:
          delta(j):= generated by Monte Carlo,
                     each element is independently from Bernoulli distribution
                     with probability 0.5
          a(iter) := a / (iter + A)^beta
          c(iter) := c /(iter)^gamma
          g(iter) := gradient( y(theta(iter) + c(iter) * delta),
                               y(theta(iter) - c(iter) * delta), c(iter) )
          theta(iter + 1) = theta(iter) - a(iter) * g(iter)
          diff = abs( a(iter) * g(iter) )
          j += 1
```

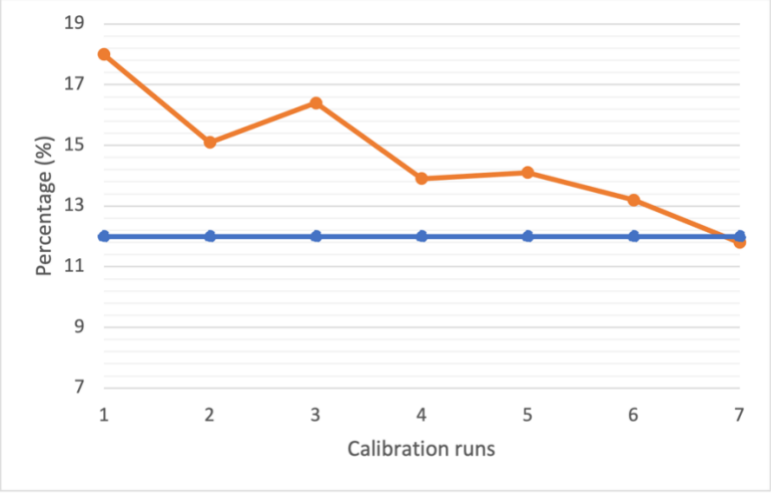


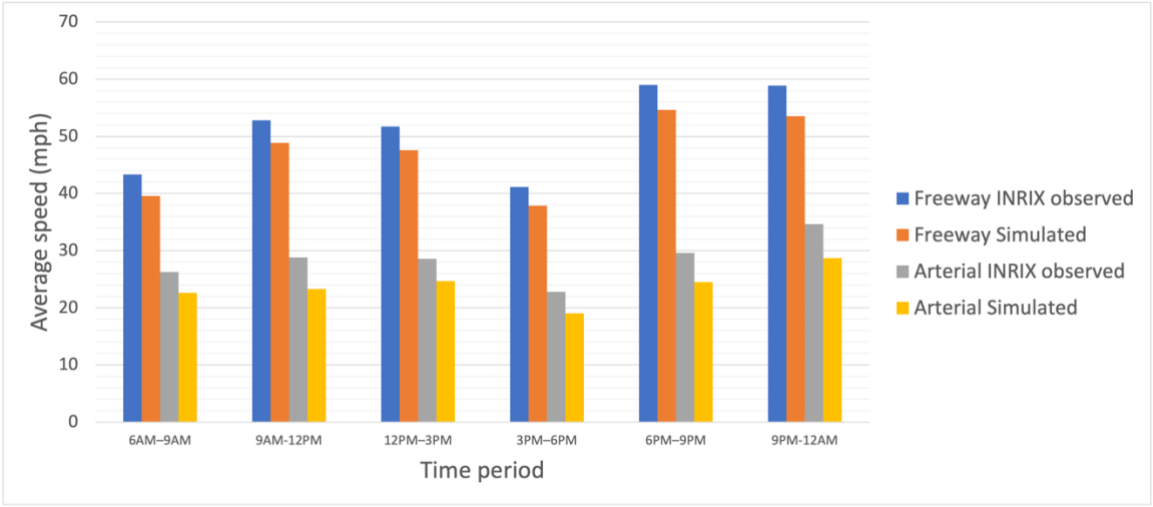
Figure 23: Link Capacity Errors with Calibration Runs

The computed speed factors and capacity factors are listed in Table 7 and Table 8, respectively. The results of calibration for link speed limits and capacities are shown in Figure 24 and Figure 25, respectively. The average difference between the calibrated highway speeds and the observed speeds

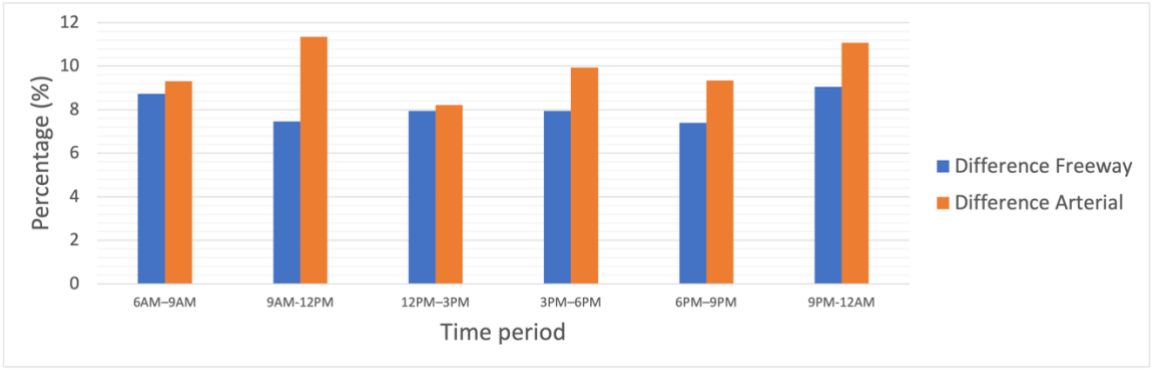
was 8.09 percent; the average difference between the calibrated arterial speeds and the observed arterial speeds was 9.87 percent; and the average difference between the calibrated capacities and the observed capacities was 11.55 percent.

Table 7: Speed Factors

	6 - 9 a.m.	9 a.m. - 12 p.m.	12 - 3 p.m.	3 - 6 p.m.	6 - 9 p.m.	9 p.m. - 6 a.m.
Highway	1.10	1.08	1.09	1.09	1.08	1.10
Arterials	1.16	1.24	1.16	1.20	1.21	1.21



(a) Speed Calibration

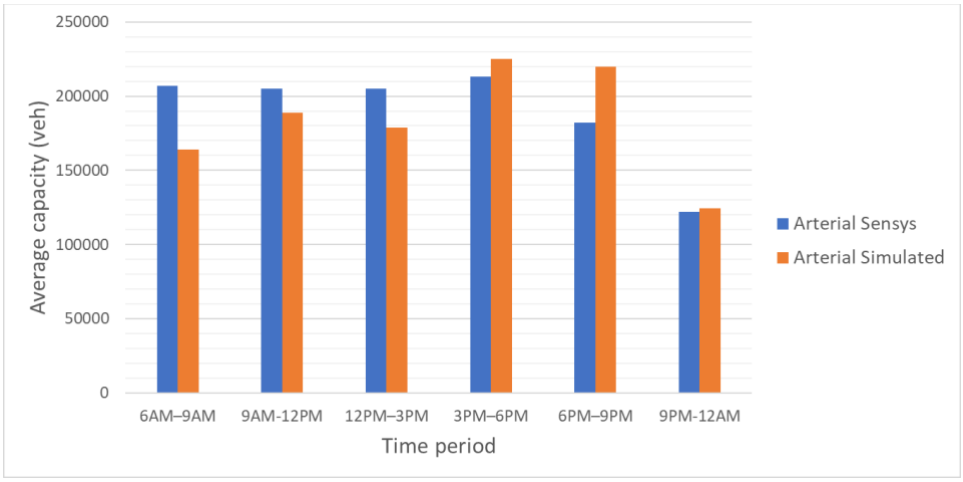


(b) Difference between the Simulated and the Calibrated

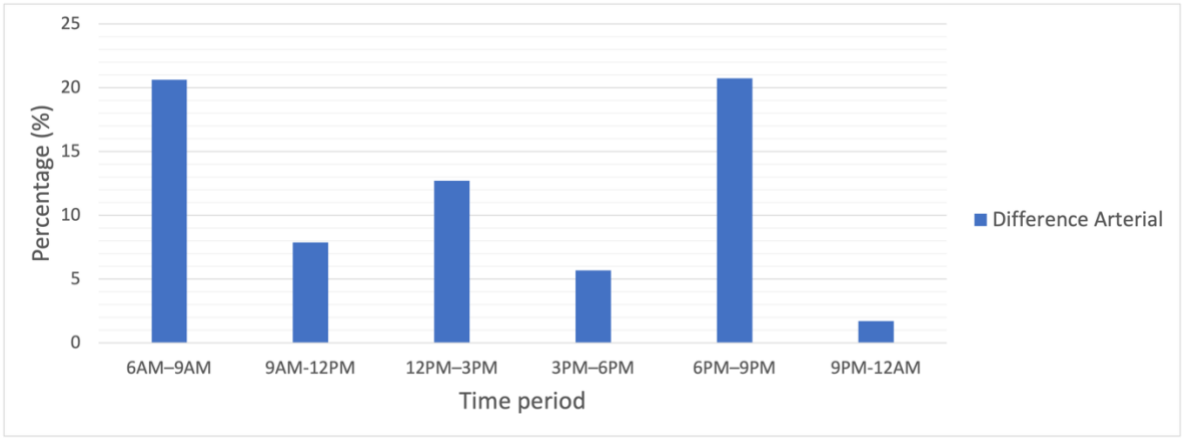
Figure 24: Speed Calibration Result

Table 8: Capacity Factors

6 - 9 a.m.	9 a.m. - 12 p.m.	12 - 3 p.m.	3 - 6 p.m.	6 - 9 p.m.	9 p.m. - 6 a.m.
0.96	1.43	0.73	0.69	1.06	0.51



(a) Capacity Calibration



(b) Difference between the Simulated and the Calibrated

Figure 25: Capacity Calibration Result

Section 4: SUMO Calibration

Subsection 4.1: Network Set-up

We constructed the SUMO network in the Downtown Seattle area on the basis of the OSM's digital road network data. The network data were then converted to SUMO by using the *netconvert* function, a command line application that imports digital road networks from various resources and generates road networks for SUMO [20]. The range map and the SUMO simulation network are shown in Figure 26 (north to Mercer Street, south to South Atlantic St/Edgar Martine Dt St, west to Alaskan Way and east to 12th Ave). The built-in SUMO network consisted of three major inputs: (i) the network file (.net.xml), the basic network file covering network information about edges, lanes, junctions and right-of-way (ROW), and connections; (ii) the route file (.rou.xml), a routes description file covering vehicle routes, pedestrian routes, and public transit routes (bus and link light rail); and (iii) the additional file (.add.xml), a further description file covering the TAZs, bus stops, and traffic signals. In addition, the traffic mode included in the SUMO simulation covered passenger vehicles, public buses, pedestrians, and Link light rail.

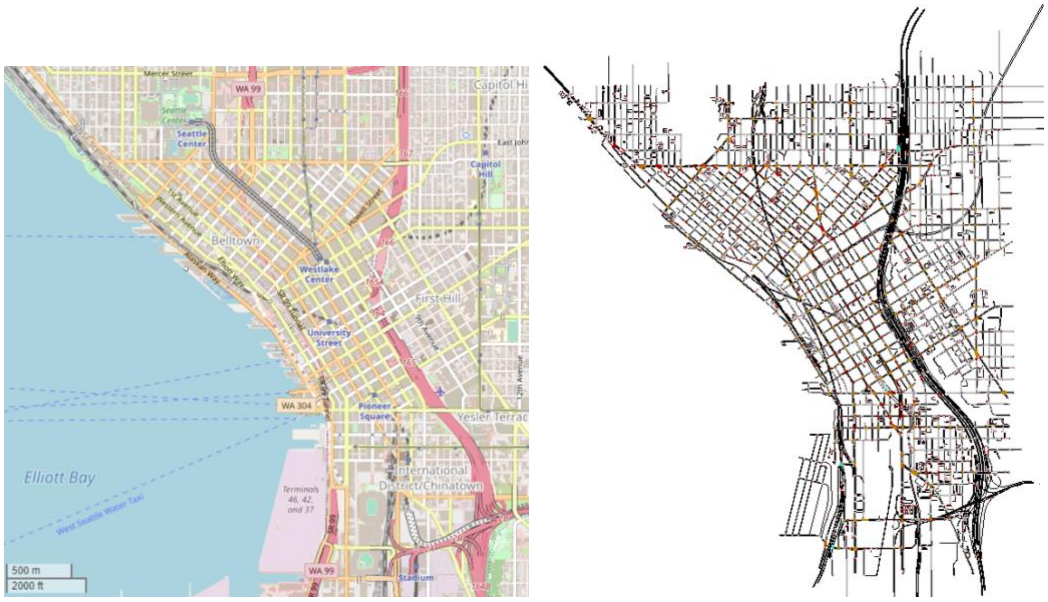


Figure 26: Range Map of SUMO Simulation OpenStreetMap Contributors (Left), SUMO Simulation (Right)

Subsection 4.1.1 Network File Set-up

The SUMO network file "describes the traffic-related part of a map, the roads and intersections the simulated vehicles and pedestrians run along or across" [21]. Network file cleaning and editing, given

the converted OSM network, were then conducted on the basis of specific physical characteristics, including number of lanes, intersection connections, road widths, speed limits, restricted lanes, crosswalks, and junction merges. Details of the features of the cleaned SUMO network are summarized in Table 9.

Table 9: Network Design in the SUMO Simulation

Main features	Detailed design	# of simulated elements in SUMO
Edges and Lanes	Edge is the connection between two nodes, each edge consists of a certain number of lanes. The edge and lane cover the road information includes: • Road length • Speed limit • Road priority	24259 edges for vehicles, (including bus) 5343 edges for pedestrians, 50 edges for link light rail
Junctions and ROW	Junction represents the area where different streams (edges) cross, covering the ROW rules of vehicles crossing the intersection.	7119 junctions
Connections	The connections describe the connection between each lane, i.e., which outgoing lanes can be reached from an incoming lane.	69283 connections

Subsection 4.1.2 Route File Set-up

The route file input in the SUMO simulation covered three types of routes: vehicle, pedestrian, and public transit (bus and Link light rail). Both vehicle and pedestrian routes were generated on the basis of the OD demand estimated by SoundCast, a travel demand model system built for the Puget Sound Region [22]. The starting location of a vehicle or pedestrian was generated randomly based on the SUMO default settings. To generate the route file, we used *OD2TRIPS*, a SUMO function that imports OD

matrices and converts them into single vehicle/pedestrian trips. The public transit routes were constructed from GTFS data [23]. As a result, 62 public transit routes (including one route for Link light rail) were included in the SUMO network. Details of the route design are listed in Table 10.

Table 10: Route Design in the SUMO Simulation

Type of routes	Detailed Design
Vehicle	Routes are defined on the basis of the features below. The passing routes given the O/D TAZ were generated automatically based on the shortest path. • fromTAZ (origin) • toTAZ (destination) • from (starting edge in the origin TAZ) • to (ending edge in the destination TAZ) • depart (Depart time) • departLane (the departure lane in the starting edge) • departSpeed (the departure speed)
Pedestrian	Routes are defined on the basis of the features below. The passing routes given the O/D TAZ were generated automatically based on the shortest path. • from (starting pedestrian lane in the origin TAZ) • to (ending pedestrian lane in the destination TAZ) • depart (Depart time) • arrivalPos (detailed position on the starting pedestrian lane) • departPos (Departure position on the ending pedestrian lane)
Public Transit	Routes are defined on the basis of the features below. • type (vehicle type) • from (starting edge of the bus route) • to (ending edge of the bus route) • depart (departure time at the first bus stop) • busStop (passed bus stops) • until (for each bus stop, the departure time at the current bus stop)

Subsection 4.1.3 Additional File Set-up

The additional file input into the SUMO simulation consisted of three components: TAZs, bus stops, and traffic signals. The TAZs and bus stops were defined in the same .add.xml file. A TAZ was geo-coded on the basis of the traffic analysis zone defined by the PSRC. Meanwhile, bus stops and Link light rail stations were geo-coded on the basis of GTFS data. Detailed settings for the TAZs and bus stops are summarized in Table 11. Traffic signal information was provided by SDOT, which uses Synchro as the design tool [24]. Therefore, we proposed a method to convert the traffic signal timing data from Synchro for use in SUMO. The remaining subsections discuss the traffic signal conversion in detail.

Table 11: TAZ and Bus Stop Design in SUMO Simulation

Type of additions	Detailed Design	# of simulated elements in SUMO
TAZ	TAZs in SUMO is described by the lists of source and destination edges, covering the edge and node information.	180 TAZs, including 10 pseudo TAZs for destination/origin outside the Simulation range.
Bus Stop	The bus stop is defined by the features below, • lane (the name of the lane the bus stop located at) • startPos (the begin position on the lane in meters) • endPos (the end position on the lane in meters) • friendlyPos (whether invalid stop position should be corrected automatically)	270 bus stops

For traffic signal set-ups, we converted the signal timing plans from Synchro for use in SUMO. Synchro is based on the Highway Capacity Manual's (HCM) 6th Edition for the design of signalized intersections, unsignalized intersections, and roundabouts, which provides users more options for signal timing designs [24]. On the other hand, SUMO is an open source, microscopic and continuous traffic simulation

package that was initially designed for large-scale networks. It allows modeling of intermodal traffic systems covering road vehicles, public transport, and pedestrians.

Because of their distinctive characteristics, Synchro and SUMO have been used to explore transportation problems from separate scales and perspectives. For instance, several studies [25, 26] have used Synchro to test various traffic signal timing algorithms and have used SUMO to test the corresponding impacts on vehicles. Most studies have focused on small areas, such as a single corridor with a limited number of intersections, which can easily be built manually. Few studies have used both Synchro and SUMO for a large road network. One of the leading reasons is that it can become a great challenge when the study area is expanded to a large range for various simulation platforms. A large-scale network simulation always brings more variation and complexity, given multiple types of signal timing plans, intersections, and transportation modes. Figure 27, as an example, displays diverse intersections in our SUMO simulation. Such variation increases the complexity of traffic signal conversion from Synchro to SUMO.

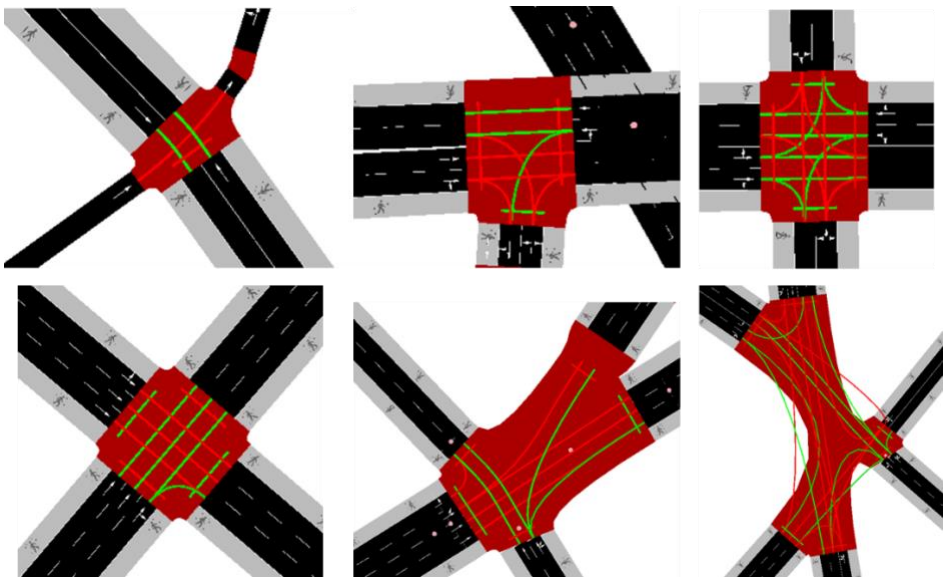


Figure 27: Diverse Intersections Display in SUMO Simulation

To convert traffic signals from Synchro to SUMO, we first compared the input network features to better understand the mechanisms behind the two simulation packages. The network data structure of Synchro can be extracted from a single CSV file. Specializing in traffic signal optimization at a macroscopic level, the network features of Synchro mainly focus on traffic signal settings. The network data of Synchro consist of timing/signing, phasing, lanes, volumes, and detectors. SUMO, on the other hand, is constructed on the basis of multiple xml files. Each file covers distinctive network features from Synchro, including road networks, vehicle routes, and additional features such as advanced signal timing

plans. By checking the main network features and corresponding sub-features, we identified the features that were transferrable between SUMO and Synchro, summarized in Table 12.

Table 12: Similar Feature Comparison between Synchro and SUMO

Settings	Synchro		SUMO	
	Features	**Details**	**Features**	**Details**
Road Network	Links	Road setting, including road name, direction (e.g., north bound), road distance, grade, number of lanes, etc.	Edge	From .net.xml file, including road type (allowed vehicle class), priority, number of lanes, etc.
	Lanes	Sub feature of link, a link with one direction can cover multiple lanes. Including speed limit, width, and storage etc.	Lane	From .net.xml file, sub-feature of edge, including road distance (length), coordinates and allowed vehicle classes, etc.
Intersection	Nodes	The position of each intersection, including X, Y, Z coordinates, intersection type, etc.	Junction	From .net.xml file, represents the area where various streams cross, covering the right-of-way rules, X, Y coordinates, and connected lanes, etc.
			connection	From .net.xml file, describes the connection between two lanes (from lanes and to lanes), including the direction of the connection (e.g., straight, left), the state

Settings	Synchro		SUMO	
	Features	**Details**	**Features**	**Details**
				(e.g., major link), and corresponding signals, etc.
Traffic signal	Timeplans	The signal timing plans, covering signal control types, cycle length, offset, etc.	tlLogic	From .net.xml/.add.xml file, defines the phases of traffic light, covering control types, offset, and phase index, etc.
	Phases	The phasing data specifically for ring barrier control, including BRP (barrier, ring and position), minimum and maximum green time, yellow time and red time, etc.		
Simulation	Network	Basic simulation settings, normally applied for the entire simulation network, including all red time, vehicle length, and scenario date and time, etc.	Configuration	Defines under .sumocfg file, including the basic config for sumo simulation, including the input and output .xml files, simulation time, and devices, etc.

On the basis of the compared features, we identified the feasibility and limitations of the simulation conversion. First, both road network and signal timing plans under the two simulations had similar features, while some of them were defined in separate ways. Nevertheless, a direct transfer based on feature mapping was not viable, as the features were not matched one-to-one. For instance, for traffic lights that control a single intersection, the traffic timing plans in Synchro use the road direction (e.g., northbound). However, SUMO [27] applies a clockwise pattern from 0 to 12 o'clock, with right turns ordered before straight movements and left turns. Second, simulation conversion from a macroscopic level to a microscopic level faces more challenges with missing features. As shown in Table 12, although network structures in SUMO and Synchro are similar (regarding lanes, links, and intersections), some

sub-features such as road type are missing in Synchro. Such conversion challenges between different simulation scales increase for a large traffic network. The larger the simulation area, the greater the unmatched error and noise, making network checking and cleaning time-consuming. Moreover, manually revising traffic signals between the two platforms for a large road network might even be harder, given the various signal control types and various ways of defining signal timing. Therefore, it was critical and necessary to explore a viable approach to efficiently converting traffic signal data between the two simulation packages.

We first learned the traffic signal settings of both Synchro and SUMO. The traffic signal design for a specific intersection in Synchro is displayed in Figure 28. In comparison to SUMO, traffic signal settings in Synchro are more intuitive. Synchro designs timing settings for each direction (e.g., NBL, northbound left), covering the minimum initial, minimum split, yellow, and all-red time. SUMO, on the other hand, defines traffic signals following a clockwise order rather than traffic direction. Each feature within a phase describes the state of one signal of the traffic light, and each link has the current state at each phase. As shown in Figure 29, the traffic signal in SUMO is defined on the basis of a clockwise circle starting in the northern direction. Each phase includes the traffic light states, minimum and maximum duration, etc.

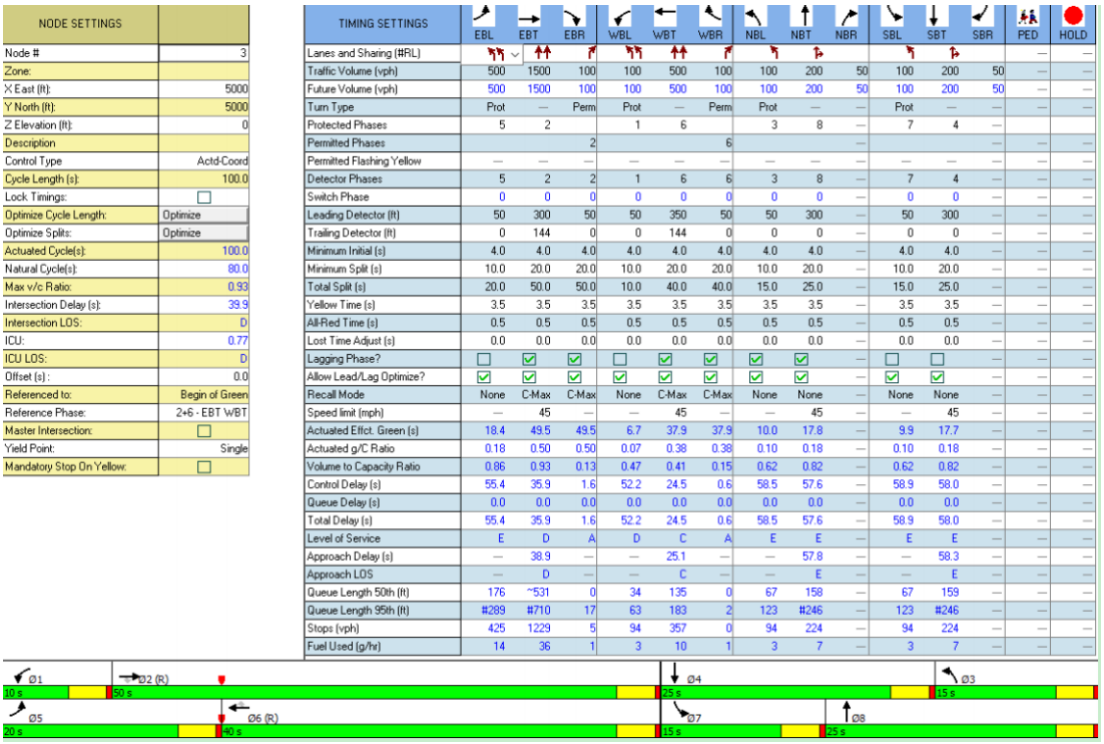


Figure 28 Signal Timing Settings in Synchro [28]

```
<tlLogic id="0" programID="my_program" offset="0" type="actuated">
  <param key="max-gap" value="3.0"/>
  <param key="detector-gap" value="2.0"/>
  <param key="passing-time" value="2.0"/>
  <param key="vTypes" value=""/>
  <param key="show-detectors" value="false"/>
  <param key="file" value="NULL"/>
  <param key="freq" value="300"/>

  <phase duration="31" minDur="5" maxDur="45" state="GrGr"/>
  ...
</tlLogic>
```

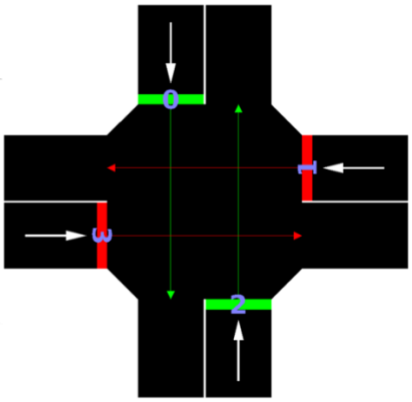


Figure 29: Traffic Signal Settings in SUMO [27]

A detailed traffic signal feature comparison between SUMO and Synchro is summarized in Table 13. It shows that most of the features between the two simulations could be well matched. However, variation occurred in some sub-features and default settings. For instance, Synchro applies ring barrier control for traffic signal design, whereas SUMO uses fixed traffic timing plans as the default. Therefore, traffic signal conversion required a bridge that would enables the feature information between the two platforms to be transferrable. We proposed a four-step approach that could automatically translate selected information between the two simulation platforms.

Table 13: Traffic Signal Feature Comparison

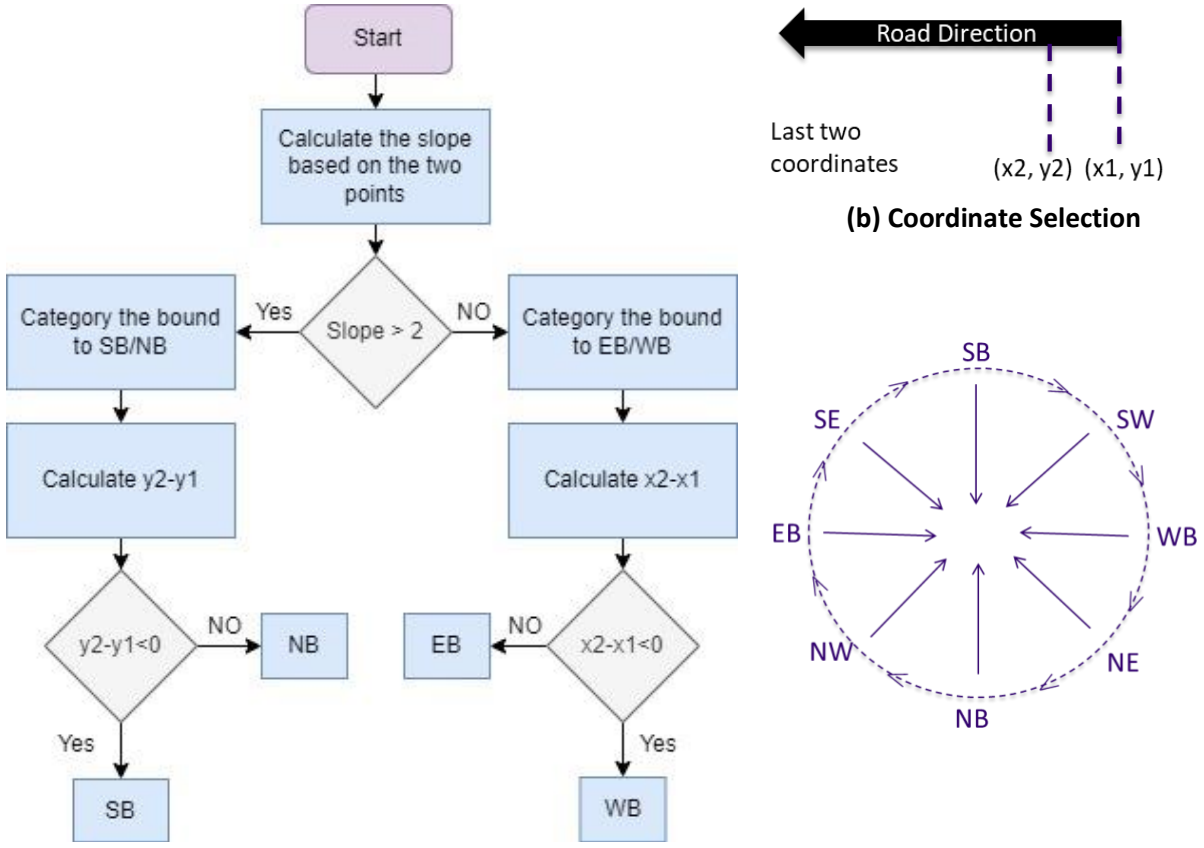
Item	Synchro	SUMO
Data source	• Timeplans • Phases • Lanes • Links	• tlLogic • Connection • Edge
Direction	Direction for each phase is specified as bound + direction (NBL, NBT, NBR), bound information varies, such as NW, NE, NB	No defined bound information, direction in SUMO means straight (s), left (l), right(r), etc.

Item	Synchro	SUMO
Phases	Designed based on the BRP (barrier, ring and position) with time on the horizontal axis and each row describes the states for one signal. Each phase includes transition (red/yellow) between green phases.	Each phase describes all signal states that last for a fixed duration with time on the vertical axis. Transition phase can be made up of multiple intermediate phases.
Detector	Detector is not specified	Detector information needs to be specified
Pedestrian button	Designed in Timeplans	Needs to be programmed use TraCI

The four-step approach includes the following: (i) intersection mapping, (ii) signal direction bound mapping, (iii) features extraction and mapping, and (iv) phase mapping. This method is under the prerequisite that the network has been successfully built with correct and clean road features within each simulation platform. Step (i) to step (iii) compose the intersection feature extraction and mapping, while step (iv) specifically focuses on the traffic signal phase and timing conversion. Step (i) involves mapping the intersection IDs between SUMO and Synchro. Each intersection from the two simulations generates a unique ID, and the intersection mapping prevents the traffic signal conversion conducted at the right (matched) intersections. Note that manual mapping is still needed when the intersection ID between the two simulation platforms is not defined in the same way.

Step (ii) involves mapping the signal direction. As summarized in Table 13, traffic signal information in SUMO does not cover the traffic direction; instead, signal status for each lane is defined in a clockwise order, with the pedestrian phase defined after the vehicular phase. The 0 o'clock (starting direction) is often located at due north (i.e., the southbound direction, SB, shown in Figure 30) under the condition that the intersection is built as a positive cross. Given that intersections are constructed in different shapes and with various angles in the actual road network, identifying the starting direction as well as the traffic direction in SUMO became essential for successful and smooth signal direction mapping. To achieve this, we used the shape information from the connected lanes of each intersection. More specifically, the last two coordinates were selected from each incoming edge of the intersection to

calculate the traffic direction (Figure 30 (b)). As shown Figure 30 (a), after determining the traffic signal timing order, we estimated vehicular direction. First, we categorized the direction into four types (eastbound as EB, southbound as SB, westbound as WB, and northbound as NB). Then we identified the most possible direction based on the slope and coordinate difference of the chosen coordinates. Given that direction in Synchro is defined in multiple ways (e.g., not only NB, but also NE and NW), the final direction was estimated on the basis of the order of Figure 30 (c)). With the traffic direction and direction order, the direction was assigned on the basis of both the lane direction in SUMO (under the Connection feature) and the traffic direction in Synchro, following the direction order from right turn (R), through (T), and left turn (L). Having estimated the direction of vehicles, we then estimated pedestrian direction. Unlike vehicular direction, pedestrians have no unique direction, as they are able to cross an intersection from EB to WB and vice versa. On the other hand, the order of the pedestrian direction begins with the crossing direction in SUMO of the first vehicular direction. Thus, the corresponding crossing vehicular edges were identified to estimate the pedestrian direction.



(a) Flow Chart of Direction Identification

(b) Coordinate Selection

(c) Direction Order

Figure 30: Direction Identification

Step (iii) involves transfering and inspecting all the other features/sub-features from the two simulation platforms for further conversion. Most of the features were selected under the phase feature of Synchro

and the tlLogic feature of SUMO, covering traffic signal control type (e.g., fixed timing, actuated timing), offset, the appropriated BRP, minimum and maximum green time for each phase, yellow and all-red times, and time settings for the pedestrian interval. Such feature information was then transferred to the corresponding intersections. Nevertheless, not every intersection allowes a perfect mapping (with all feature information well mapped from Synchro to SUMO), especially the directions, for instance, missing the right/left turn at a specific direction. These variances need to be inspected and corrected to guarantee a successful traffic signal simulation conversion.

With traffic signal feature information mapped, step (iv) involves assigning the signal phase order and timing for each intersection. The format of signal phase settings varies in SUMO and Synchro. Synchro has a built-in Ring-and-Barrier Designer to simulate the signal ring-barrier controller. Once the user assigns the phase number to the BRP field, signal phase transition in Synchro is conducted automatically. On the other hand, SUMO uses National Electrical Manufacturers Association (NEMA) type logic to manage ring-barrier control signals [29]. The NEMA phase setting example is displayed in Figure 31. It requires features including the phase number of each ring, barrier phase numbers, recall time for the signals, and phase definitions. For phase definitions, it requires the minimum and maximum duration for each green phase, timing for yellow and red phases, and vehicle extension timing in seconds.

```
<add>
    <tlLogic id="2881" offset="10" programID="NEMA" type="NEMA">
        <param key="detector-length" value="20"/>
        <param key="detector-length-leftTurnLane" value="10"/>
        <param key="total-cycle-length" value="120"/>
        <param key="ring1" value="1,2,0,4"/>
        <param key="ring2" value="0,6,0,4"/>
        <param key="barrierPhases" value="2,6"/>
        <param key="coordinate-mode" value="true"/>
        <param key="barrier2Phases" value="4,4"/>
        <param key="minRecall" value="2,6"/>
        <param key="maxRecall" value=""/>
        <param key="whetherOutputState" value="true"/>
        <param key="fixForceOff" value="false"/>

        <phase duration="99" minDur="6"  maxDur="16" vehext="2" yellow="4" red="1" name="1"
state="srrrrrrGGrrr"/>
        <phase duration="99" minDur="10" maxDur="67" vehext="2" yellow="4" red="1" name="2"
state="srrrrrrrrGGG"/>
        <phase duration="99" minDur="10" maxDur="22" vehext="2" yellow="3.5" red="1.5"
name="4" state="GGGGGrrrrrrr"/>
        <phase duration="99" minDur="10" maxDur="88" vehext="2" yellow="4" red="1" name="6"
state="srrrrGGrrrrr"/>

    </tlLogic>
```

Figure 31: Ring Barrier Control Example in SUMO [29]

Subsection 4.2: Network Calibration

We calibrated the SUMO network to the observed traffic volumes and travel times.Three data resources were applied for calibration, including the OD demands estimated by SoundCast as simulation input, TRACFLOW data from WSDOT (highway traffic volumes), and NPMRDS (National Performance Management Research Data Set) data (highway and local street travel time) as the field measurements. The objective of the network calibration was to acquire the best match between model performance estimates and field measurements. Parallel to this, the Wisconsin DOT freeway model calibration criteria were applied as our calibration targets [30]. The calibration criteria are shown in Table 14.

Table 14: Wisconsin DOT Freeway Model Calibration Criteria [30]

Criteria and Measures	Calibration Acceptance Targets
GEH Statistic < 5 for Individual Link Flows	> 85% of cases
Travel Times, Model Versus Observed	
Journey Times, Network: Within 15% (or 1 min, if higher)	> 85% of cases

The GEH statistic was computed as follows:

$$GEH = \sqrt{\frac{(E-V)^2}{(E+V)^2}}$$

where E denotes the model estimated volume while V denotes the field count.

We then conducted the SUMO network calibration through various perspectives, including network features, car-following model parameters, and OD demand, as summarized in Table 15. The estimated data from the SUMO simulation were collected by using the edge/lane-based traffic measures [31]. The data collection covered the traffic volume and travel time information for each edge, including the number of vehicles emitted onto the edge/lane within the settled time interval (departed), the number of vehicles that finished their routes on the edge/lane (arrived), the number of vehicles that entered the edge/lane from upstream (entered), the number of vehicles that left the edge/lane from downstream (left), and times needed to pass the edge/lane (traveltime).

Table 15: Tune Features for SUMO Calibration

Category	Focused features	Description
Network structure	Speed validation	Speed limit check along each edge, especially the local street for better calibration
	Lane configuration	Specifically for the lanes that encounter unusual congestion (compared with actual traffic), including lane priority, edge direction, route connectivity.
	Signal & Stop sign configuration	Focused on the lane priority (yield or zipper) of the merge intersection/merge ramp.
Vehicles	Car following model parameter	The car-following model related parameters for calibration mainly include tau (the driver's desired time head way) and sigma (the drive imperfection).
	Basic attributes	Including acceleration and deceleration ability check of vehicles, and the minimum gap when standing.
OD demand	Trip arrival time	For each given OD pair, check the estimated trip arrival time based on the SoundCast demand and revise the routes for better calibration.

Category	Focused features	Description
	OD input	Specifically for the origin/destination located on the rim of the simulation network since it connects with the inbound/outbound demand.

The calibration results are shown in Table 16. The SUMO calibration was conducted with 15 road segments selected for the highway traffic volume calibration and 25 road segments chosen for the local street travel time calibration. As Table 16 shows, 80 percent of the traffic volumes and 50 percent of the travel times were well calibrated based on the calibration criteria in Table 14. The calibration performance could be further improved, which however requires more time and computational resources.

Table 16: SUMO Calibration Results

Index	Test Road Segment ID*	Road name	Peak hour expected	Peak hour simulated	GEH	Calibrated	Test Features
1	4722443	I-5 SB Exit Stewart	770	698	7.06267	TRUE	Volume
2	4755219#0	I-5 SB Enter Yale Ave	1110	1020	7.605634	TRUE	Volume
3	96260970	I-5 SB Exit Union St	1020	1413	126.961776	FALSE	Volume
4	96260967	I-5 SB Exit 6th Ave	1470	1686	29.56654	TRUE	Volume
8	35824613	I-5 SB CD Enter Spring	1070	1526	160.197227	FALSE	Volume
6	4748988	I-5 SB CD Exit Dearborn	200	288	31.737705	TRUE	Volume
7	4748998	I-5 SB CD Exit 4th Ave S	420	541	30.470343	FALSE	Volume
8	4712866	I-5 NB CD Enter Spokane	410	381	2.126422	TRUE	Volume

Index	Test Road Segment ID*	Road name	Peak hour expected	Peak hour simulated	GEH	Calibrated	Test Features
9	402084478	I-5 NB CD After Spokane	1070	1071	0.000934	TRUE	Volume
10	4848517	I-5 NB CD Enter Dearborn	370	410	4.102564	TRUE	Volume
11	105899959	I-5 NB CD Exit James	1170	1012	22.88176	TRUE	Volume
12	56178982	I-5 NB Exit Seneca	960	1021	3.756689	TRUE	Volume
13	171121268	I-5 NB Enter Univ St	530	516	0.374761	TRUE	Volume
14	436165683#0	I-5 NB Exit Olive Way	700	684	0.369942	TRUE	Volume
15	621342731	I-5 NB Enter Olive Way	1150	1184	0.990574	TRUE	Volume
16	114+08028	Madison St	16.12	15.427	0.030447	TRUE	Travel time
17	114+08088	Battery St	32.75	31.618889	0.039753	TRUE	Travel time
18	114+08224	Marion St	38.21	60.567778	10.121107	FALSE	Travel time
19	114+08225	Marion St	34.3	59.222222	13.282772	FALSE	Travel time
20	114+08542	5th Ave	211.18	133.438	35.075467	FALSE	Travel time
21	114+08571	6th Ave	35.45	34.57	0.022119	TRUE	Travel time
22	114+10562	James St	39.61	33.262	1.105969	TRUE	Travel time
23	114+10563	James St	125.31	197.37	32.184478	FALSE	Travel time
24	114+11093	Pike St	83.4	63.135	5.60508	FALSE	Travel time
25	114+11102	Pike St	90.08	26.535	69.252961	FALSE	Travel time
26	114-08015	2nd Ave Ext S	227.3	136.936	44.837152	FALSE	Travel time
27	114-08018	2nd Ave	13.43	12.898	0.0215	TRUE	Travel time
28	114-08093	Wall St	31.24	51.361667	9.803228	FALSE	Travel time

Index	Test Road Segment ID*	Road name	Peak hour expected	Peak hour simulated	GEH	Calibrated	Test Features
29	114-08208	Madison St	45.03	55.7425	2.277559	TRUE	Travel time
30	114-08209	Madison St	26.78	44.023	8.398544	FALSE	Travel time
31	114-10098	Columbia St	6.43	9.865	1.448202	TRUE	Travel time
32	114-10501	S Jackson St	69.89	51.298889	5.703979	FALSE	Travel time
33	114-10502	S Jackson St	27.26	5.653333	28.368324	FALSE	Travel time
34	114-10560	James St	39.09	59.9275	8.770195	FALSE	Travel time
35	114-10561	James St	51.21	45.498889	0.674536	TRUE	Travel time
36	114-10562	James St	118.61	173.628	20.715857	FALSE	Travel time
37	114-11101	Pine St	87.85	95.1875	0.588283	TRUE	Travel time
38	114-11376	Stewart St	40.46	81.0775	27.148515	FALSE	Travel time
39	114-11557	Pine St	73	63.106	1.438456	TRUE	Travel time
40	114-15723	Pine St	62.76	63.64375	0.012357	TRUE	Travel time
41	114N10562	James St	13.6	22.385	4.289355	TRUE	Travel time

* for traffic volume, the road segment ID is the sumo edge id, for travel time is the NPMRDS road segment ID

Section 5: Testing Results

We show in this section some testing results using the developed multi-scale simulation platform, including multiscale traffic-vehicle control algorithms and learning traffic dynamics.

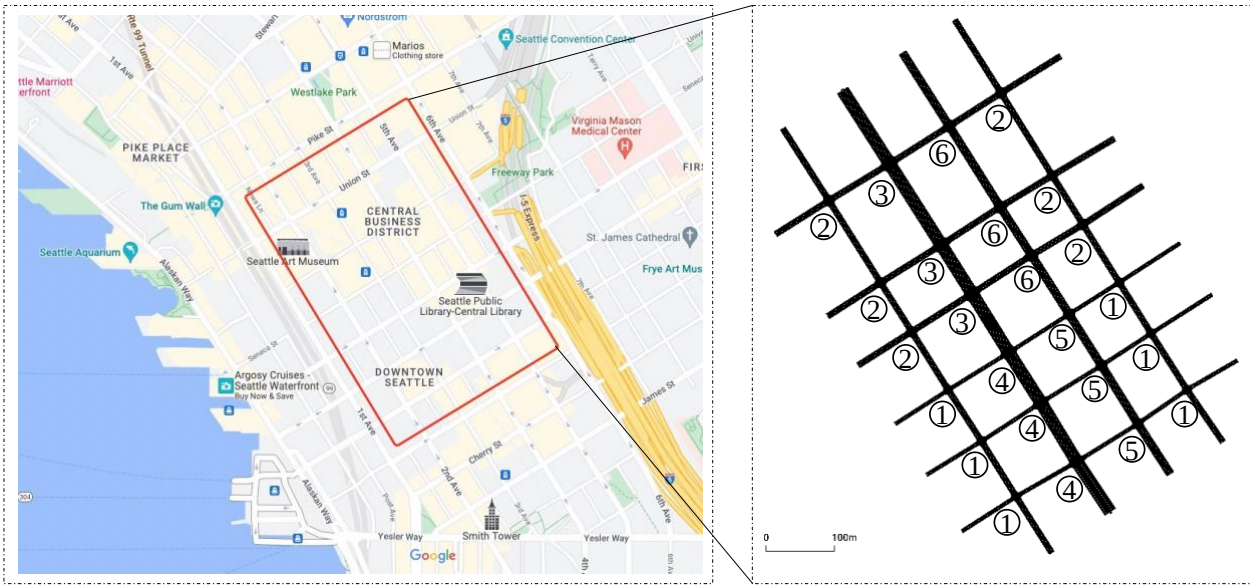
Subsection 5.1: Testing of a Multiscale Signal-Vehicle Coupled Control Algorithm

We built a multi-scale, signal-vehicle coupled control framework to improve network-wide urban traffic performance and used the VTD platform to test the proposed algorithm. The urban transportation system is multi-modal and multi-scale both spatially and temporally. For example, when signal control is at a macroscopic spatial scale (i.e., intersection scale) and a slower temporal scale (i.e., signals should be adjusted at least in seconds to avoid confusion), vehicle control is at a microscopic spatial scale and a faster temporal scale (i.e., vehicles should be controlled with a high frequency to guarantee safety). In this test case, we built a slower-scale signal control model to generate optimal signal phases based on a mixed integer, nonlinear programming formulation, and a faster-scale vehicle control model to generate longitudinal vehicle accelerations based on a nonlinear programming formulation [32]. The goal of signal control is to maximize total vehicle throughput, and the goal of vehicle control is to minimize fuel consumption while maintaining acceptable travel time. In addition, the slower-scale signal control algorithm will generate reference trajectories for all surrounding vehicles at slower-scale time points (i.e., waypoints with larger time steps), and the faster-scale vehicle control algorithm will allow vehicles to achieve the same trajectories but with a much smaller faster-scale time step.

The multi-scale signal-vehicle coupled control algorithm was first developed for a single intersection and fully connected and autonomous vehicle (CAV) penetration (so that we could control all the vehicles). Then, we extended it to multiple intersections and mixed traffic flow of both CAVs and human-driven vehicles (HDVs, which we cannot control). For the multiple intersections scenario, we used the information sharing technique to enable communications between neighboring intersections and then developed a revised, distributed multi-scale, signal-vehicle coupled control algorithm. Specifically, for a specific intersection, we collected the predicted vehicle trajectories generated by neighboring intersections and calculated the arrival times of those vehicles. Then we integrated those upcoming vehicles with the vehicles that were currently and physically on the surrounding roads to formulate a new, slower-scale problem. In this way, the current intersection algorithm could generate optimal signal phases by knowing the incoming vehicles. For the mixed traffic flow scenario, we designed a linear interpolation method to estimate the HDVs' states, based on which we formulated the slower-scale problem in the same way as above. Then we developed a safety check mechanism to adjust the commands generated by the faster-scale problem. Finally, we combined the two techniques, i.e., information sharing and the safety check, into an integrated, multi-scale, signal-vehicle coupled control framework that can be used for real-world traffic networks. We omit the technical details here for the

sake of brevity (also because this was not the focus of this project). Below wee illustrate the testing of the proposed framework using the VTD platform developed in this study.

As shown in Figure 32, we used a four-by-six-block downtown network in Seattle to test the multi-scale signal-vehicle coupled control algorithm. There are different road types (e.g., one-way and two-way roads) in this network, leading to six distinct types of intersection geometries. In Figure 32, the numbers in the circles in the SUMO networks represent the types of intersections. We extracted the OD volume data from the bigger Seattle-wide SUMO simulation network (as discussed in the previous chapter). The number of OD pairs is large, and we omit the detailed OD volumes here for the sake of brevity. To have an intuitive understanding, the maximum volume is 332 veh/hour from the upper right, north incoming road to the bottom right, south outgoing road.



Google map SUMO

Figure 32: The Tested Downtown Seattle Area

We compared the proposed multi-scale signal-vehicle coupled control algorithm (denoted as Multiscale) with the actuated signal control (denoted as Actuated). Table 16 shows the performance of these two control methods. Note that for the Actuated method, the vehicles were controlled by SUMO's default car-following models. We used the average waiting time, time loss, queue length, and fuel consumption as the evaluation indexes. The waiting time (in seconds) was defined as the number of seconds a vehicle had a speed of less than 0.1 m/s. Time loss was defined as the time lost due to traveling at speed below the maximum speed. The queue length was calculated by using the end of the last standing vehicle. There were two values for each performance index for the Multiscale method, in which the first one was

the value under 0 percent CAV penetration and the second one was the value under 100 percent CAV penetration.

Table 17: Performance of the Multi-Scale Signal-Vehicle Coupled Control

Method		Performance indexes			
		avg. waiting time (s)	avg. time loss (s)	avg. queue length (m)	avg. fuel (mg/s)
Actuated		19.06	44.78	9.04	0.144
Multiscale	value	17.79 - 10.51	40.12 - 33.23	8.80 - 7.00	0.140 - 0.110
	improvement	6.66% - 46.75%	10.41% - 25.79%	2.65% - 22.57%	2.78% - 23.61%

The table shows that the Multiscale method outperforms the Actuated signal control on all evaluation indexes. The performance of the Multiscale method increases as the CAV penetration rate increases because higher CAV penetration rate provides more accurate traffic state information and more controllability. The lowest performance gain is 2.65 percent (for the average queue length under 0 CAV penetration), and the highest gain is 46.75 percent (for the average waiting time under full CAV penetration).

In summary, the VTD platform provided a test platform for the signal control and vehicle control algorithms. Various indexes such as waiting time, time loss, queue length, and fuel consumption were generated and collected to help better evaluate the performance of the algorithm. Note that for this particular case study, we used the SUMO simulation only; other scales and layers (i.e., MATSIM and Unity) could bring more evaluation tools, control flexibilities, and design possibilities.

Subsection 5.2: Investigation of Traffic Dynamics

In this case study, we built a neural network to learn the car-following model from the simulated data. Data were collected from the SUMO simulation results. In our SUMO simulation, we used the Krauss Car-following model [33], as written in equation (3). In the equation, v_f and v_l are the velocity of the following vehicle and the leading vehicle, respectively; $g(t)$ is the gap between the two consecutive vehicles at time t; a_{max}, b, and $v_{\max}$ are the maximum acceleration, the deceleration, and the maximum velocity of the vehicle, respectively; v_s and v_d are the safe velocity and the desired velocity, respectively; ϵ is the random value for simulating human driving, it is in the uniform distribution; T refers to time interval. The following test focused on passenger vehicles, so a_{max}, b, and $v_{\max}$ are fixed:

$a_{max} = 2.6\ m/s$, $b = 4.5\ m/s^2$, and $v_{\max} = 55.55\ m/s$. We used 1 second as the time interval: $T = 1$. Plugging in these parameters, equation (4) shows the final learning equation.

$$
\begin{aligned}
v_f(t+T) &= \max(0,\ v_d - \epsilon a_{max}) \\
v_d &= \min\left(v_f(t) + a_{\max}T,\ v_s,\ v_{\max}\right) \\
v_s &= v_l + \frac{g(t) - v_l(t)T}{\frac{v_f(t) + v_l(t)}{2b} + T}
\end{aligned}
\tag{3}
$$

$$
\begin{aligned}
v_f(t+T) &= \max(0,\ v_d - 2.6\epsilon) \\
v_d &= \min\left(v_f(t) + 2.6,\ v_s, 55.55\right) \\
v_s &= v_l(t) + \frac{9.0[g(t) - v_l(t)]}{v_f(t) + v_l(t) + 9.0}
\end{aligned}
\tag{4}
$$

We then collected data by vehicle pair. Vehicle pairs were collected if vehicles were consecutive in the same lane of the same road section. In the example shown in Figure 33, we highlight the vehicle in red as the follower, whereas green is the leader. Vehicles were considered to be a pair if the vehicle pair was driving as the case shown in Figure 33 (a): vehicles were driving consecutively in the same road section and the same lane so that none of the vehicles in black was the leader or follower. Vehicle pairs in black and grey in Figure 33 (b) were not considered a pair because they did not drive in the same road section, although they were consecutive vehicles. After collecting data, we removed the vehicle pairs with either the follower or the leader idle because zero speeds would not provide any information to the candidate function; instead, it would confuse the learning model. The processed vehicle pairs were input to the learning model, each pair a data point with information about the velocity of the leading vehicle, the velocity of the following vehicle, the gap between them, and the timestamp. The velocities between the leading and following vehicles (see Figure 34) were greater than zero, 0.0001 at minimum, and the maximum leading velocity was slightly greater than the following velocity.

We attempted to build a neural network that would be able to learn meaningful features of car-following dynamics. The neural network structure was designed to be compared to equation (4). Equation (4) has three steps to calculate the predicted velocity: (i) safe velocity calculation, (ii) desired velocity calculation, and (iii) predicted velocity calculation. The first step uses $v_l(t),\ g(t), v_f(t), b = 4.5$ to calculate v_s, the second step uses v_s and $a_{\max} = 2.6$ to calculate the desired velocity, and then the

final step utilizes v_d and $a_{\max} = 2.6$ to predict the velocity at the next time frame. On the basis of these three steps, the neural network was designed to have an input layer, an output layer, and three hidden layers in between, as shown in Figure 35. The input layer received the six variables: $v_l(t)$, $g(t)$, $v_f(t)$, $b = 4.5$ and $a_{\max} = 2.6$; the output layer was the predicted velocity of the follower; in between, the first hidden layer was designed to initially process the input information while the other two layers were designed for calculating the safe velocity and desired velocity, respectively. Each hidden layer used the sigmoid activation function.

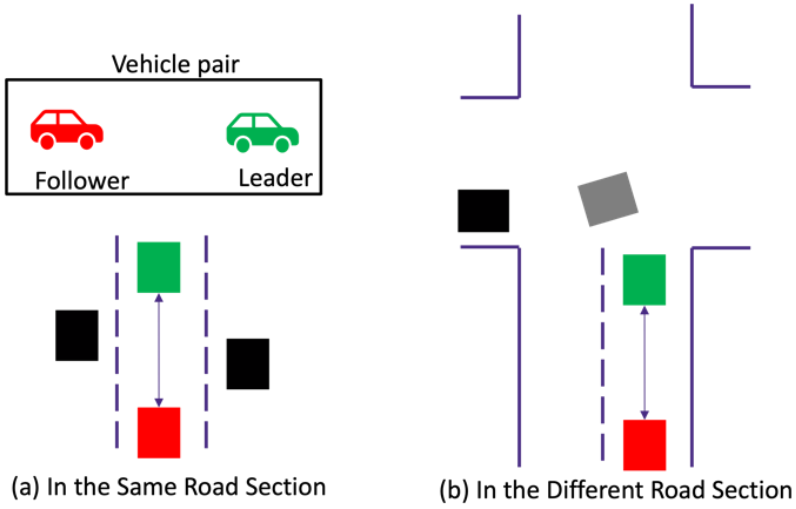


Figure 33: Data Collection by Vehicle Pairs

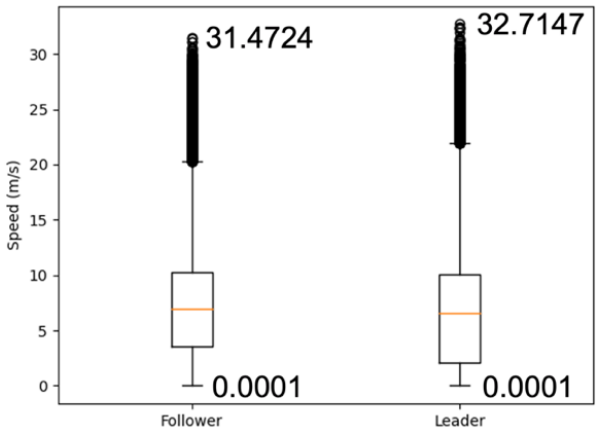


Figure 34: Velocities of Leaders and Followers

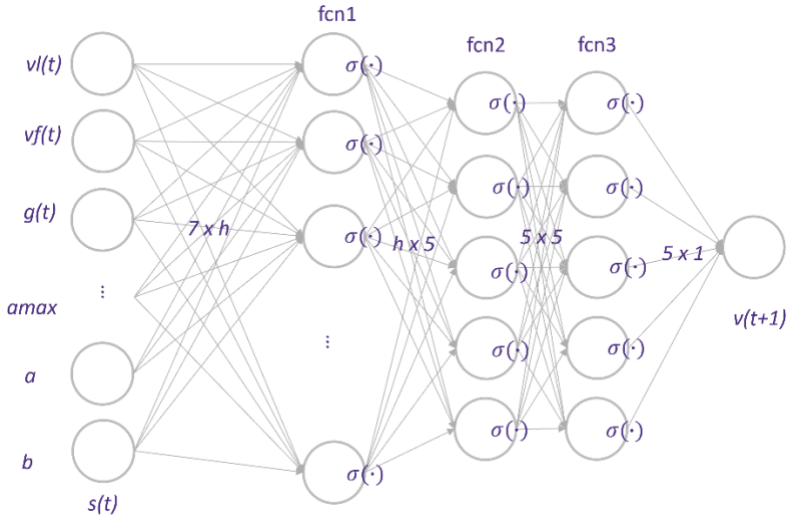


Figure 35: Neural Network Architecture

The loss function we adopted was squared error loss, as written as equation (5). In the loss function, we compared the predicted velocity $\hat{v}_i(t+T)$ and the actual velocity $v_i(t+T)$ of vehicle i, then we added up all the squared differences between the two. After using 10-fold cross-validation, we decided to use the learning rate 10^{-5} and the parameter $h = 8$ in the neural network. The training result can be seen in Figure 36. The final validation loss and the test loss were 2.403 and 4.240, respectively.

$$\min_{w} \sum_{i=1} \left(v_i(t+T) - \hat{v}_i(t+T)\right)^2 \tag{5}$$

However, features provided by the last two hidden layers did not have physical meanings. Feeding a set of information of another simulated vehicle pair to the neural network, we obtained many negative outcomes from the last two layers. Table 18 documents the outcomes from the second hidden layer. The result reflects that a neural network is powerful at fitting data; however, interpreting the black box is difficult. It also implies that the model yielded by a neural network could be a more difficult puzzle if we added more layers and neurons aiming to increase prediction accuracy. In fact, learning physical dynamics from data is a data-driven discovery question, which has been tackled for decades mostly by statistical models on understanding statistical relationships [34, 35]. However, statistical relationships between the input and the response are usually given by humans, meaning that training a statistical model can be limited by human knowledge. In other words, discovering the underlying structure of a dynamical system directly from data remains underexplored. The problem of discovering insights from data is worth further study.

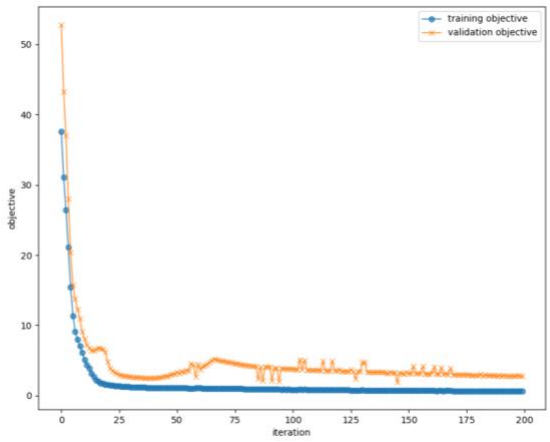


Figure 36: Training and Validation Results

Table 18: Outcomes of the Second Hidden Layer vs. Given Variables

The second hidden layer
10.680
-22.1601
-197.6575
-119.4152
35.4486

Input Variables	
v_s	12.533
$g(t)$	2.5
$a_{\max}$	2.6
b	4.5
$vl(t)$	12.533

Section 6: Discussion and Conclusion

The goal of this project was to establish a transportation simulation platform that can model connected and automated transportation systems at multiple scales. For this research, we proposed a framework for integrating traffic models that can simulate transportation systems at different scales. We used open-source transportation simulators, i.e., MATSim and SUMO, as the main simulation models. We also chose Unity to manage sub-microscopic visualization because Unity gives developers great freedom to create any gaming environment. This project developed a way to not only include the three model components in a platform but also established a communication system for them to interact with each other. Meanwhile, we chose the Greater Seattle area as the study region for implementing the developed VTD platform. The network datasets were calibrated to the observed traffic data. In summary, this project had three key parts: (i) development and implementation of the VTD multiscale simulation platform, (ii) calibration of the SUMO network, and (iii) calibration of the MATSim network.

The SUMO network was calibrated by following the system performance simulation calibration criteria provided by FHWA [36]. The SUMO simulation was built by using PSRC OD data for vehicle and pedestrian demands, and GTFS data for public transit. To conduct the simulation, this project used loop data from WSDOT and travel time data from NPMRDS for field measurements. Our calibration results showed that 80 percent of the traffic volumes and 50 percent of the travel times were well calibrated. The calibration process had several limitations. First, travel time calibration should be enhanced; one of the main issues for travel time calibration is a lack of auxiliary data resources. With only OD demand in hand, it was hard to speculate the traffic state for each road segment accurately. For future study, a project should attempt to find data for traffic volume calibration along the local street. Second, the data resource for the input OD data (2014) and the filed measurement data (2018) were from different years. Such variance may also play a role in calibration performance. Third, more calibration for SUMO should be conducted, such as route calibration and car-following model calibration suggested by the FHWA. The project team will continue to explore and gather more data for further calibrations.

The MATSim network was calibrated to well represent traffic in the real world. The Greater Seattle network combined network data from the City of Seattle and network data from the City of Bellevue, both published in 2020. The calibration of the MATSim network considered home-based work traffic demand extracted from PSRC OD data. During the project working period, the latest demand estimations by the PSRC were based on a 2014 survey. To be consistent with the demand data, we collected 2014 bus route data from the GTFS. To simulate demand in which either the trip origin or destination was outside of our study area, we deployed TAZ gates around the simulated region. The TAZ gates were selected from PSRC TAZ data. The latest TAZ dataset that we could access during our project working period was from 2010. On the basis of these, traffic links were calibrated to real traffic data by using average observed traffic volumes and speeds. Because of the SR 99 tunnel project [37], we

collected traffic volume data from SDOT and traffic speed data from INRIX recorded in May 2018. We mainly calibrated link speed limits and link capacities for the MATSim network. The network links were categorized into highway and arterial types. These two types of links were calibrated individually for link speed limits. The simulated speeds of both types of links were calibrated to be within a 10 percent difference from observed speeds. The capacity calibration focused only on arterial links. We chose 21 checkpoints on major streets. The calibrated traffic volumes were 11.55 percent different, on average, from the observed volumes. However, the calibration processes can be improved for the following reasons. First, datasets collected in this project were generated or recorded in different years because of the survey year and the tunnel project discussed above. With the year difference between traffic data and network data, traffic might have behaved differently from the real world in the study area even though there was only a marginal gap at the checkpoints. Second, the network calibration was mainly for the Seattle part and can be expanded. Third, capacity calibration for highway links remained undone. The calibration would have been closer to perfect if it could have been extended to the Bellevue part and included all types of network links.

The VTD platform was developed and implemented by a computer with 32 GB memory and an Intel Core i9-9900K CPU. The implementation of the VTD platform showed capabilities of traffic simulation at multiple scales. Regarding the design of the integration between the macroscopic simulation model (i.e., MATSim) and the microscopic simulation model (i.e., SUMO), the implementation showed that vehicles traveling between the MATSim and the SUMO areas can be simulated by this VTD platform. The communication between the microscopic simulation model (i.e., SUMO), and the vehicle simulation/visualization model (i.e., Unity 3D) was designed to be managed by the control center. The implementation also displayed that the vehicle simulation/visualization model can visualize the ego car and the surrounding environment. The implementation showed that the VTD platform has the potential to help simulate traffic behaviors in a study area at multiple scales. With the integration between the macroscopic simulation model and the microscopic simulation model, researchers can model traffic flows and, at the same time, vehicle interactions, e.g., lane-switching and car-following. Researchers can also observe vehicle interactions at the vehicle level by using the vehicle simulation/visualization model.

This platform can also help test and improve traffic control algorithms. We tested a multi-scale, signal-vehicle coupled control algorithm on this platform. The control flexibilities enabled us to design complex control algorithms and various indexes that can be collected and provided us with plentiful data to evaluate and improve the control algorithms. We also used the simulated trajectory data to learn the car-following model. Passenger vehicles were the only vehicle type considered in this car-following model learning task. The learning used a neural network that had three hidden layers. The neural network gave us a decent model. The nonlinear model, however, did not provide physical insights between the input and output variables. This result leaves a valuable research topic for the project team.

Section 7: Future Research Directions

Below we discuss potential future research directions for improving the calibration for the SUMO network, calibration for the MATSim network, the VTD platform development, and possible applications using the VTD platform. For the SUMO network, further calibration may consider different types of vehicles, such as freight. Apart from using the GEH for traffic volume and travel time calibration (system performance calibration), future calibration of SUMO can focus on more detailed features such as route calibration. As discussed in the previous chapter, the MATSim network calibration used traffic data from different years, and several of them had gaps of about six years from the network datasets. Furthermore, the calibration focused mainly on the City of Seattle. The network can be calibrated to the latest travel estimations on the basis of the updated survey data. Bellevue can also be included in the calibration process for the latest datasets. For the VTD platform development, we simplified the simulation features of the vehicle simulation/visualization model. This feature has the potential for future development. Specifically, the vehicle simulation/visualization model can further simulate CAVs equipped with onboard sensors, such as LiDAR. The CAVs in the vehicle simulation/visualization model can report the latest sensing data back to the control center. These data can then provide extra information for the control center for traffic control, including traffic signal control, vehicle control, or coupled signal-vehicle control. This feature may rely on techniques such as computer vision (CV) algorithms and simultaneous localization and mapping (SLAM).

Speaking of applications, the VTD platform can help provide insights into traffic dynamics. The VTD platform may not only help to verify existing traffic dynamics models but also make it possible to discover new or simplified models for traffic dynamics. More specifically, the VTD platform has the potential to test and verify link-level traffic dynamic models. Researchers can use the traffic data simulated by the VTD platform to further validate existing link dynamic models, such as the point queue model, link transmission model, and double queue model. In this potential analysis, a link-level fundamental diagram could further provide the relationship between the network link properties (e.g., capacities and speed limit) and the characteristics of the fundamental diagram (e.g., shape and slope). It could also be possible to learn traffic dynamics from the VTD simulated data using data-driven machine learning techniques.

For the multi-scale, signal-vehicle coupled control algorithm, the potential future research directions include the following: (i) testing the algorithm on a larger area and (ii) integrating Unity into the testing to simulate the vehicle dynamics more accurately and generate more detailed vehicle data. For the traffic dynamics learning task, the project team plans to continue to extract meaningful insights from the collected data. This research topic could be (i) developing a neural network that can capture physical relationships between the input and the targets, (ii) testing the neural network on multiple vehicle types, and (iii) testing on real data with more realistic vehicle dynamic models.

References

[1] J. X. Ban, Q. Guo, O. Angah and Z. Liu, "Vehicle-Traffic Control with Limited-Capacity Connected/Automated Vehicles," C2SMART, New York, 2020.

[2] K. W. Axhausen, A. Horni and K. Nagel, The Multi-Agent Transport Simulation MATSim, London: European Research Council (ERC), 2016.

[3] X. Zhou and J. Taylor, "DTAlite: A queue-based mesoscopic traffic simulator for fast model evaluation and calibration," *Cogent Engineering,* vol. 1, no. 1, 2014.

[4] PTV, "Multimodal Traffic Simulation Software," PTV Group, [Online]. Available: https://www.myptv.com/en/mobility-software/ptv-vissim. [Accessed 20 June 2022].

[5] abstreet, "A/B Street," 12 June 2022. [Online]. Available: https://github.com/a-b-street/abstreet. [Accessed 20 June 2022].

[6] Eclipse, "Simulation of Urban MObility," Eclipse SUMO, [Online]. Available: https://www.eclipse.org/sumo/. [Accessed 20 June 2022].

[7] I. T. Haman, V. C. Kamla, S. Galland and J. C. Kamgang, "Towards an Multilevel Agent-based Model for Traffic Simulation," *Procedia Computer Science,* vol. 109, pp. 887-892, 2017.

[8] A. Poschinger, R. Kates and H. Keller., "Coupling of Concurrent Macroscopic and Microscopic Traffic Flow Models using Hybrid Stochastic and Deterministic Disaggregation," in *Transportation and Traffic Theory in the 21st Century. Proceedings of the 15th International Symposium on Transportation and Traffic Theory*, University of South Australia in Adelaide, Austral, 2002.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to algorithms, MIT Press, 1994.

[10] K. C. M. Transit, "2014 Service Guidelines Report," King County, 2014.

[11] F. Poletti, "Public transit mapping on multi-modal networks in MATSim," IVT, ETH Zurich, Zurich, 2016.

[12] B. Y. He, J. Zhou, Z. Ma, D. Wang, D. Sha, M. Lee, J. Y. Chow and K. Ozbay, "A validated multi-agent simulation test bed to evaluate congestion pricing policies on population segments by time-of-day in New York City," *Transport Policy,* vol. 101, pp. 145-161, 2021.

[13] D. Ziemke, I. Kaddoura and K. Nagela, "The MATSim Open Berlin Scenario: A multimodal agent-based transport simulation scenario based on synthetic demand modeling and open data," *Procedia Computer Science,* vol. 151, pp. 870-877, 2019.

[14] M. Balmer, K. Meister, M. Rieser, K. Nagel and K. W. Axhausen, "Agent-based simulation of travel demand: Structure and computational performance of MATSim-T," *Arbeitsbericht Verkehrs - und Raumplanung 504,* 07 2008.

[15] S. Hörl, M. Balac and K. W. Axhausen, "Dynamic demand estimation for an AMoD system in Paris," *2019 IEEE Intelligent Vehicles Symposium (IV),* 09-12 June 2019.

[16] A. Horni, K. Nagel and K. W. Axhausen, "High-resolution destination choice in agent-based demand models," *Arbeitsberichte Verkehrs - und Raumplanung 682,* August 2011.

[17] J. C. Spall, "A stochastic approximation algorithm for large-dimensional systems in the Kiefer-Wolfowitz setting," *Proceedings of the 27th IEEE Conference on Decision and Control,* 1998.

[18] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization.," *IEEE Transactions on aerospace and electronic systems,* vol. 34, no. 3, pp. 817-823, 1998.

[19] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins apl technical digest,* vol. 19, no. 4, pp. 482-492, 1998.

[20] SUMO, "netconvert," 27 6 2022. [Online]. Available: https://sumo.dlr.de/docs/netconvert.html.

[21] SUMO, "Road Networks," 27 6 2022. [Online]. Available: https://sumo.dlr.de/docs/Networks/SUMO_Road_Networks.html.

[22] "SoundCast: the PSRC Activity-Based Model," 27 6 2022. [Online]. Available: https://github.com/psrc/soundcast.

[23] KCM, "Developer Resources," 27 6 2022. [Online]. Available: https://kingcounty.gov/depts/transportation/metro/travel-options/bus/app-center/developer-resources.aspx.

[24] "Synchro Studio," 28 6 2022. [Online]. Available: https://www.trafficware.com/synchro-studio.html.

[25] K. Udomsilp, T. Arayakarnkul, S. Watarakitpaisarn, P. Komolkiti, J. Rudjanakanoknad and C. Aswakul, "Traffic Data Analysis on Sathorn Road with Synchro Optimization and Traffic Simulation.," *Engineering Journal 21(6),* pp. 57-67, 2017.

[26] S. K. Singh, P. Komolkiti and C. Aswakul, "Impact Analysis of Start-up Lost Time at Major Intersections on Sathorn Road Using a Synchro Optimization and a Microscopic SUMO Traffic Simulation," *IEEE,* pp. Vol 6, pp. 6327-6340, 2018.

[27] "Traffic Lights - SUMO Documentation," 28 6 2022. [Online]. Available: https://sumo.dlr.de/docs/Simulation/Traffic_Lights.html.

[28] Trafficware, "Synchro Studio 10 User Guide," [Online]. Available: https://www.trafficware.com/synchro-studio.html.

[29] "Traffic Lights with NEMA Phases," 28 6 2022. [Online]. Available: https://sumo.dlr.de/docs/Simulation/NEMA.html.

[30] "Traffic Analysis Toolbox Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software.," 28 6 2022. [Online]. Available: https://ops.fhwa.dot.gov/trafficanalysistools/tat_vol3/sect5.htm..

[31] "Lane- or Edge-Based Traffic Measures - SUMO Documentation.," 29 6 2022. [Online]. Available: https://sumo.dlr.de/docs/Simulation/Output/Lane-_or_Edge-based_Traffic_Measures.html.

[32] G. Qiangqiang and X. J. Ban, "A Multi-scale Control Framework for Urban traffic Control with Connected and Automated Vehicles," *Submitted to Transportation Research Part B: Methodologies.*

[33] D. Krajzewicz, G. Hertkorn, C. Rössel and P. Wagner, "SUMO (Simulation of Urban MObility) - an open-source traffic simulation," in *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM20002)*, Berlin-Adlershof, 2002.

[34] S. H. Rudy, S. L. Brunton, J. L. Proctor and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science Advances,* vol. 3, no. 4, 2017.

[35] S. L. Brunton, J. L. Proctorb and J. N. Kutz, "Discovering governing equations from data by sparseidentification of nonlinear dynamical systems," *Proceedings of the national academy of sciences,* vol. 113, no. 15, pp. 3932-3937, 2016.

[36] "Traffic Analysis Toolbox Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software," [Online]. Available: https://ops.fhwa.dot.gov/trafficanalysistools/tat_col3/sect5.htm. [Accessed 29 6 2022].

[37] WSDOT, "SR 99 tunnel," [Online]. Available: https://wsdot.wa.gov/construction-planning/search-projects/sr-99-tunnel. [Accessed 20 6 2022].

[38] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences,* vol. 104, no. 24, pp. 9943-9948, 2007.

[39] M. Schmidt and H. Lipson, "Distilling Free-Form Natural Laws from Experimental Data," *science,* vol. 324, no. 5923, pp. 81-85, 2009.

[40] J. Koza, Genetic Programming: On the Programming of Computers by Meansof Natural Selection, Cambridge, MA: MIT Pres, 1992.

[41] T. Back, D. B. Fogel and Z. Michalewicz, Evolutionary Computation 1: Basic Algorithms and Operators, CRC press, 2018.

[42] B. K. Petersen, M. L. Larma, T. N. Mundhenk, C. P. Santiago, S. K. Kim and J. T. Kim, "Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients," in *International Conference on Learning Representations*, 2021.