Evaluating Third-Party Impacts in Urban Air Mobility Community Integration: A Digital Twin Approach

Alexander Ireland

A Thesis

in

the Department

of the

Concordia Institute for Information Systems Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

November 2025

© Alexander Ireland, 2025

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Alexander Ireland

Entitled: Evaluating Third-Party Impacts in Urban Air Mobility Community Integration: A Digital Twin Approach

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____ Chair & Examiner

Dr. Ali Ayub

_____ Examiner

Dr. Di Wu

_____ Supervisor

Dr. Chun Wang

Approved by _____

Dr. Chun Wang
Chair of Concordia Institute for Information Systems Engineering

_____ 2025 _____

Dr. Mourad Debbabi
Dean, Gina Cody School of Engineering and Computer Science

ABSTRACT

Evaluating Third-Party Impacts in Urban Air Mobility Community Integration:
A Digital Twin Approach

Alexander Ireland

Future Urban Air Mobility (UAM) operations present unique impacts on communities since UAM vehicles will operate primarily in populated areas. This shift from traditional aviation that typically operates point-to-point between urban areas, means that the impacts on third parties are increasingly more important. Safety, privacy, and nuisances such as noise all have third-party outcomes that should be explored to properly design a UAM transportation system that works for users and non-users alike. This thesis explores the use of a Digital Twin to minimize third-party safety and privacy impacts and shows that with accurate live population or mobility data, EVTOL vehicle flight planning is possible that considers where people currently are and adjusts the approach flight path accordingly.

A Digital Twin Prototype was developed to represent population density and live traffic data as an equivalent agent-based simulation of the physical world, which becomes the basis for assessing safety and privacy. A case study covering a portion of downtown Montreal is performed for multiple pedestrian and vehicle traffic settings to compare an EVTOL baseline vertiport approach suggested by the Federal Aviation Administration to 127 alternate vertiport approach scenarios to explore generalizations and demonstrate Digital Twin technology for the optimization of UAM operations.

It was found that flight path guidelines provided by aviation regulators will not necessarily provide optimal flight paths over urban areas when considering third-party impacts, and it was shown that Digital Twin technology is promising and may play a significant role in promoting community safety and privacy for UAM operations.

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Chun Wang for the time and effort that he spent supporting me on this educational journey. I am forever appreciative that he took a chance on me despite my desire to continue to work full-time outside of Concordia during the master's. His guidance throughout was invaluable and I especially appreciate how he was able to find opportunities for me to challenge myself by writing articles and other academic activities.

I would also like to thank Dr. Yong Zeng who I met in 2016 when he was a fantastically active and present academic partner for an industrial R&D project I worked on at CAE. Thanks to this connection, he provided me with excellent advice on my options for pursuing graduate school and introduced me to professors and the research in CIISE, which turned into this master's program.

Finally, I am deeply grateful for the support that my family, friends, and especially my wife Amália have given me, supporting me throughout. Knowing that I was not in this alone gave me the drive to push on and finish my thesis.

Contribution of Authors

This thesis contains excerpts or content similar to two as-yet unpublished works with multiple authors where the primary (first author) is the author of this thesis. The two works are as follows:

Book: Next-Generation Cities Institute Encyclopedia

Chapter:	Equitable Community Integration of Urban Air Mobility: Challenges, Approaches, and Opportunities,
Submission:	December 2023
Co-Authors:	Alexander Ireland, Sarah Farahdel, Nima Moradi, Chun Wang, Fereshteh Mafakheri, Anjali Awasthi
My Role:	As main author, the vast majority of the content was written by me. The transportation equity literature review was written entirely by me, and it is reproduced in-line, verbatim, in section 3.1 of this thesis.

Conference: Winter Simulation Conference 2025

Paper:	Evaluating Third-Party Impact in Urban Air Mobility Community Integration: A Digital Twin Approach
Submission:	April 2025
Co-Authors:	Alexander Ireland, Chun Wang
My Role:	As main author, this peer-reviewed article submitted for publication and presentation at the conference was written nearly entirely by me. Editorial and other suggestions from the co-author were incorporated. The content of this article is similar to various content presented in this thesis, especially sections 5.1 to 6.4 since the same metrics are presented and the same case study was used. While the content and results are similar, none of the content was copied verbatim.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Urban Air Mobility (UAM) and Advanced Air Mobility (AAM) are recently popularized terms that describe the use of low-level airspace for the transport of people, goods, and a variety of other non-transport applications using Unmanned Aerial Vehicles (UAV) and Electric Vertical Take-Off and Landing (EVTOL) vehicles that are capable of new operations that traditional aviation is not well suited for. UAM refers to operations that occur primarily above urban/sub-urban settings, while AAM is a broader term that refers to air mobility operations that include those of UAM but may operate regionally as well. For reasons that will become clear this thesis will focus on UAM, but many sources cited will refer to AAM.

A commonly repeated remark is that flying is the safest form of travel. Since the first powered aircraft took flight in the early 20th century, the traditional aviation industry has continually improved. With over 100 years of advancement to technology, operations, regulation, and safety it really is statistically the safest form of travel[1]. Urban Air Mobility as a transportation system by contrast is in its infancy so there are many unknowns and questions to be answered, most importantly to this thesis, how will EVTOL vehicles integrate into urban communities [2]. Many other unanswered questions as foundational as what will be the "rules of the sky", how and where will EVTOLs and UAVs be allowed to operate, and how will the traditional aviation industry be affected, all remain to be investigated. Traditional civil aviation is highly regulated around the world, meaning changes will likely be slow and conservative in nature. The International Civil Aviation Organization (ICAO) is an agency of the United Nations established by the Chicago Convention for fostering cooperation amongst signatory countries to share their skies for civil aviation purposes[3] . It maintains civil aviation Standards and Recommended Practices (SARPs) that are further developed by National Aviation Authorities (NAAs) (also called Civil Aviation Authorities) into legal regulations governing aviation in each jurisdiction [4, p. 52]. Some important examples of NAAs are the European Union Aviation Safety Agency (EASA) in Europe, the Federal Aviation Administration (FAA) in the United States, and Transport Canada (TC) in Canada.

Organizations such as these and others have begun to create UAM/AAM Concepts of Operations[5], [6], Blueprints [7], Playbooks[8], as well as regulation proposals for vehicle certification circulars[9], and infrastructure standards (such as vertiports[10]). UAM is an example of technology-lead change. With the recent improvements to battery technology, it has become feasible to fly with electrically driven propulsion systems instead of direct-drive fossil fuel engines, which greatly opens the design possibilities of new air vehicles. There are many designs with many different capabilities being developed [11] so designing policy and regulations is challenging, but offers opportunities for research that will help to guide policy makers and regulators.

UAM operation is a paradigm shift from traditional aviation. Commercial and other civil aircraft generally fly between airports point-to-point. Airports are often co-located with populated areas, but most of the flight time is not spent flying over densely populated or urban areas. Considering this, when incidents happen, it is easy to imagine that the people most affected are the people on board the aircraft. The term used for the risk to these users (participants) is first-party risk, which contrasts with the term third-party risk that describes the risk to non-users[12]. UAM will operate primarily above populated and urban areas, which highlights the importance of managing risk and other impacts to third-parties including pedestrians, ground vehicles, building occupants, and anyone else impacted by the UAM operations that themselves do not

participate in it. This is an easy to understand concept and one that could be susceptible to strong emotional responses by the public, so many sources discussing UAM community integration have conceptually identified various possible third-party risks [2], [7].

Modeling and simulation are powerful tools for designing systems. Agent-based modeling is a common simulation technique where individual agents can interact independently with their environment and other agents to perform tasks they are given to achieve their objectives. Complementarily, Digital Twins allow the bidirectional flow of information between the Physical world and the Virtual Twin (the modelled digital world [13]). Combining these techniques enables real-time decision making as well as alternative scenario simulation.

There are currently millions of UAVs in operation globally, but there are as-yet no EVTOL vehicles that are certified for flight by TC, EASA, or the FAA. The concepts in this thesis are relevant for both EVTOL and UAV operations, but the case study is designed around EVTOL operations.

1.1. Research Purpose

With the uncertainties surrounding the integration of EVTOL vehicles into urban environments coupled with the lack of operational data, this thesis aims to develop modeling and analysis tools that can assist a safe and equitable community integration of UAM.

The purpose of this research is to develop a prototype of a Digital Twin capable of modeling EVTOL vehicle operations near vertiports using agent-based and discrete event modeling techniques. The Digital Twin aims to measure some of the impacts that EVTOL operations have on third parties to support early policy decisions and community integration planning. The thesis is based around following research objectives:

- To use real-world data, in the creation of the virtual twin simulation environment including live data representing third parties.
- To model third parties using agent-based modeling techniques enabling person-centric metrics.
- To evaluate third-party impacts and use these metrics to optimize flight approach paths and provide analysis and guidance of policy decisions.
- To demonstrate resulting EVTOL flight operations in an engaging and presentable format.

This work sets an early foundation for policy makers, regulators, and the UAM industry interested in tools available to assess how their decisions will impact the future community integration.

1.2. Contribution

The main contribution of this research is the creation of an Urban Air Mobility Digital Twin Prototype that creates a dynamic agent-based representation of the people in a community (pedestrians, vehicles, and building occupants) based on real-world data and relates them to EVTOL operations at the individual level. The Digital Twin Prototype is able to evaluate multiple scenarios and choose the optimal flight path that minimizes the third-party impacts.

1.3. Thesis Outline

The thesis is structured as follows. Chapter 2 elaborates on the problem this thesis is addressing. Chapter 3 is a literature review of the four key elements of this thesis, transportation equity, Third-party impact evaluation, Digital Twins, and Aviation Standard and Regulations. Chapter 4 describes the methods used to construct the digital twin simulation. Chapter 5 details the third-party metrics developed. Chapter 6

presents the case study and results for an area in downtown Montreal. Chapter 7 includes the concluding remarks and suggestions for future research.

Chapter 2

Problem Statement

2.1. Problem Context and Definition

Designing a new transportation system that works well with (and complements) existing transportation systems is not something that happens often. Other transportation systems like the motor vehicle road network, or the aviation industry have benefited from over a hundred years of refinement. Regulations, policies, and laws evolve over time as technologies advance and incidents occur that necessitate changes. Urban Air Mobility is similar in principle to general aviation, in that it exploits the large open airspace above the ground, but the usage cases, flight locations, and possible scale of operations requires many changes from traditional aviation. Many competing interests exist and would benefit from a UAM transportation system that better matches each of their interests. Online retailers likely want to ship their packages to customers as quickly and cheaply as possible. Ride-hailing services likely want to have as many unpiloted EVTOL vehicles in the air as possible to provide faster and better service than their competitors. Emergency services likely want priority operations to save more lives. The wealthy and powerful may want a priority/executive tier of travel that gives them an advantage over others simply because they can afford it. Many people likely want their future flying cars to take them anywhere they want, as easily and simply as possible. Conversely, many people also may not like the idea of vehicles flying near or directly over them or their property [14].

Unlike traditional aviation, UAM will operate primarily above populated areas, which means that community integration is an important topic to consider. NASA recently released a playbook for advanced air mobility community integration that does a thorough job of describing some possible community integration challenges including Institutional Readiness, Equity, Community Engagement and six others[15]. Each of these integration challenges has many research opportunities. The chosen integration challenge for this master's thesis is related to equity (fairness of the system), specifically some of the negative impacts imposed on the community (third-party impacts) when in close proximity to UAM operations.

2.2. Challenges

Studying third-party impacts comes with many challenges. Firstly, there needs to be data that can be used to accurately describe non-users of UAM. In today's western world, private companies are the entities that have free access to individualized data that they collect for where people are and what they are doing. For example, anyone carrying a cellphone that is connected to and using Google services is providing their live location and other information to Google [16]. Cell-phone networks also have a record of every cellphone in their coverage area so anyone carrying a cellphone with radios turned on is sharing its location with companies like Telus [17]. Other companies with popular apps that are phone platform and carrier agnostic also have access to the information about their users. Meta for instance, that currently owns Facebook, Whatsapp, and Instagram, has their apps installed on the phones of a significant percentage of the population in Canada [18]. This raw population data is not openly available to other researchers, instead some companies provide free or subscription access to APIs that contain processed, randomized, or otherwise modified data that represents population data in proprietary formats for pre-determined usage. This lack of

readily available raw live population data means that efforts must be made to work-around the available processed data in a way that achieves the research objectives.

The second challenge is that UAM operations are not yet standardized or defined. There exists no operational data for UAM that can be used for analysis. Regulations and policies are still being defined. In some cases, UAM operations are problematic due to constraints imposed by existing airspace rules, for example within 5 nautical miles of airports [19]. Urban/suburban communities are often located with 5 nautical miles of airports. Regulators around the world have started to propose regulations that can be used as the basis for analyses. For instance, the FAA has released an engineering brief [20] related to Vertiport design, which is based heavily on helicopter operations despite some of the envisioned operations would not fit neatly into these schemes. Evaluating third-party impacts is difficult when basic data about the transportation system is not available. This includes performance characteristics of vehicles and safety considerations for vehicle designs (e.g. Target level of safety (TLOS), or historical incident data, considerations for malfunctions, adverse weather conditions, etc).

The third challenge is a result of the first two. While there have been some attempts at evaluating third-party impacts for UAM, the domain-specific body of knowledge is limited, and due to the differences in available data, the research methods are not standardized. Assessing third-party impacts in analogous systems such as drone operations and traditional aviation has been explored, but applying similar techniques depends heavily on the available data and what portions of the research are applicable to UAM. For instance, with the new paradigm of highly maneuverable vehicles that can take-off and land vertically, hover, and maneuver in any direction raises some outcomes that have unique aspects when applied to UAM such as safety, privacy, and nuisances such as airborne noise. These outcomes exist for UAV and traditional aviation but must be tailored to the UAM reality.

2.3. Consequences

The enabling technologies driving EVTOL vehicles are continuing to improve quickly while the availability and capabilities of UAVs is creating an environment where technology is leading ahead of regulations. It is only a matter of time before EVTOL vehicles are certified for flight in Canada. When technology-driven change leads ahead of policy and regulatory change, there is a change of negative impacts. For example, when ride-hailing and short-term rental apps became available, the taxi and hotel industries were affected because these new technology-driven services were able to operate without the same restrictions put on the existing heavily regulated industries. If research is not conducted into creating a fair system from the beginning, it will likely necessitate a reactionary response after these unforeseen negative impacts begin to materialize.

2.4. Research Gap and Significance

This master's thesis mixes systems design concepts with policy evaluation using a novel agent-based digital twin framework that decouples the format/nature of the population data from the generated metrics enabling future research opportunities, independent of the available input data. Third-party metrics for safety, and privacy will be proposed, along with an integrated metric capable of providing insights to optimize flight paths that minimize third-party impacts and evaluate proposed UAM transportation system regulations and provide guidance on alternatives. A notable exclusion is that of third-party noise and other nuisance impacts. These impacts are more specific to the individual vehicle designs so they are not explored in this research.

Chapter 3

Literature Review

The initial idea behind this research was to provide a means to evaluate policies that are currently being written for UAM with the goal of capturing some aspects of fairness (equity). When designing an entirely new system for the movement of people and goods that will harness a public resource (the volume of airspace above communities), it is important that the transportation system is fair from the beginning. A literature review of Transportation Equity was performed. From this, one aspect of Transportation Equity that aligned well with the author's interests was External Costs, which can also be described as third-party impacts or risks. The term third-party describes people who are not participating in an activity, so in this case, third-party describes anyone that is not participating in UAM, but who could be impacted by UAM operations. A literature review was conducted to understand where current research directions are going in this domain, and a key element was found to be simulation and modeling. It became clear that an interesting and growing field of research uses Digital Twins to recreate the physical world digitally in order to derive some meaning and provide feedback to the physical system to improve or control it. The last element of the literature review was to understand how aviation policies, including standards and regulations are created, but more importantly, how research can be used to guide these policy decisions.

In summary, literature reviews were conducted in the domains of Transportation Equity, Third-party Impacts in Aviation, Digital Twins, and Aviation Standards and Regulations. These 4 literature reviews are presented below and cover the main elements of this research.

3.1. Transportation Equity

The following literature review on transportation equity (comprising the remainder of section 3.1, beginning after this explanatory paragraph) was written entirely by the author of this thesis. It is included here verbatim from the future publication: Next-Generation Cities Institute Encyclopedia, Volume 03, Part 01, Chapter 08 - Equitable Community Integration of Urban Air Mobility: Challenges, Approaches, and Opportunities [21].

The definitions and terminologies used to describe equity as it relates to transportation systems are varied throughout literature. The Oxford English Dictionary's general definitions for the word equity include references to being equal, fair, impartial, or right, as well as relating to the principals of justice [22], which is easy to understand conceptually but difficult to quantify tangibly to transportation systems outcomes [23]. This difficulty leads to a wide variety of interpretations of equity throughout policy and literature, so a common understanding of equity as a singular concept is not available.

Lewis et al. (2021) provide an in-depth analysis and review of how equity is defined across transportation research and literature where they use the foundational work of Robbins (1932) [25] to describe equity using Robbins' categorization of "assessments of the world as it is (positive) vs. the world as it ought to be (normative)" [24, p. 2]. Their exhaustive review of equity in transportation was further broken into 14 theories: Simple Equality, Formal Equality, Proportional Equality, Utilitarianism, Libertarianism, Marx, Smith, Paretto, Rawls, al-Sadr, Capabilities Approach, Sufficientarian, Prioritarian, Intuitionism. The complexity and broadness of equity theories led them to conclude that any individual equity study or research for transportation does not need to encompass all theories but should clearly scope the interpretation of equity used.

A popularly cited approach to transportation-related equity *"*refers to the fairness with which impacts (benefits and costs) are distributed*"*[26, p. 1] which can be further categorized into horizontal and vertical equity. The paper has been updated frequently and published in different formats, but as of 2022, the definitions presented for horizontal and vertical equity are: "Horizontal equity assumes that people with similar needs and abilities should be treated similarly. Vertical equity assumes that disadvantaged people should receive favorable treatment." [27, p. 44]. These two categories as presented in Litman (2022) are further split into 5 objectives that can be used to encompass the meanings of equity most often described in literature; for horizontal equity the objectives described are 1) fair share, and 2) external costs (i.e. the burden imposed on others), and for vertical equity the objectives described are 3) inclusivity, 4) affordability, and 5) social justice.

To understand the challenges that come with the consideration of equity into transportation systems, surveying transportation planners and transportation system experts is a necessary first step. Cantilina et al. (2021) surveyed 59 transportation practitioners from 4 different sectors; public, private, academic, and non-profit and found that 93% say they address issues of equity in their work. The most identified challenges or barriers faced by them were a lack of data (80%), a lack of access to data (78%), a lack of standards or metrics for transportation equity outcomes (61%), minimal legislative support or public process (56%), and a lack of data integration and analysis tools and skills (51%). The top two most identified approaches were collaboration with other organizations and other sectors (84%), and using data not directly related to transportation (80%) [28].

Transportation plans that include equity goals or objectives are an integral step in building equity into the transportation systems, but without measures for the equitable outcomes, the goals and objectives have no target to achieve. Manaugh et al. (2015) reviewed long-range transportation plans for 18 cities in North America that had been published from 2005 to 2015 to look for any indicated goals, objectives, and measures and found that while 14/18 mentioned at least one goal and/or objective related to equity, only 9/18 indicated applicable measures for the equitable outcomes [23].

In addition to transportation plans, some governments and organizations create programs specifically aimed at addressing and improving transportation equity. Van Dort et al., (2019) provides extensive analyses of 24 equity promotion programs, In addition to an in-depth analysis of the programs, they provide a noteworthy categorization of those meant for "addressing inequities of transportation system" vs. those meant for "addressing social inequities via transportation" [29, p. 24].

When it comes to infrastructure project decisions, two common methodologies used are Cost-Benefit Analysis (CBA) and Multi-Criteria Analysis (MCA). For CBA, when the concepts of equity are added, Thomopoulos et al. (2009) concluded that CBA, and its derivative methodology Social Cost-Benefit Analysis (SCBA), are not well suited because they require a monetary value to be included in the analysis, while MCA is better suited to include social impacts and there are an extensive number of MCA methodologies available to be tailored for a particular use case [30].

The evaluations of specific concepts of transport equity have most commonly focused on accessibility and to that end, the Gini Index is the most popular index used in those evaluations [31]. The Gini index was originally created in 1912 as a published work in Italian by Corrado Gini where it derived statistical formulae for measuring the variability (for quantitative phenomena) and mutability (for qualitative phenomena) of data distributions and was later related to Lorenz curves [32]. Various researchers have used the Gini Index and Lorenz curves to measure transportation equity outcomes. Some examples of this include; Delbosc & Currie (2011), who created a measure of equity performance for the transportation system as a single value [33], Jang et al. (2017) who measured spatial equity in public transit in the city of

Seoul [34], Hosein Mortazavi & Akbarzadeh (2017) who measured what they called “Connectivity power” [35, p. 47] reflecting public transit service, or Pritchard et al. (2019) who evaluated job accessibility when incorporating bike-and-ride into the motor-vehicle and public transportation system [36].

Aside from accessibility, other transportation equity outcomes that are commonly studied include safety, and environment [31]. This is supported by Guo et al. (2020) who after performing an extensive review of transportation equity assessment literature suggest that equity related outcomes commonly fall into accessibility, traffic emissions, and safety where the majority can be characterized within a proposed 3-step framework of population measurement, cost/benefit measurement, and inequality measurement [38].

Optimization of the transportation system to achieve various outcomes has been an active field of study for decades. To this end, the network design problem (NDP) is common in transportation research as its goal is to optimise the functioning of the network based on constraints [39]. NDP has many applications, including the transit network design problem (TNDP). Ibarra-Rojas et al. (2015) provided an in-depth literature review of TNDP research going back to the 1970s [40], where, as pointed out by Camporeale et al. (2019) only 2 sources were cited that included equity aspects [41]. Since 2015, incorporating equity into the TNDP has become more popular, with multiple sources investigating novel methods. For instance, Camporeale et al. (2019) created a two-step method to quantitatively include a combined horizontal and vertical equity measure as a constraint [41]. Similarly, Behbahani et al. (2019) created a similar framework whereby equity is modelled as a constraint but with the flexibility to use different equity theories (e.g. utilitarianism, Rawls, etc.) in defining of the constraint [42]. Other recent studies [43], [44] have been conducted with their own approaches to implementing equity into the TNDP problem.

In general, transportation equity research has been aligned with positive assessments of existing systems using available data.

3.2. Third-party Impacts in Aviation

When referring to risks or impacts on people, first-party risks are those where the user/participant of a system has accepted any personal trade-offs related to costs and benefits whereas third-party risks refer to the risks that are imposed by the system onto a non-user/non-participant of a system [12]. Research into third-party impacts or risks in aviation has been an area of research for decades [45], but has been overshadowed by the aircraft-centric aviation safety research including a large number of past and current research studies into airspace management including Air Traffic Management(ATM), Air Traffic Control(ATC), and separation strategies.

Recently, interest in aviation-related third-party risk research has grown where third-party safety and privacy impacts have been explored for UAM, Unmanned Aerial Systems (UAS) and traditional aviation. Various strategies have been employed to evaluate third-party risks in the UAM/UAS domains. There has been research into the generation of risk metrics in various forms based on the nature of the studies and available data, research into optimization of flight paths that minimizes risk metrics, as well as propositions for frameworks, service models, and use cases.

In the operational domain, proposals for how to use third-party metrics as a tool for operators have been explored. Ancel et al. proposed a service provider meant to assist UAS operators in the pre-flight process by analyzing their flight plans and providing information about third-party risks so that the operators can choose to modify their flight plans accordingly [46]. The service provider used a combination of techniques including a referenced risk assessment model [47] that was comprised of a probabilistic graph model, in-air flight failure/crash trajectory model, and a severity estimation model based on population density data.

Their described risk assessment model and therefore the service provider that uses it is considers only casualty risk. In this case, the research does not describe any specific path planning algorithms.

A commonly used approach to calculating metrics is to use a grid in 2 or 3 dimensions to segment a study area into small areas (or volumes) that can be used to calculate area-specific risk factors [48], [49], [50], [51], [52]. The path is then optimized to achieve the minimized risk cost value. This strategy is valid in some operational situations for UAVs because many are highly maneuverable and can perform complex flight paths with ease, at the cost of longer flight times and more expensive operations. These tradeoffs provide for interesting results. The optimization strategies are similar but have some key differences. Recently Tang et al. created an approach where they optimized UAV flight paths in 3-dimensions using a min-cost A* algorithm while considering damage to buildings and injury to ground personnel in their cost formulas [53]. A* algorithms are commonly used. Another example is Celdran Martinez et al. who also created an approach to optimizing 3-dimensional UAV flight paths using a machine learning model while considering first and third-party impacts such as safety, privacy and noise [54]. The similarities of these examples illustrate the unique opportunities available to route UAV flight paths when limiting flight path complexity is not a requirement in order to optimize UAV operations. EVTOL vehicle operations are inherently different. They are not as maneuverable as small UAVs can be, and the human factors will drive the decision that complex 3-dimensional flight paths will likely not be a viable solution. Many EVTOL designs achieve aerodynamic lift from wing surfaces during forward movement in some flight modes [55] for efficiency purposes, so complex flight paths may not even be feasible. Traditional aviation is generally operated point-to-point with relatively simple waypoints so an opportunity exists to research optimizing flight paths that use some design constraints from traditional aviation, such as exploring final approach flight using well established paradigms such as using polar coordinates (heading, descent rate, distance).

A common challenge for estimating third-party risks is the nature and use of population data. The format and availability of acceptably high-resolution third-party data varies as can be seen in the past research. Choi et al. used live population data available through the City of Seoul open data platform [56] to create a 250m x 250m grid of population density segments, further combining them with a ground coverage model that included the ratio of areas buildings occupy to open areas, and finally using building height data and a aviation vehicle crash area model described by Ale and Piers [45] to create a risk model that characterizes the risk level in each grid segment so that a flight can be planned to minimize the third-party risks[51].

The most studied risk is that of an aircraft crash. Crash area modeling for traditional aviation has been extensively studied and there are various approaches developed to predict the crash area size and shape. Modeling generally uses geometric or weight-based approaches. An in depth review performed by Melnyk et al. concluded that weight-based approaches are the most accurate [57], specifically they cite a study by Ale and Piers that shows a linear relationships between aircraft weight and the impact area [58]. They further show that these relationships can be split into impacts in open areas and impacts in built areas. The data used by Ale and Piers uses crash data from traditional aviation, so the accuracy of applying these linear relationships to UAM vehicles is unknown because key differences exist such as fuel vs batteries, or flight differences between VTOL and non-VTOL aircraft.

Vehicle crashes are not the only third-party safety risk that has been identified. It is easy to predict that an object may fall from a UAM/UAS vehicle for a variety of reasons. Falling objects pose a third-party risk that has not been well explored in the literature. A bird striking the vehicle and falling to earth happens regularly in traditional aviation [59] but has a limited risk due to the limited flight time of traditional directly over urban areas. In-air collisions with other vehicles or small UAVs [60] pose a greater risk as the number of vehicles sharing the airspace for different purposes increases. Icing is a common occurrence in aviation, and the impacts are normally thought of in terms of degraded performance or risk to the vehicle, but ice

shedding and falling in an urban environment is a significant risk as well [61]. Finally, there could be any number of other reasons an object may fall or be ejected from the vehicle.

Privacy is another topic where the third-party impacts have been discussed, but their impacts have not been thoroughly explored. Celdran Martinez et al. studied third-party privacy risks based around the concept that UAVs have cameras (that can record) with a specific resolution which, based on distance, dictates whether a person can be recognized and provided a simple binary metric based on distance [48]. Similar camera resolution-based approaches were utilized in other studies as well [49]. Fitwi et al. take a different approach and proposes a method to remove personal information from video recordings by detecting windows and visually scrambling the recorded data seen through it [62]. Even though third-party privacy risks have not been thoroughly studied in the context of UAM, the need is apparent. Hu et al. extended a technology acceptance model with risk and trust theories and made the case that privacy risks are one of a section of important considerations for the acceptance of UAM [14]. Similarly, Simic et al. perform a meta-analysis on societal acceptance of UAM and found that about 1/3 of the sources highlighted third-party privacy as an area of concern. They further sub divide the privacy concerns into 5 categories which are a mix of the themes; UAV trajectory, impact area, impact exposure, impact duration, and general annoyances [63].

3.3. Digital Twins

Digital twins are a technological concept that has grown in popularity in recent years. A keyword search for "digital twin" on Scopus shows the evolution of the popularity of the term from 0 per year in the year 2000, to about 10000 per year in 2024. Figure 1 shows graphically the nearly exponential growth since about 2015.

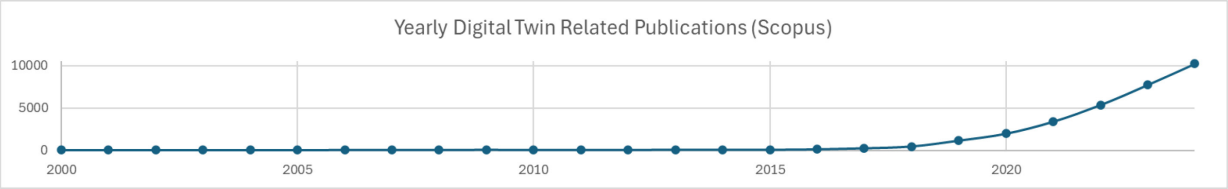

Figure 1- Number of yearly publications including the concept of a Digital Twin as listed on Scopus

The terminology "Digital Twin" has an interesting history. According to Grieves and Vickers the concept was first described by Grieves in 2002 in the domain of product lifecycle management where a mirrored set of real space and virtual space were linked via bidirectional communication of information, at the time being called Mirrored Space Model. The concept matured and expanded in the domain, later referred to as Information Mirroring Model. In 2011 the first usage of the term Digital Twin was employed to describe this type of system [64]. Other researchers, Singh et al, claim that similar concepts were devised even earlier [65] such as the concept of " Mirror worlds" where in 1993 Gelertner describes how software will be used to derive insights into a physical system and when interacting with a virtual representation of a system, you are also interacting with the physical version of it[66].

Despite this history, the terminology Digital Twin was used in unrelated contexts prior to it becoming the generally accepted term for the specific type of conceptual model. For instance in 2005, Wang et al. used the term for describing a method related to human motion [67]. In 2010, Puig and Duran used the term to refer to a virtual replica (an avatar) of a person [68]. In April 2011, Tuegel et al. used the term for a virtual copy of an aircraft's structural components to predict the aircraft structural lifespan [69]. This third example highlights two questions about Digital Twins; 1) is there a generally accepted definition of what a digital twin is? 2) how does a Digital Twin differ from a simulation model?

With the rapid growth of the usage of Digital Twin, the meaning of the term has expanded and evolved causing the blurred line between a simulation model and a digital twin to become better defined. A recent review by Wright and Davidson highlighted the differences between a model and a digital twin. They argue that vagueness of the definition of a Digital Twin may eventually cause disillusionment and therefor cause interest in the technology to not reach its full popularity or potential [70]. Another recent review by Singh et al. attempted to categorize the technology into various types and classifications[65].

When data for an entire physical system is not available, modelling techniques (like agent-based modeling) can be used to complement the data that does exist. For systems with a data maturity level such as this, they can be considered Augmented Digital Twins[71]. Similarly, digital twins are used for systems under development in order to aid in maturing the system design before the physical twin exists in its final state. This predictive nature is a useful tool in design because it can predict behaviors that the physical system will exhibit. This predictive type of digital twin is known as a Digital Twin Prototype. [72, p. 95].

There continue to be many interpretations and definitions for digital twins but a summary of commonly accepted themes are 1) the existence of a physical system, 2) an accurate virtual (digital) representation of the physical system, 3) an information flow at an appropriate frequency from the physical system to the virtual representation to keep it up-to-date, 4) the use of the virtual representation to derive insights about the physical system through analysis, evaluation, etc., 5) a means for the virtual twin to communicate back to the physical system in order to modify or improve it in some way. These 5 themes form the basis for the definition of Digital Twin as used in this research. In summary, a digital twin can be defined as a system that communicates data bidirectionally in real time (or near real-time) between a physical system (physical twin) and a virtual representation of the physical system (virtual twin). The insights gained from analysis performed using the virtual twin provide useful insights that are communicated to and used to modify the physical twin in some useful way.

Digital Twins have been developed in various domains and for various purposes. Many recent reviews have been conducted in how Digital Twins are used in domains such as healthcare[73], medicine[74], manufacturing[75], supply chains[76], and countless others, but more relevant to this research, domains such as intelligent transportation systems[77], aviation[78], and advanced/urban air mobility[79].

UAM/AAM research is a popular and growing field of study utilizing Digital Twin technologies, much of it related to UAS with relatively less research on EVTOL operations. A recent literature review by Zhang et al. verified the number of UAM/AAM Digital Twin peer-reviewed English-language publications published each year over the past 5 years has increased from 8 in 2020 to 76 in 2024 [79]. There are many applications that UAM/AAM-related digital twins have been successfully studied including traditional aviation (structural, manufacturing, aircraft systems, diagnostics, failures), Risk assessment and safety, flight and mission planning, UAV swarm coordination, and cyber security, and niche industry uses. The modelling approaches are varied based on the availability and format of data and tools, but generally fall into the following four dimensions; geometric modelling, physical modelling, behavioral modelling, and rule modelling [79].

Representing third parties in digital twin ecosystems requires accurate and highly detailed urban population density data. Census data can generally be used as an approximation, though it lacks the accuracy of live data. There has been examples of live population density information being used for third-party impact analysis in UAM [51]. A significant hurdle for researchers is access to live high-resolution population density as its availability and form is limited due (likely) to concerns about security and privacy. The data exists with data providers including mobile phone carriers which have access to triangulated device locations for all devices connected to their networks, as well as device and software companies that report

back device location to the service providers. A useful alternative is to mix census data, live population density data, and simulation. Pinto Neto et al. used this strategy and discrete-event simulation to evaluate hundreds of trajectory possibilities to evaluate efficiency and safety of UAM operations [80].

3.4. Standards and Regulations

As with traditional aviation, UAM will be subject to standards and regulation. In traditional aviation, airspace usage is shared by military and civilian vehicles and operators. Military operations are generally governed by their own internal rules and policies while civil aviation is governed by National Aviation Authorities (NAA). For international civil aviation to function smoothly, the Chicago Convention was created which gave rise to the International Civil Aviation Organization (ICAO). As part of the United Nations (UN) its main function is to assist signatory member states to cooperate and use airspace to achieve growth and their shared civil aviation needs [3]. ICAO maintains and publishes Standards and Recommended Practices (SARP) that each of the NAAs implement as regulations within their own jurisdictions.

In Canada the NAA is Transport Canada and maintains the Canadian Aviation Regulations (CAR) [81].Similarly, in the United States, the NAA is the Federal Aviation Administration(FAA) and maintains the Federal Aviation Regulations(FAR). Each signatory of the Chicago convention has a NAA, although some states used shared authorities (e.g. European Union Aviation Safety Agency [82]), while other states leverage other NAAs and incorporate references to their regulation into their own equivalent regulations, for example Brazil includes references to specific FAA regulations [83].

UAM/AAM regulations are still being discussed because the capabilities and expected usage of EVTOL vehicles and UAVs can be substantially different than traditional aviation despite some obvious overlap. The usage of low-level airspace for transport in an urban environment is neither novel nor new, in fact urban transportation of passengers by helicopters has existed since the 1950s [84]. Some important distinctions exist between the operation of helicopters compared to the expected operations of EVTOL vehicles that necessitate the need for modified regulations. For instance, the FAA advisory circular 150/5390-2D: Heliport Design [85] was written to assist heliport operators to design and build heliports that meet all FAA regulatory requirements, while the FAA has recently issued engineering brief 105A: Vertiport Design [20] which is supplementary guidance to 150/5390-2D that helps vertiport operators to design and build vertiports for use by EVTOL vehicles.

Due to the large scope, academic research related to aviation regulations varies. Some relevant examples include: challenges related to certifying EVTOL vehicles in multiple NAAs that have regulatory differences caused by regulations that were updated differently to support EVTOL vehicles [86], identifying regulatory changes required for greener operations [87], determining optimal EVTOL vertiport locations considering local regulations [88], and a survey-based approach to UAM policy guidance [89]. What these have in common is that they provide useful research and studies that can guide decisions of policy makers and regulators.

Chapter 4

Digital Twin Modeling and Simulation

Urban Air Mobility for the transportation of people is not an operational system today. While UAVs are used for many government, commercial, and private activities their operations are quite limited in urban environments. There is no UAM system operational data available, and no means to modify the real world, so the research intent is to create a Digital Twin Prototype that can use available data sources and provide an optimized approach heading that minimizes third-party impacts. The metrics developed are inherently person-centric as opposed to population-level or regional, so an agent-based approach will be employed. Simulation-based optimization is a concept that is used for complicated systems that are difficult to describe mathematically, so using simulation techniques to model the system of human behaviors is a good strategy because these behaviors are stochastic at the individual level.

Creation of the digital twin involved 3 steps: choice of an appropriate simulation and modelling engine, identification of required and available input data, and then creation of the model including (its architecture, input data processing and digital world assembly, agent creation, metric generation, and analysis. Additionally, a case study will be performed so a specialized architecture that will support pre-determined scenarios will be developed as well.

4.1. Choice of Simulation and Modelling Engine

A Digital Twin approach requires a level of modelling fidelity that can accurately twin the physical system. Urban Air Mobility is a complex 4-dimensional system where the physical system being represented includes both the operation of UAM vehicles, and the rest of the Urban environment. The intent of this research is to provide metrics for third-party impacts of UAM operations so the chosen software will need to support the modelling of EVTOLs as well as urban infrastructure and non-users of UAM in a way that metrics can be used to evaluate different scenarios.

In support of the research purpose the chosen simulation and modelling engine shall support the following system requirements:

SyRS-1: The software shall support 3-dimensional placement and movement of virtual content.

SyRS-2: The software shall support the inclusion of virtual content (2-dimensional, 3-dimensional, database, API) in non-proprietary formats.

SyRS-3: The software shall support the simulation of time, both real-time and faster than real-time.

SyRS-4: The software shall include a robust set of included libraries (including support for pedestrians and vehicles, and road networks)

SyRS-5: The software shall support agent-based modelling and discrete event simulation.

SyRS-6: The software shall be extensible for the creation of custom classes and methods.

SyRS-7: The software shall be available as freeware or as an affordable student usage model.

SyRS-8: The software shall support repeatable experiments.

SyRS-9: The software shall support the output of experimental data to a database and/or files.

SyRS-10: The software shall support manual controls for live modifications of experiments.

While there are many Commercial Off the Shelf (COTS) software packages available with different feature sets, the chosen software package was Anylogic [90]. Anylogic supports all the system requirements and has a freeware version that, while limited, has the functionality required for this research.

4.2. Evaluation and Choice of Available Input Data

To create the digital twin, various data sets are available that serve different purposes. The City of Montreal has a large collection of data freely available [91], including information about the built environment and the movement of people within it. A pass was made through the relevant data sets available from the city of Montreal as well as other sources of interest. A summary of some of the data sets are shown below.

4.2.1. Geospatial Data

Data related to the creation of the virtual representation of the built environment is categorized as geospatial data. These data sources generally do not receive frequent updates since the physical systems that they are describing change very slowly.

4.2.1.1. 3D Buildings

The 3d building dataset [92] is available for much of the Urban area of Montreal in 3dm (Rhino3D), CityGML, GDB (Esri), and DWG (AutoCAD) formats. The most recent data was collected in 2016 generated by photogrammetry. Figure 2a shows the boroughs with available 3D building data including the Ville Marie borough (highlighted). Each borough is further subdivided into sections shown in dwg format in Figure 2b.

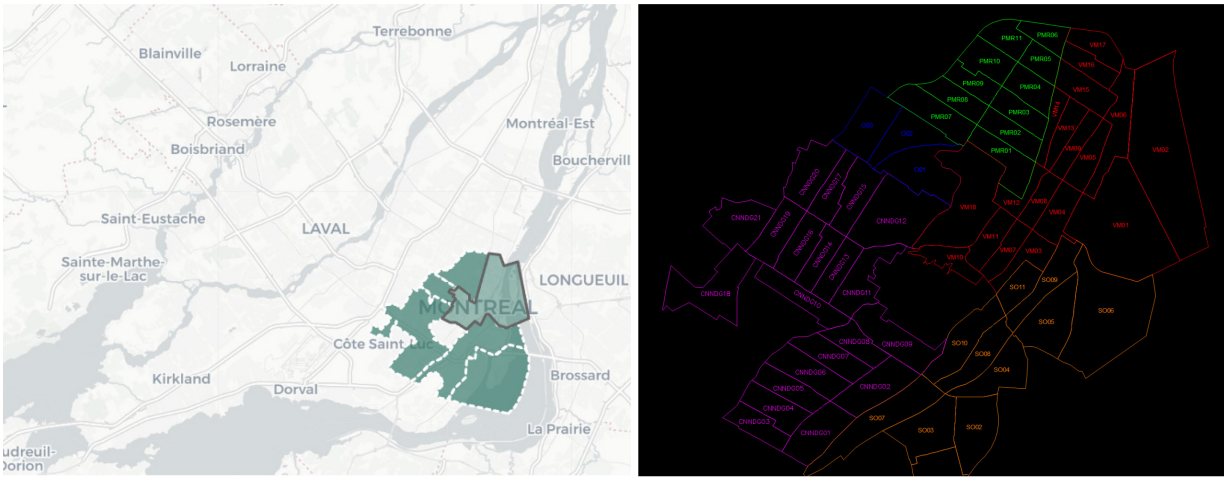

Figure 2 - 3d Building Datasets. (a) boroughs available in the dataset, (b) subdivision of boroughs

Figure 3a shows the 3d content available in section VM11 that includes the Concordia Hall building shown on the right-side edge of the visualization. The buildings shown in section VM11 contain their elevation data, meaning the base of all buildings conforms to the surface of the ground. The geometric accuracy of the building data is claimed to be +-30cm to +-40cm. Despite this, the models are highly detailed with many small sized details present. Figure 3b shows the Concordia Hall building including roof equipment and other small protrusions

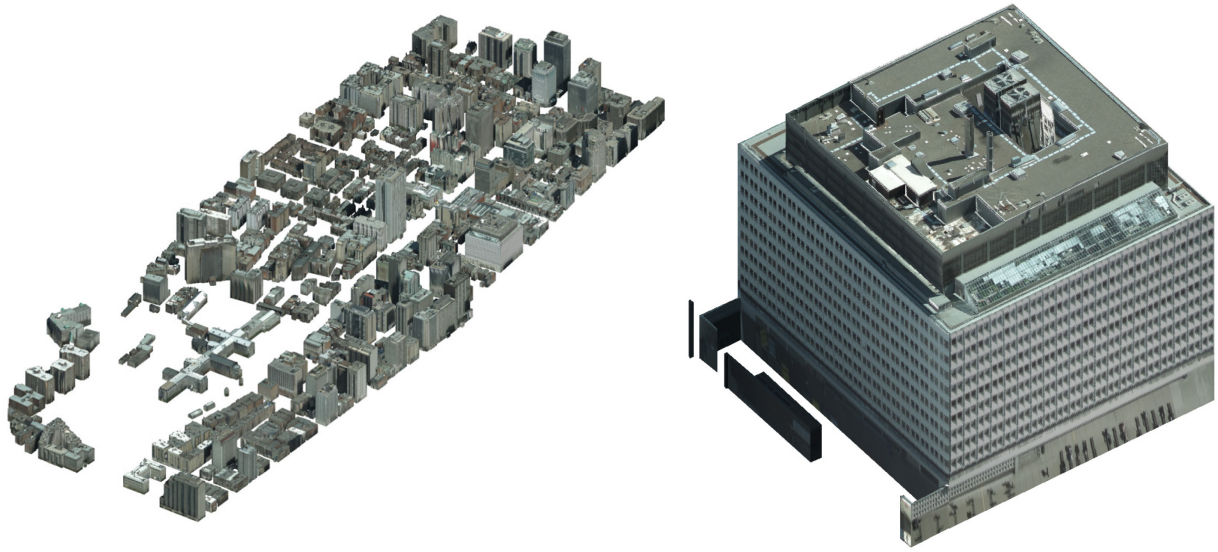

Figure 3 –3d building data (section VM11). (a) visualization of VM11, (b) visualization of Concordia Hall building within VM11

The data is in metres using a NAD83(CSRS) coordinate system, specifically NAD83 CSRS (98) MTM-Zone 8 which is a Transverse Mercator projection coordinate system [93]. An example of the x, y, z coordinates for the bounding box of the Concordia Hall building are (298575.90, 5039659.00, 42.71) to (298680.88, 5039764.00, 117.00).

4.2.1.2. Digital Elevation Model

Elevation data for all regions of Montreal are available in CityGML and LandXML formats [94]. Figure 4a shows the boroughs on Montreal Island with the Ville Marie borough highlighted. According to the information source, the data was generated by processing LIDAR data taken in 2015 and removing the buildings using a program called TerraModeler. It is accurate to +-20cm in both altitude and planimetry. Figure 4b is the visualization of the Ville Marie borough using FZKViewer [95].

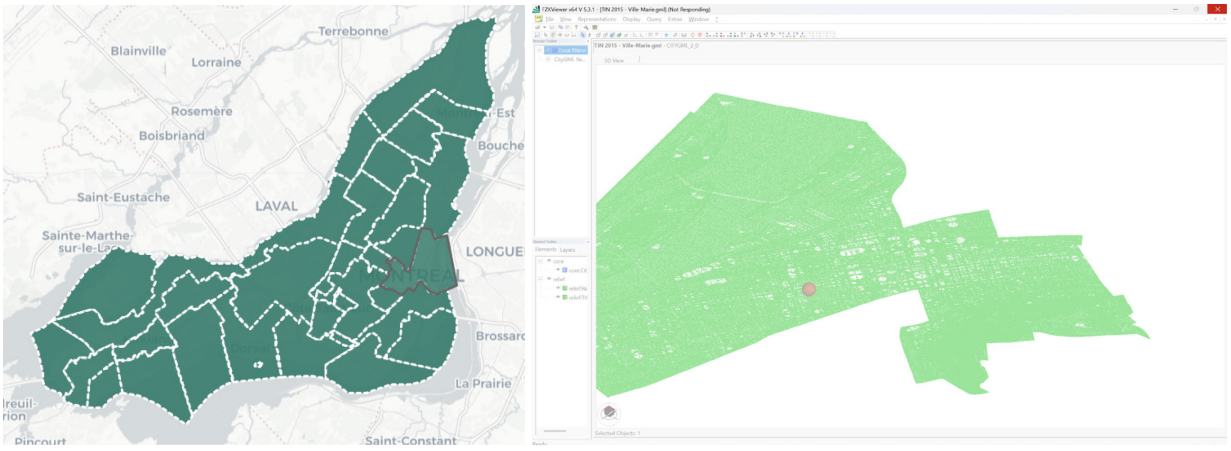

Figure 4 - Digital Elevation dataset (a) boroughs available in the dataset, (b) Ville Marie borough visualized

Similar to the 3D building data, the digital elevation data uses a NAD83 (CSRC) coordinate system. It consists of connected triangles in 3d space. The Ville Marie borough data shown in Figure 4b contains over 2 million triangles.

4.2.1.3. City Base Maps

The basic city maps contain a lot of useful data, including elevation lines, building footprints, building roof elevations, sidewalk and road edges, and foliage cover data. The data is available in DWG (AutoCAD), DGN (Microstation), and PDF formats [96]. Like the 3D buildings and elevation data, the base maps use a NAD83 (CSRC) coordinate system. The data set is split into tiles where Figure 5a shows the data tiles colourized by update date which range from 1985 to 2024 depending on the region. Figure 5b shows an example of the data visualized in PDF format with the image centred around the Concordia Hall building. The source data for the Ville Marie borough was last updated in 2024.

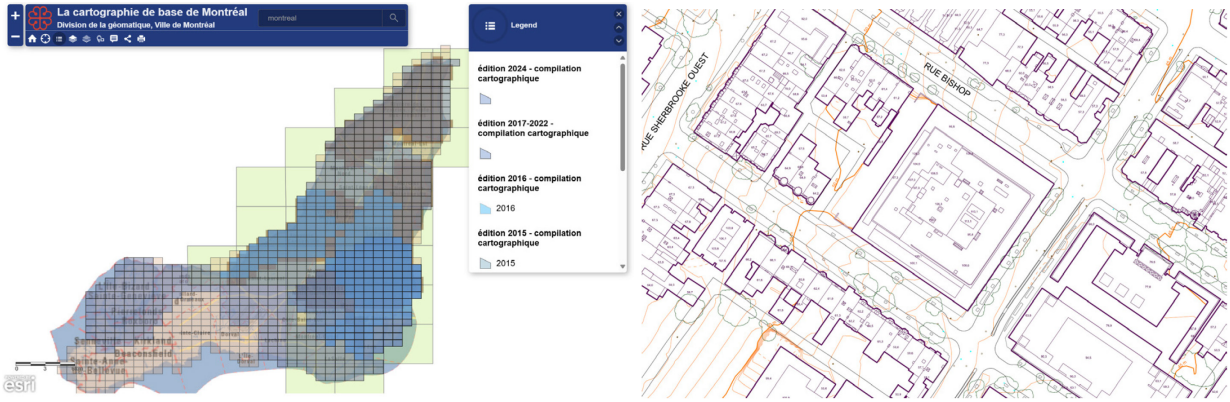

Figure 5 - Base maps. (a) Available data tiles colourized by date, (b) basemap showing the Concordia Hall building

4.2.1.4. GeoBase – Road Network

The road network is described in a dataset covering the entirety of the island of Montreal [97]. It is available in JSON format, csv, and SHP(Esri) formats. Figure 6a is a visualization of the shapefile showing all roads in Montreal. The roads are split into segments with specific criteria determining whether there is a new segment required, e.g for intersections, or when the lane configurations change. Each road segment contains metadata including the name of the road, the coordinates of each point making the segment, and whether the segment is one-way or two-way (but it does not specify the number of lanes). The road segments are identified using a lat-long coordinate system (e.g. -73.56519863944575, 45.4891540059485). Figure 6b shows the section of the Ville Marie borough where the Concordia Sir Georges Williams campus is located.

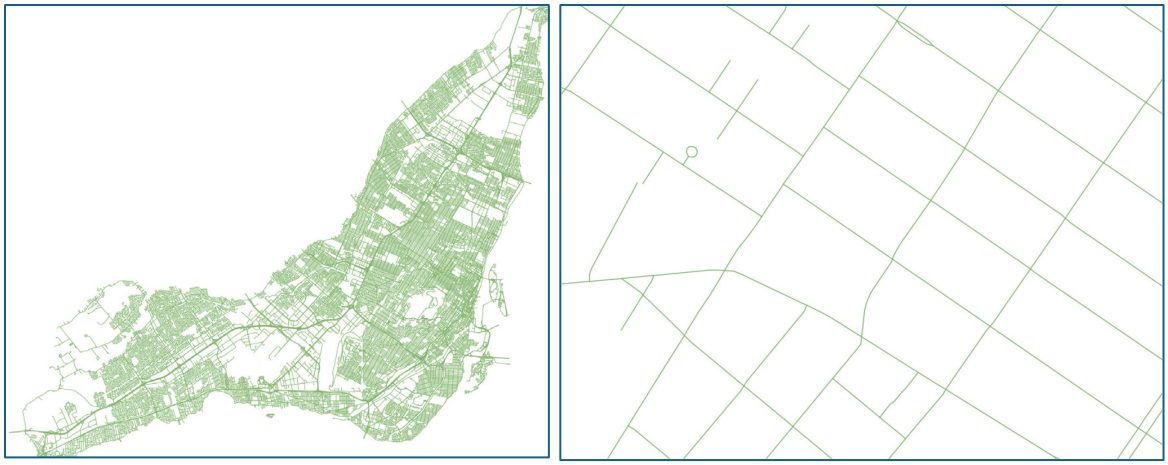

Figure 6 - Road network data. (a) Roads in Montreal (b) Portion of Ville Marie centred around the Concordia Hall building

4.2.1.5. Aerial LiDAR

Aerial LiDAR data for all regions of Montreal are available in LAZ format which is a compressed version of LAS (ASPRS) format [98]. Figure 7 shows the available tiles from the 2015 LiDAR survey.

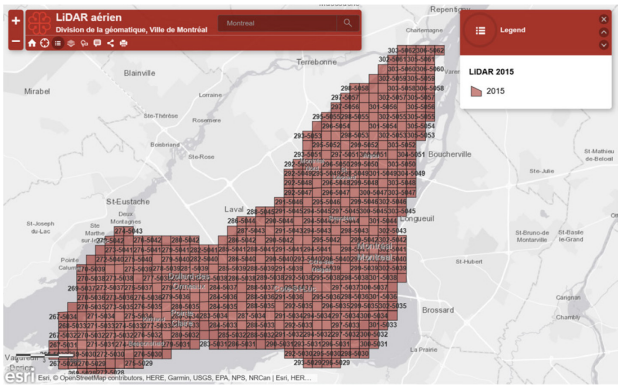

Figure 7 - Aerial Lidar data tiles

The LAZ file data forms a point cloud. The data is accurate to +-20cm and uses the NAD83 (CSRS) coordinate system. The data points have been processed by the data provider into 9 classifications including classifications for vegetation, buildings and terrain.

Figure 8 shows the visualizations in 2d and 3d of a section of the Ville Marie borough where the Concordia Sir George Williams campus is located. The classification scheme is easily visible where for example vegetation is shown in green, while buildings are shown in red.

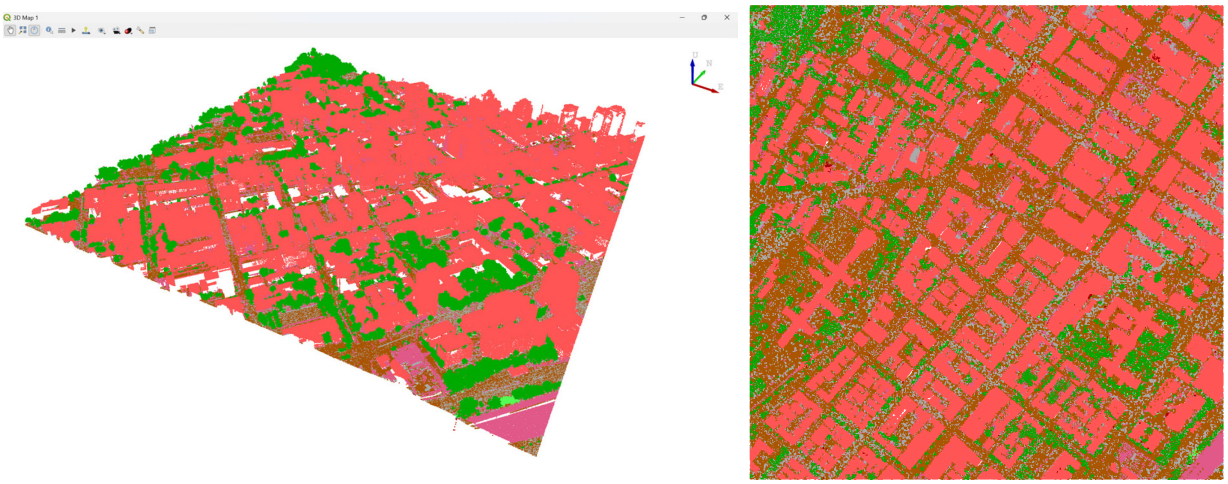

Figure 8 - Aerial Lidar Tile example (a) 3d visualization, (b) 2d visualization

4.2.1.6. Urban Tree database

A catalogue of trees is maintained and currently contains over 300000 trees [99]. The catalogue has trees located in the boroughs highlighted in Figure 9a. The data contains information such as the locations in NAD83 coordinates system (and their Latitude / Longitude), the tree species, the trunk diameter at a standard 1.4m above ground level, the most recent measurement date, the planting date. Figure 9b is a visualization from a webapp named Quebio [100] of the urban tree locations (colourized by species) in the Ville Marie borough near the Concordia Sir George Williams campus. Tree height is an important measurement that is missing from the database. Research has been done to provide a statistical model of trunk diameter vs height for many tree species [101] so the height could potentially be modelled similarly.

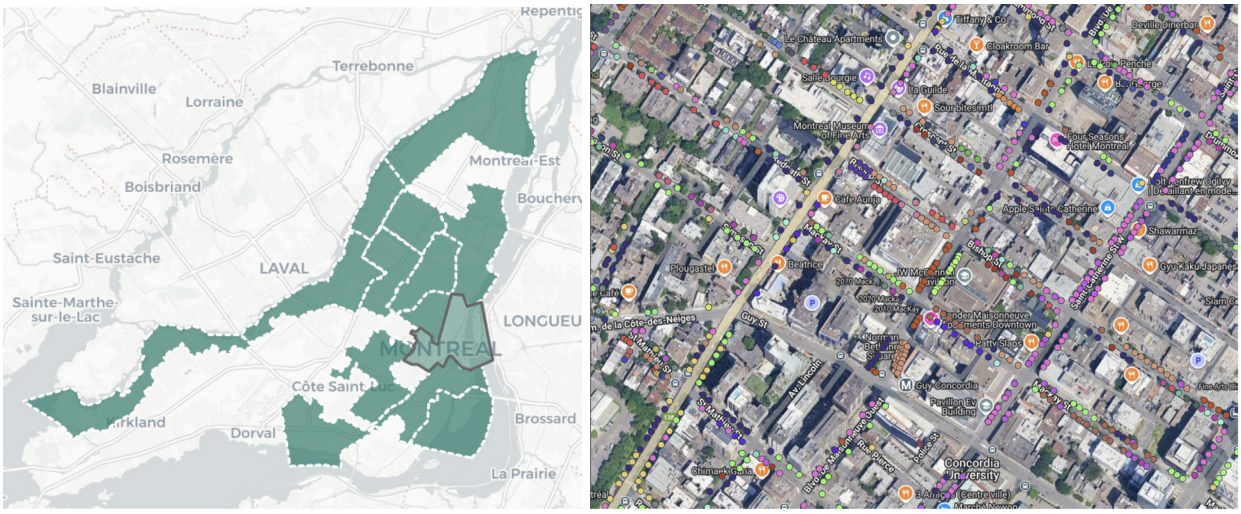

Figure 9 - Urban Tree database. (a) boroughs available in the dataset, (b) Visualization near the Concordia Hall building

4.2.1.7. OpenStreetMap

OpenStreetMap [102] is a crowdsourced mapping system where users contribute to adding details and updating content as needed. It is open data available through the Open Data Commons Open Database

License (ODbL) [103]. OSM has a lot of content for various purposes, and different layers can be displayed thematically, e.g. the standard usage view, cycling maps, transport maps, etc. There are multiple APIs available for downloading content from OSM. The export feature available from the main webpage exports all content shown in the current map layer in OSM XML format [104]. The OSM XML format contains lots of information, including position data for intersections, sidewalks, roads and their lane configurations.

The web interface also has an export to PNG/JPEG/WEBP/SVG/PDF function available for coordinates specified by the user. Figure 10 is a screenshot showing the Standard layer view for a portion of the Ville Marie borough near the Concordia Sir George Williams campus. It uses a latitude and longitude coordinate system to 6 decimal places and therefore has an accuracy of at best 11cm, although since content is user generated, the actual accuracy is unknown.

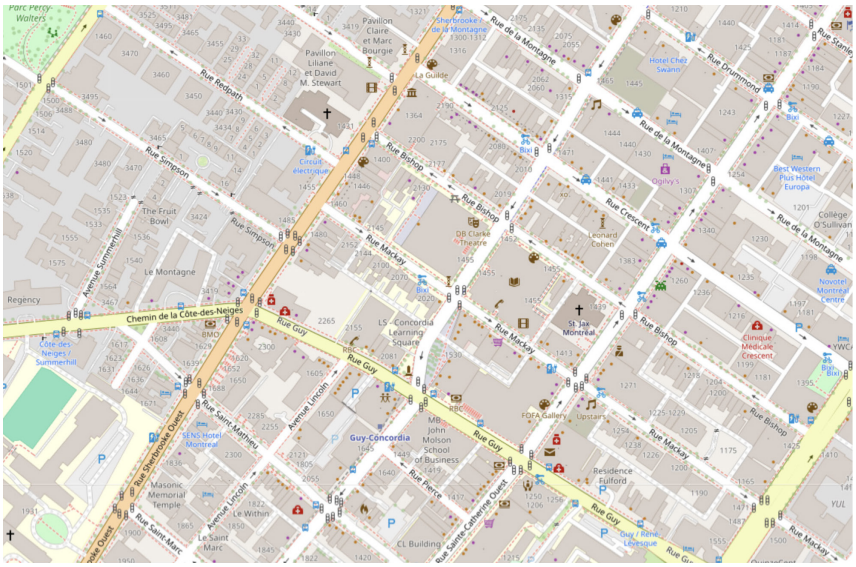

Figure 10 - Open Street Map visualization centred around the Concordia Hall Building

OpenStreetMap is also available natively in Anylogic when using the built-in GIS functionality. It is easy to create a map whose content is automatically downloaded using the Anylogic API. As of Anylogic version 8.9.3, maps created using the GIS functionality only appear in 2D when running the simulation. The 3D viewer does not support the GIS functionality.

4.2.2. Mobility and Traffic Data

Data related to the position of people comes from mobility and traffic data. The real-time position of people is not public data because it is generally owned by private organizations that collect the data themselves, but also there are privacy, and other legal challenges. Because the position of people changes continuously, the available and required update frequency is in the order of minutes.

4.2.2.1. Telus Insights

Telus is a popular mobility carrier in Canada that operates a vast network of cellphone towers. The location of every phone connected to these towers can be triangulated so Telus has access to live location data for millions of people and records billions of device locations daily [17]. Researchers can gain access to their Location API that has many features meant for understanding mobility. According to their publicly available API documentation, the REST-based API is capable of receiving a call that contains a study zone area in the form of a shapefile and returning many types of mobility data. The documentation

describes the data as being collected every 5 minutes, but the output metrics are limited to a minimum update interval of 15min, and the data is delayed intentionally for 1-2hours [17]. Table 1 is a summary of the main API response types.

Table 1 – Telus Insights Location API data descriptions [17]

Data	Description of returned content during the request timeframe
Destination	The number of devices where study zone was the destination
Demographic	The estimated demographic breakdown of devices in the study zone
Dwell Time	The number of devices spending a given amount of time in the study zone
Home	The number of devices in the study zone with a given home location
OD Matrix	The number of devices that travelled between given origin and destination
Origin	The number of devices where study zone was the origin
Repeat Visitation	The number of devices that visited the study zone more than once
Total Trip	The number of trips that devices made to a study area
Trade Area	Where customers live and how far they will travel to reach the study area
Unique	The number of unique devices in the study area
Work	The number of devices in the study area with specified work location

The "Unique" data listed in Table 1 appears to contain the type of data that could be used to generate the virtual content of a digital twin, but the update characteristics, specifically the intentional 1-2h delay make the data inaccurate for live decision making.

4.2.2.2. Here.com Traffic

Here Technologies [105] is a mapping company with a free-tier API access level. They support many APIs including map imagery APIs as well as a few APIs for live traffic data [106]. For purely visualization purposes, Here has APIs for raster or vector tiles and for other purposes, it has APIs for street-level traffic slowdown data. According to the API details, it provides traffic data updates every 1 minute and incident report updates every 2 minutes [106].

The data source for traffic data is primary through its network of Here probes (users of connected applications). While it is likely not as complete as dominant phone provider data (like google) or phone network providers (like Telus), with exploratory testing it was determined that the Here probe data has good coverage for the Montreal area. The probe data is processed using anonymization techniques to remove personally identifiable information, which means that the data available is not at the individual user level, despite this data existing and being owned by the data provider.

The Traffic API is a REST API where the user specifies the attributes of the request via a URL and the service returns a JSON file. The traffic API accepts as inputs coordinates and shapes in WGS84 format, along with other filters as required), and it returns the traffic congestion data for all roads that fit the filter and where enough data exists for the service to calculate a confidence measure. An example REST call for the Concordia Sir George Williams Campus is as follows:

https://data.traffic.hereapi.com/v7/flow?in=bbox:-73.584277,45.493454,-73.573477,45.501054&locationReferencing=shape&apiKey=<API_KEY>

Figure 11 - Sample Here.com Traffic API REST call

Which will yield a sample response with a series of points describing road segments with their associated parameters; speed, uncapped speed, free flow, jam factor, confidence, and traversability:

```
    "sourceUpdated": "2025-11-04T15:24:12Z",
    "results": [
        {
            "location": {
                "description": "AUT-720/Autoroute Ville-Marie",
                "length": 491.0,
                "shape": {
                    "links": [
                        {
                            "points": [
                                {
                                    "lat": 45.49385,
                                    "lng": -73.57509
                                },
                                {
                                    "lat": 45.493,
                                    "lng": -73.5759
                                }
                            ],
                            "length": 114.0,
                            "functionalClass": 4
                        },
                        {
                        {
                        {
                        {
                        {
                        {
                        {
                        {
                        {
                        {
                        {
                        {
                    ]
                }
            },
            "currentFlow": {
                "speed": 7.2222223,
                "speedUncapped": 7.2222223,
                "freeFlow": 10.555556,
                "jamFactor": 2.5,
                "confidence": 0.7,
                "traversability": "open"
            }
        },
```


Figure 12 - Sample JSON response from Here.com traffic API REST Call

Using the sample call shown in Figure 11 and sample response shown in Figure 12, the data can be easily visualized on the GIS map in Anylogic using some supporting code. Figure 13 is an example comparing the live here.com web app view with the output from the traffic API of the Concordia Sir George Williams Campus in the Ville Marie borough. Some clear discrepancies can be seen. The Traffic API has a reported delay of 1 minute, while there is no published temporal accuracy for their own online map view, which could explain the differences.

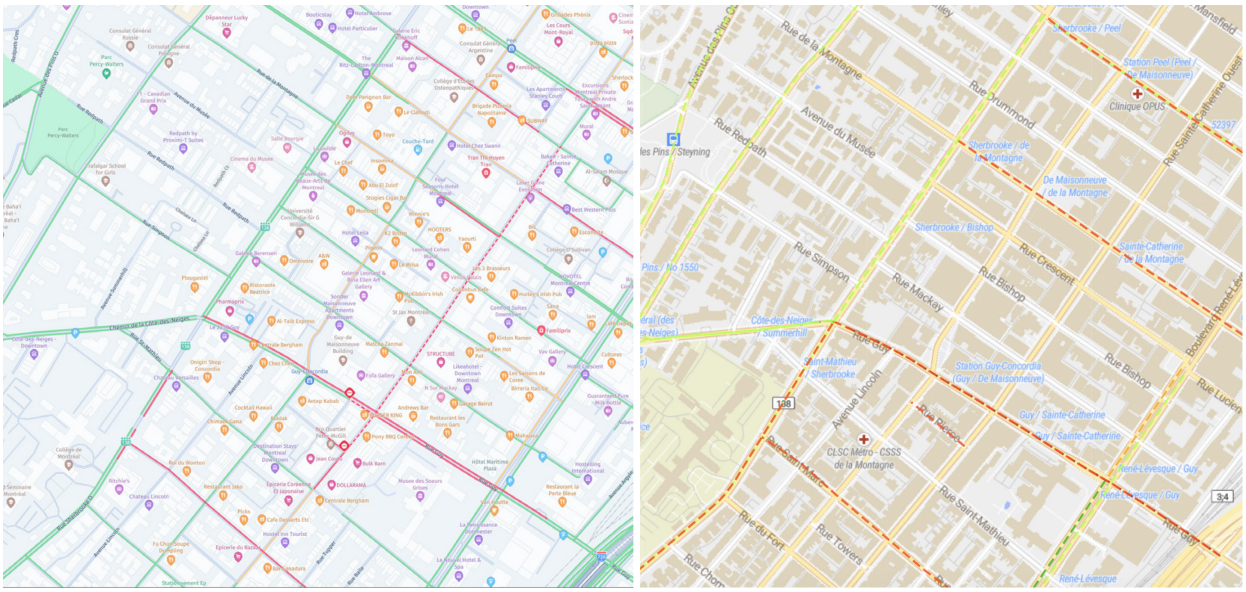

Figure 13 - Visualization of Here traffic data a) on here.com website, b) in Anylogic

After using the API, it is clear that it supports independent bidirectional traffic flow data, but it does not communicate which direction the flow data represents. As can be seen in Figure 46 of section 4.5.4.3, there exists a small lat/long offset in the road segments for each direction of travel. This offset could possibly be used when there is data for each direction of a road segment present.

4.2.2.3. Google Maps

Google Maps [107] is a well-known proprietary map system that includes multiple 2D map views including standard map, topographic, and cycling maps, traffic, satellite, and also a 3D map viewer generated by photogrammetry. The traffic view uses crowd-sourced data from mobile devices to reconstruct traffic patterns in real-time. Figure 14a is a screenshot showing the traffic data. Not every street has available data, but the roads that do, display the health of the traffic flow using multiple colourizations that are based on traffic speed and volume [107].

Aerial/Satelite/streetview photos are the easiest source to understand the road network. Figure 14b shows the satellite view of the intersection of Guy and Sherbooke in the Ville Marie borough at Concordia's Sir George Williams Campus. The individual lanes and intersection can be seen in the photo.

Figure 14 - Google Maps satellite view data (a) Traffic visualization, (b) visible lane markings

4.2.3. Data Selection and Architecture

4.2.3.1. Chosen Data

After determining the metrics that would be measured (see Chapter 5), it was decided that to accomplish the research objectives, the minimum sources of data that would be needed to create the visual world were building data, background map imagery, aerial imagery and street view as a reference for the road network lanes. For the third-party modeling, the population density data and traffic data were needed to create the agents despite live population data not being available.

Background imagery is available from multiple sources detailed above (and others). For the purposes of this research, using an API to update the map content was not necessary to achieve the research objectives. Additionally, using a static image included the benefit that some content such as the roads, lanes, intersections, and pedestrian walls could be added manually without the difficulty of adding them programmatically. OpenStreetMap was chosen for the background map imagery due to its easy browser-based export to PNG function. Google maps satellite view and street view were used as the reference for

generating the road lanes, and intersection. There was no available live population density data provider chosen due to the update characteristics of the data, but explorations into using the Telus Insights API are performed and discussed in section 4.5.3.2. Here.com traffic API is used for traffic.

Terrain was not chosen for inclusion because while it would add realism, it was deemed to be not necessary for the third-party metrics and would add unnecessary complications to the setup of the case study because roads and surfaces that ground-based agents would interact with would all need to be modelled accurately with elevation as well. No limitations were found in Anylogic that would prevent future work from including the terrain elevation data as required.

The Montreal City base maps were deemed not necessary. While they contained lots of data describing the urban environment, the nature of the information contained in DWG format was not complementary to the other data sources and therefore was not useful especially since data such as elevation, trees, and building footprints were redundant with other data sources.

The geobase road network data from the city of Montreal was not chosen because manual creation of the case study area road network was a simple enough task. Similarly, the information contained in OpenStreetMap regarding road positions and lane information was not used. Future research could extract the road network data from either (or a combination) of these sources and dynamically create the road network

The aerial LiDAR data was interesting but not useful for this research. Much of the information available in that dataset is redundant with other sources because other those other data sets use the LiDAR data as the raw data. One piece of information that was not completely redundant with other sources is the vegetation cover layer which could be useful for privacy metrics. The tree database was included for display purposes only, but due to the low accuracy of that dataset when it comes to vegetative cover, future research could incorporate the LiDAR data for the vegetation modelling as required.

Table 2 shows the chosen data sources, their format, and their expected update frequencies.

Table 2 - Chosen Data Sources

Data	Source	Format	Update Method	Physical Twin Expected Updates	Available Update Rate
Background Map Imagery	OpenStreetMap	PNG	manual	years	On-change (crowd sourced)
3D buildings	City of Montreal	CityGML	manual	years	Years
Population density	Telus Insights	JSON	N/A (exploratory)	Real-time	15minute with 1-2h delay
Traffic Data	Here.com	JSON	API	Real-time	minutes
Road Network Lanes	Google Maps	browser	manual	years	years

4.2.3.2. Digital Twin Architectures

To accomplish the research objectives, multiple architectures of the digital twin prototype were created to support different aspects of the research. Both architectures are part of the same Anylogic file. The choice of architecture is performed at load time by selecting the appropriate Simulation Experiment. As seen in Figure 15 the model has 3 Simulation experiments; ScenarioEvent and ScenarioNormal use the first

architecture, while DigitalTwinPrototype uses the second. In both architectures the assembly of the virtual Twin Geospatial Area content (3d buildings, reference map, road network, and Vertiport) are created the same way, as described in Section 4.4.

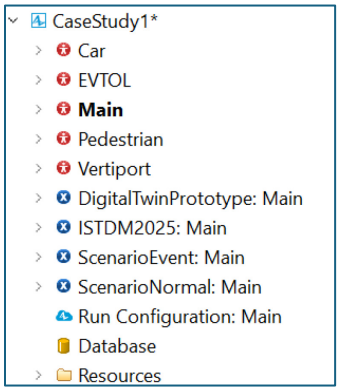

Figure 15 - Anylogic Model showing agents and Simulation Experiments

The first architecture is used for the case study and supports any number of pre-designed settings that are used to validate the feasibility of this approach in a repeatable manner. The Simulation Experiments, ScenarioEvent and ScenarioNormal, use this architecture. The architecture is shown in Figure 16 and the settings are described in section 6.2. This architecture is decoupled from live real-world data but supports representative movements of pedestrians and vehicles to validate whether a dynamic agent-based scenario can produce repeatable results and whether the objective to minimize third-party impacts is feasible based on the sensitivity of the metrics to changes in agent positions. This architecture can also be used for analyzing proposed regulations. It is built to analyze multiple flight path scenarios that present alternatives to the described regulations. In this implementation, 128 simultaneous EVTOL agents are created, each with their own independent third-party impact calculations. These simultaneous agents allow the generated metrics for each flight path to be compared directly since they are all flying in exactly the same third-party setting. This architecture can show how modifying the approach characteristics effectively changes the third-party impacts and hence can be used to evaluate the impact that proposed regulations have on third parties. As is described in Section 6.2, the case study compares third-party impacts of approach headings and descent profiles proposed by the FAA with various alternative approach headings and descent profiles.

The vehicle and pedestrian agents are created using a stochastic agent creation algorithm that determines statistically where the agents should be created to achieve the required population densities and traffic patterns. This method is described in Section 4.5. Creating realistic representations of a large group of dynamic agents is a complex problem so the settings used for the case-study were fine-tuned manually to create the expected traffic patterns and pedestrian densities. This method has an inherent “warm up time” where the vehicle and pedestrian agents come to a mostly steady state that matches the required population densities and traffic patterns. Once the steady-state is reached, the EVTOLs approaches are performed and the third-party metrics are calculated. The use of a stochastic agent creation algorithm means that every time the settings are run, the results are slightly different. Everything is output to a csv file where offline analysis and metric visualizations can be created.

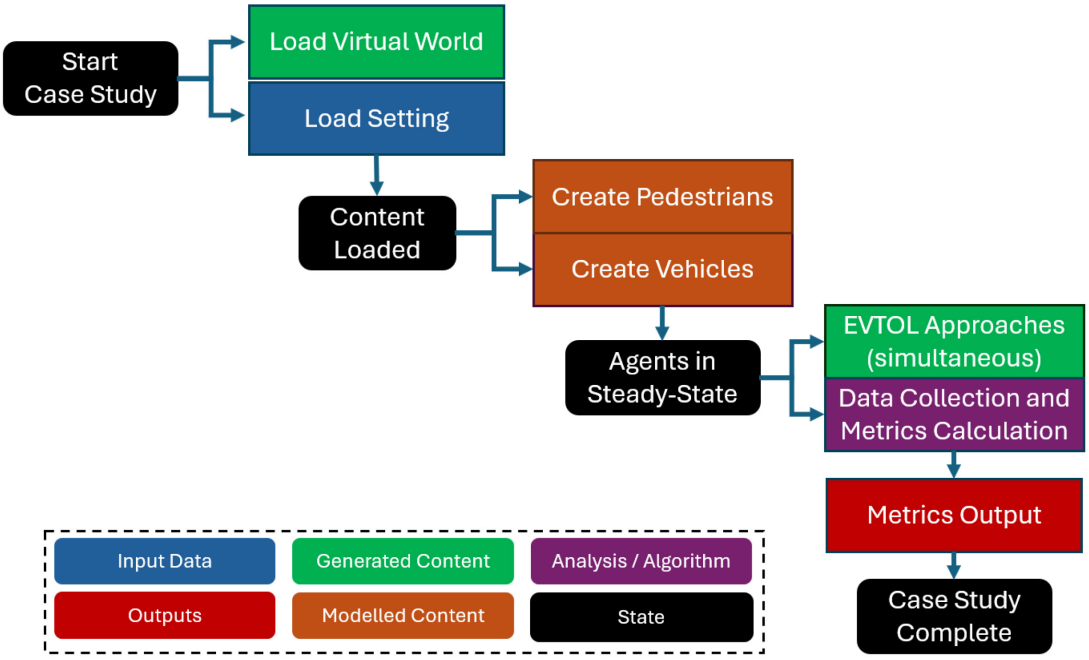

Figure 16 - Case Study Architecture

The second architecture (shown in Figure 17) is a functional digital twin prototype. The simulation experiment, DigitalTwinPrototype, uses this architecture. It uses real-world data and demonstrates that EVTOLs can be routed based on minimized third-party impacts using a Digital Twin and available data. EVTOL agents are created anytime by the user to simulate on-demand landing requests. A simulated air-traffic management system feeds optimum flight approach heading to the created EVTOL agent. As is described in Table 2, the vehicle traffic and population data are not exactly real-time. This architecture creates static scenarios where the agents are created based on the data, but do not move. The method is described in Section 4.5

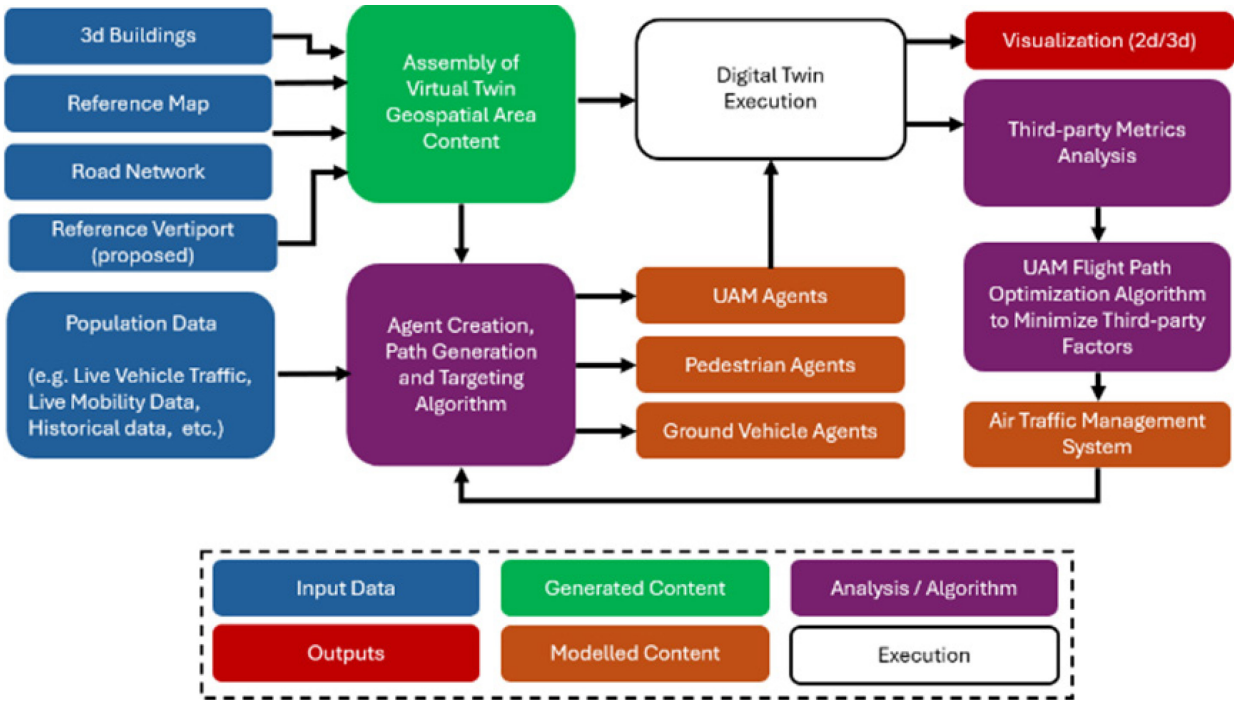

Figure 17 - Digital Twin Prototype Architecture

4.3. Digital Twin Graphical User Interface

The Digital Twin graphical user interface provides a mix of visualizations and controls. As shown in Figure 18, the interface includes 1) a radar chart showing the current integrated third-party metric (described in section 5.4) along with a blue line indicating the approach heading with the minimal metric, 2) Digital Twin Controls, 3) enablable visualizations such as impact/debris areas, traffic congestion, pedestrian density, and size control for agent visualizations, 4) manual controls of the model, 5) the main view area. All elements will be described in later sections.

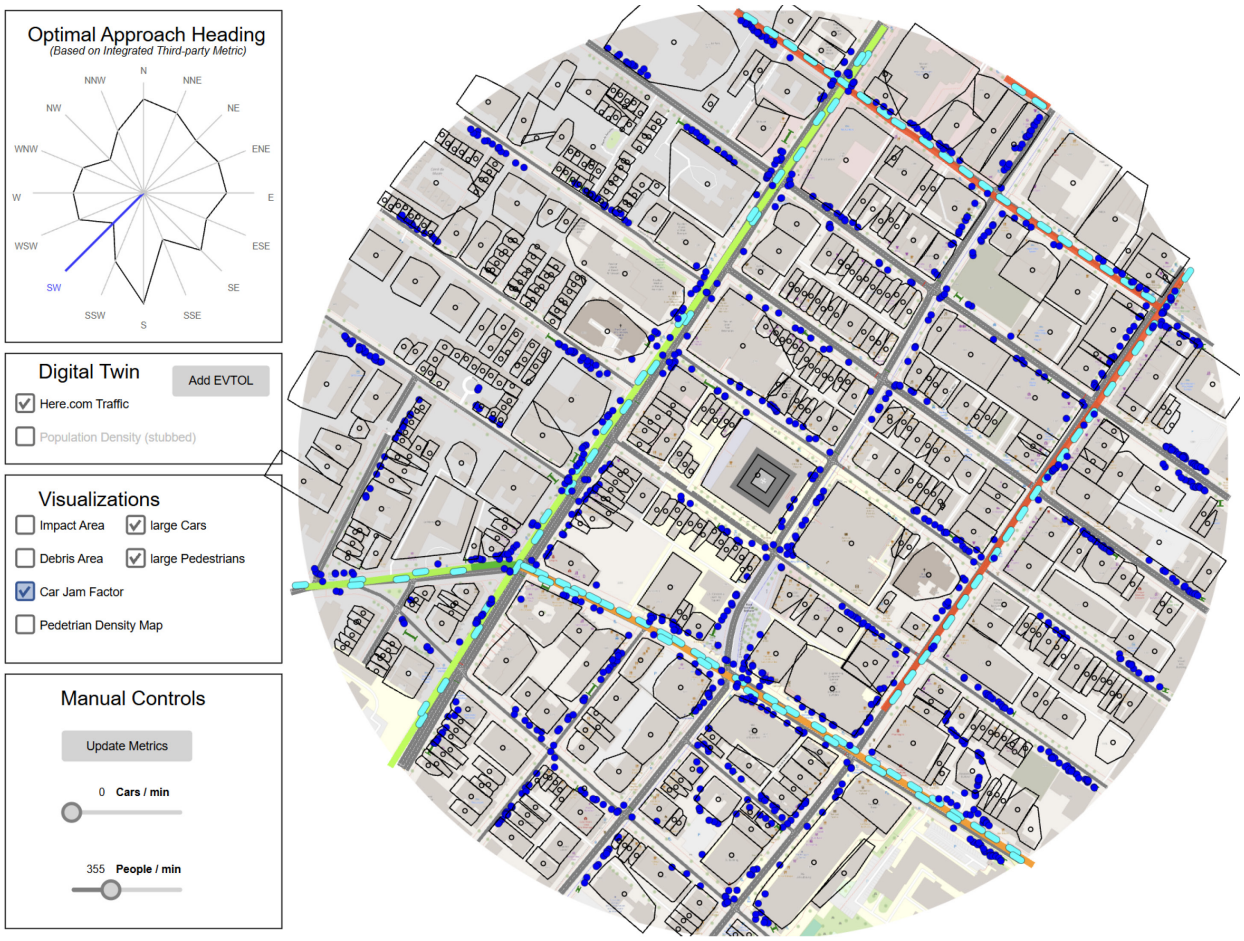

Figure 18 - Digital Twin Prototype Graphical User Interface

4.4. Virtual World Creation

The Anylogic simulation environment has many built-in features that make it an ideal solution for creating a virtual world. Many features of the software were used and described below.

The main agent for the anylogic model is shown in Figure 19 and contains most of the model logic as well as the presentation layer that is used for the visualization. As can be seen in the figure, the case study area is located at the top left quadrant while the default frame and logic are located in the bottom right quadrant. The reasoning for this will be explained in the next section.

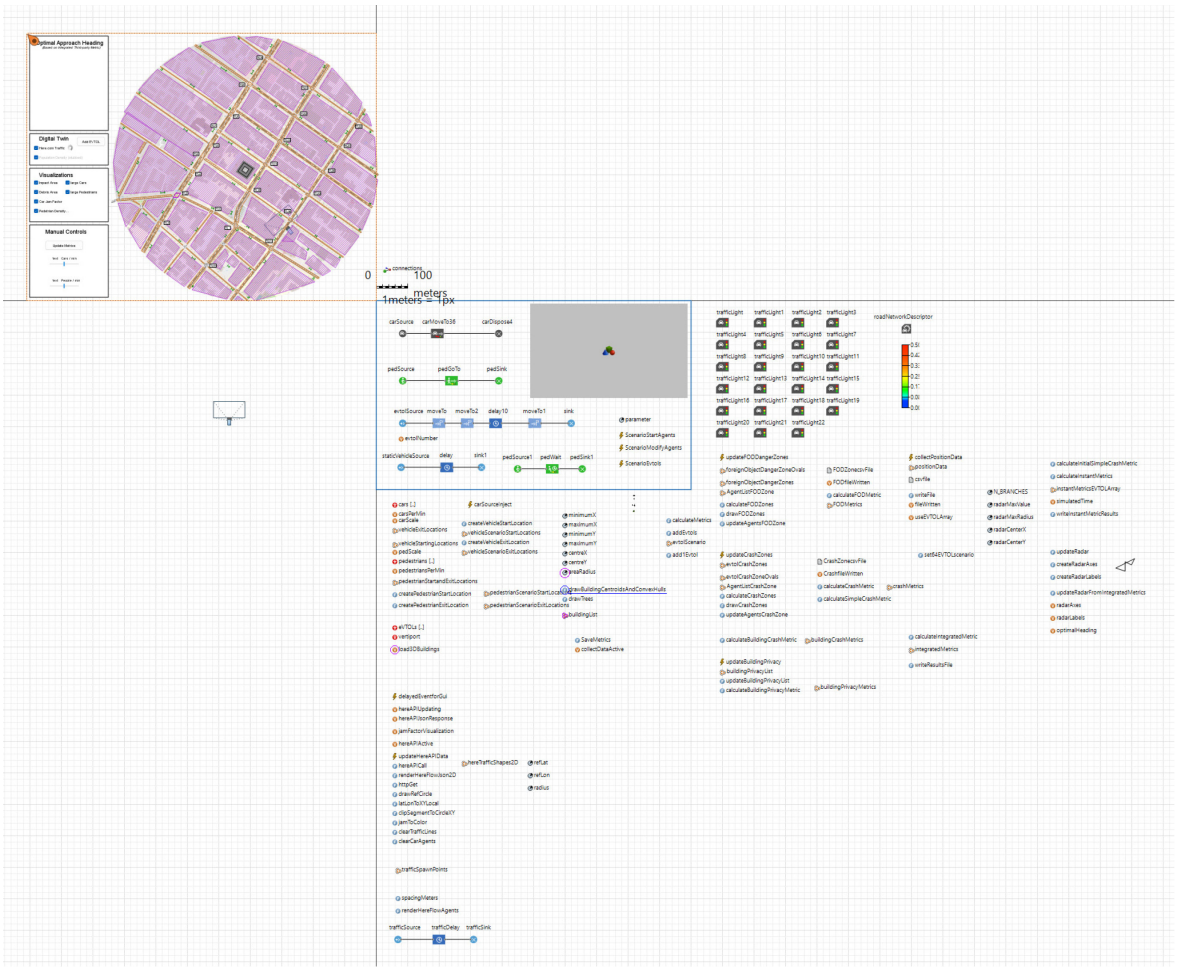

Figure 19 - Main Agent diagram

4.4.1. Presentation Layer

The main presentation layer comes with a scaling functionality that allows the display of content to be scaled appropriately. The scale chosen for this project was 1pixel equates to 1m in the real world as shown in Figure 20. This simple scaling was chosen for multiple reasons; 1) it is easy to understand and work with, 2) the standard 3d content library included with the program is in meters, 3) Many data sources including OpenStreetMap and 3d building geometry data is in meters. Figure 20 additionally shows the origin of the presentation layer. The origin is the point where x, y, and z are equal to zero in the presentation layer units. Positive x increases to the right, positive y increases downward, and positive z increases towards the observer as shown

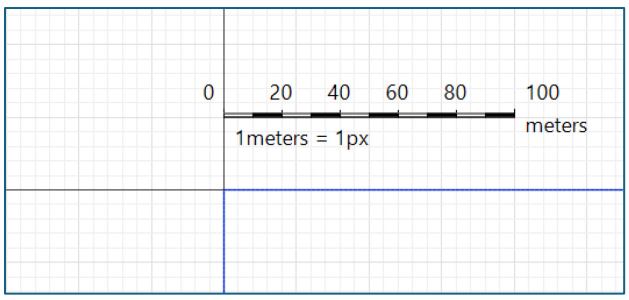

Figure 20 - Presentation layer scale

4.4.2. Background Map

The ground layer was chosen to be a flat surface for simplicity. Future work could use topological data to improve the 3d scene. To be presentable visually, a 2d map was created using imagery downloaded from OpenStreetMap. Anylogic allows the direct usage of png imagery as an image object visible in both 2d and 3d. Figure 21a shows the case study area (as shown in Anylogic) which is a circular area 840m in diameter centered around the Concordia Hall building. The background imagery is 6000px by 6000px. Figure 21b shows the maximum detail level in the imagery. The imagery was added and the image scaling factor was set accordingly with the main presentation scale so that 1m in the imagery matches 1m in the Physical Twin.

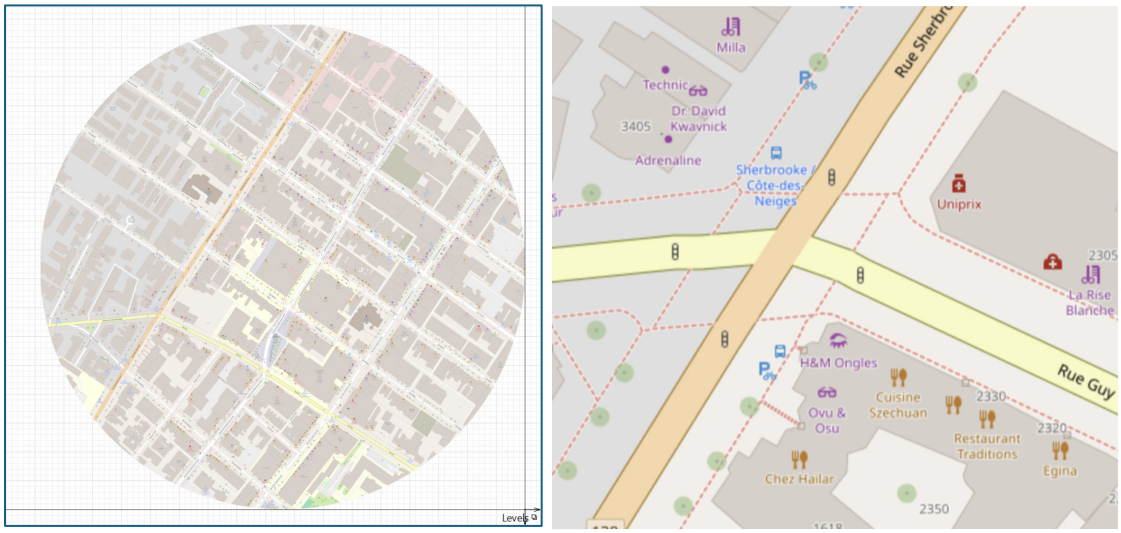

Figure 21 - Background map. (a) case study area, (b) maximum detail

4.4.3. Road Network

The road network uses the built-in road traffic library features such as roads (including customizable lanes), and intersections (stop lines and traffic lights). Figure 22a is a portion of the study area showing the road network scaled and located on top of the background 2d map view. Lanes are marked and all possible trajectories for a vehicle to pass through the intersection are shown. All roads in the case study area were accurately modelled based on data from OpenStreetMap, while lanes were created using Google Satellite view as a reference. Figure 22 shows the same intersection as was recreated in Figure 21b. Stop lights were added to match the intersections that have stop lights, but the timing was manually tuned for reasonably smooth traffic control during the case study. Stop light signaling logic is shown in Figure 22b. All stop light timings were tuned manually to keep the simulation simple but provide reasonable traffic flows.

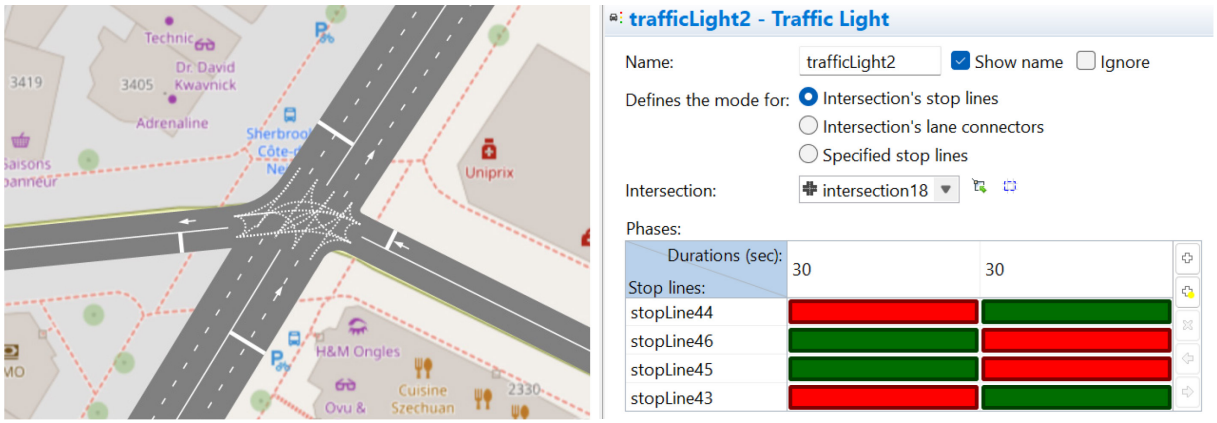

Figure 22 - Road Network creation. (a) Intersection example with lanes and stops, (b) intersection lights logic

In preparation for supporting the pedestrian agent movements to limit them to places where pedestrians are allowed to be, walls were added around the road segments to prevent pedestrian agents from cross roads except at crosswalks. Figure 23 shows the walls drawn and the associated pedestrian crossing areas.

Figure 23 - Pedestrian walls

The traffic library supports vehicle parking, both in lots and along roads. It was chosen to not utilize this functionality because the intent of the research is to measure the impacts on people, not on property so while parked cars would add some realism to the scenario, most parked cars are empty and would therefore not add much appreciable change to the third-party impact metrics.

4.4.4. Buildings

To create an accurate 3d urban environment, cultural features are required. Accurate building geometry is important for creating the safety metrics which rely on relating the position of the EVTOL vehicle and the 3d world. The city of Montreal provides detailed 3d building geometry data in multiple formats which include additional metadata for location and orientation as well as appearance textures. The chosen format for use in this research was CityGML format which is a common open standard format based around XML. Upon manual inspection of the files, it became clear that the CityGML files were converted into CityGML from some unknown format using a program named Rhino. The Montreal dataset contains much of the Island of Montreal and is split into zones where each zone contains all buildings in that zone. A toolchain was developed and used to convert the cityGML data into Collada format (.dae files) that are natively

supported by the simulation program. The toolchain is shown in Figure 24. The code for the python script (citygmltocsv_all.py) is presented in Appendix A.

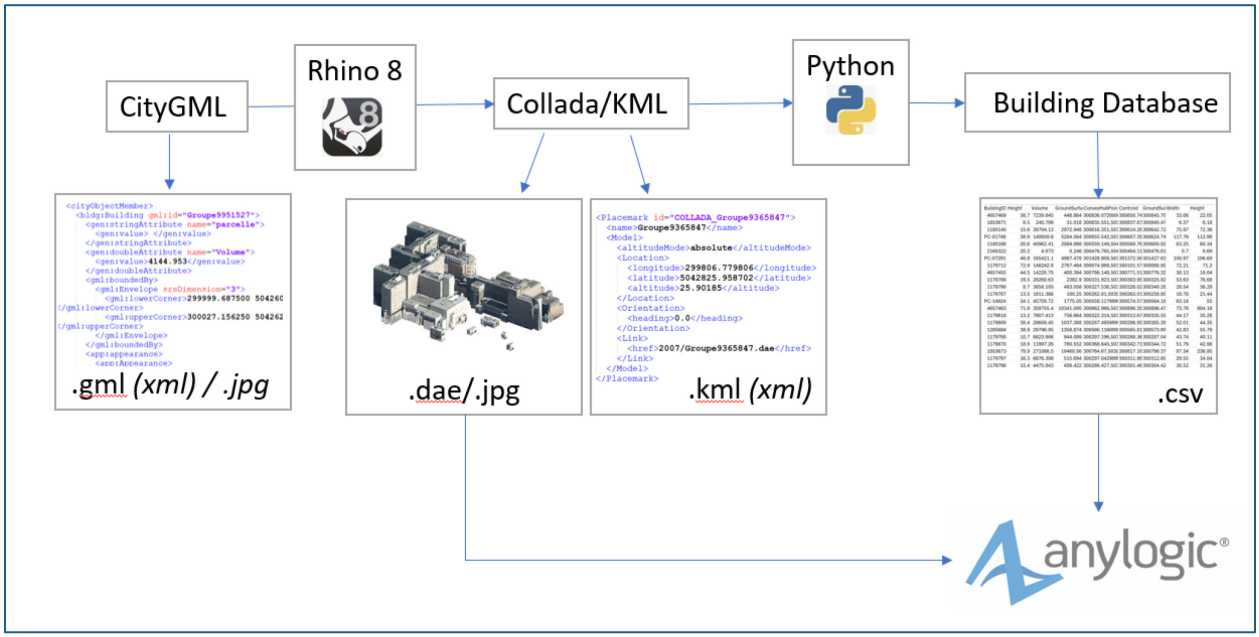

Figure 24 - Building pre-processing tool chain

The CityGML files were first processed by the program Rhino. Using the appropriate export settings in Rhino, individual building files in dae/jpg format were created in addition to a kml file containing a reference to each building along with its location and location of all extracted buildings. The kml file and dae files were further converted using python into a csv file database.

Additional processing steps were included to facilitate the easy and accurate addition of the content into the virtual world. The first step was to create a footprint of each building that can be used for the 2d representation of the building and as an input for further processing. Figure 25a shows an example building, the Concordia Hall building in its native CityGML format. Extracting the points corresponding to the ground level produces a set of complex shapes shown in Figure 25b. As can be seen, the points do not represent the perimeter of each shape; they instead represent triangles (and in some cases higher order polygons). In order to have a simple representation of the building in 2d, first a travelling salesman scheme was attempted using python, shown in Figure 25c, but the shapes were not much better. A second (and successful) attempt using convex hulls is shown in Figure 25d.

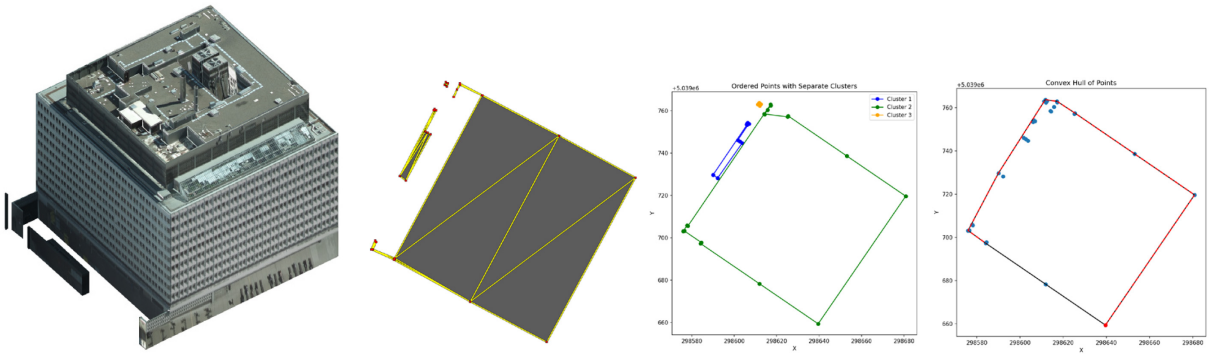

Figure 25 - Concordia Hall building. (a) 3d View, (b) ground level points, (c) travelling salesman technique, (d) convex hull

The convex hull method was chosen as it is the simplest shape that can fully encapsulate any other complicated shape. Using the complex hull, the building ground-level centroid (geometric centre) was calculated and used as the reference location for each building.

The second step was to process the CityGML 3d files and change their coordinate system. The coordinate system of the input content is in NAD83 Canadian Spatial Reference System Zone 8, which is a standard format. This was translated into a coordinate system relative to each building and a related to the footprints' convex hull centroids. This meant that instead of using NAD83 coordinates in meters such as 298734.375000, 5039188.500000, 39.076527, the local coordinates look more like 5.171247, 9.434251, 12.092388 which is more human readable and does not cause issues with the simulation program.

All CityGML files for the island of Montreal were exported in this fashion to create a large dataset of individual buildings. Table 3 shows the fields extracted or calculated by the python scripts.

Table 3 - building database fields

Field	Type	Format	Description
BuildingID	Extracted	Alphanumeric	For referencing the building and 3d model
GroundSurfacePoints	Extracted	NAD83	Accurate 2d representation of building footprint
Centroid	Calculated	NAD83	building footprint convex hull geometric centre
ConvexHullPoints	Calculated	NAD83	A simplified building footprint
LatitudinalLength	Calculated	metres	The "x" length of the building in the virtual world
LongitudinalLength	Calculated	metres	The "y" length of the building in the virtual world

4.4.5. Assembly of the Content

The virtual world content was assembled in 2 ways; manually in the Anylogic presentation editor, and automatically using Java to create dynamic content at runtime. The map, road network, and pedestrian walls were added manually, while the buildings (in 2d and 3d) were added at runtime.. This mix of content creation was thought to be the easiest method to achieve the required accuracy. The biggest challenge was

accurately positioning hundreds of buildings so the automated programmatic approach was optimal for this task. Future work could enable the entire virtual world to be created at runtime, which would involve having a large database of map imagery (in tile form), and extracting shapefile data and metadata from OpenStreetMap to generate the road network.

The coordinate system used is in meters and since the native coordinate system in Anylogic uses the convention of x increasing to the right and y increasing downward, whereas the NAD83 coordinate system increases westward and northward, it was chosen to use the presentation area that is above and to the left of the origin point where x increases (negatively) to the left and y increases (negatively) upwards. Translation of NAD83 coordinates to the local coordinate system is performed as needed. Figure 26 is an example of two cars shown in 2d) with their local positions shown beside them. The values represent the location from the presentation view origin, where 1 unit equals 1 meter. Comparing the two positions shows how the x coordinate increases (negatively) to the left, and the y coordinate increases (negatively) upward.

Figure 26 - Car agents displaying their presentation view positions

The building data is generated at runtime using the building database. A handler function, drawBuildingCentroidsAndConvexHulls(), was written with the help of AI code assistance. The function does the following. It takes as inputs the case study area dimensions, as well as a reference point (the centroid of the Concordia Hall Building) in NAD83 coordinates. It accesses the database, determines all buildings that have their centroid within the case study area, and then adds a 2d and 3d representation of each building to the presentation. The 2d buildings are drawn as PolyLine objects with their centroids drawn as small circles. The properties of the polyline and circle objects are set to display only in the 2d presentation view. The 3d models for all buildings are added to the scene and positioned using their centroids (which their internal coordinate systems are relative to). The function is written in Java and available in Appendix A.

Similarly to the way that buildings are generated, a database on trees is available from the city of Montreal. The database contains information including the position, trunk diameter and species. This tree database was used to add trees into the virtual world for visualization purposes only. The trees were not used for any third-party metric calculations nor were they used for any other purpose.

When the simulation is run it takes some time for the visualizations to load so when the visualizations are required, the rest of the simulation elements (e.g. agents and data collection) are delayed until the content is fully loaded. Figure 27 shows the generated 3d scene including map and 3d buildings and Figure 28 shows the generated 2d scene including map and building polylines.

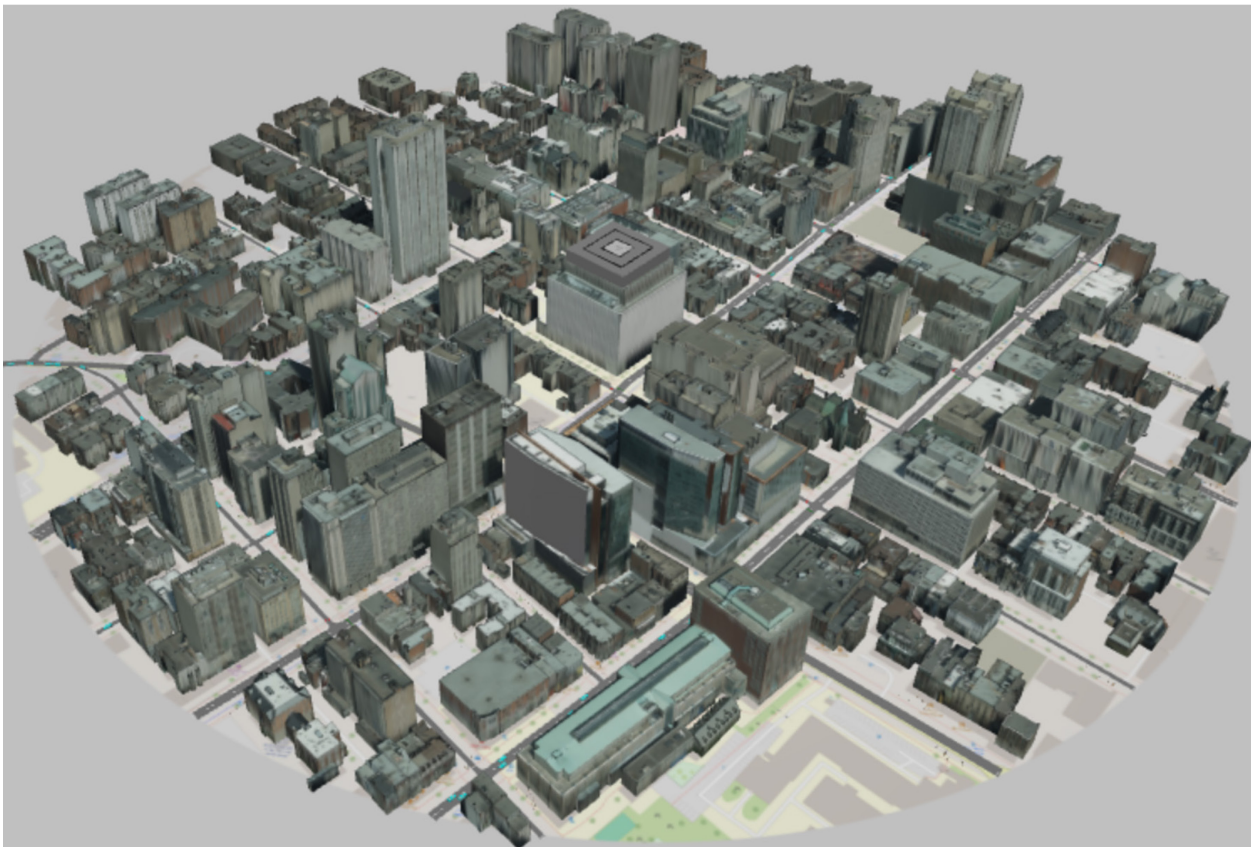

Figure 27 - Digital Twin 3d world view

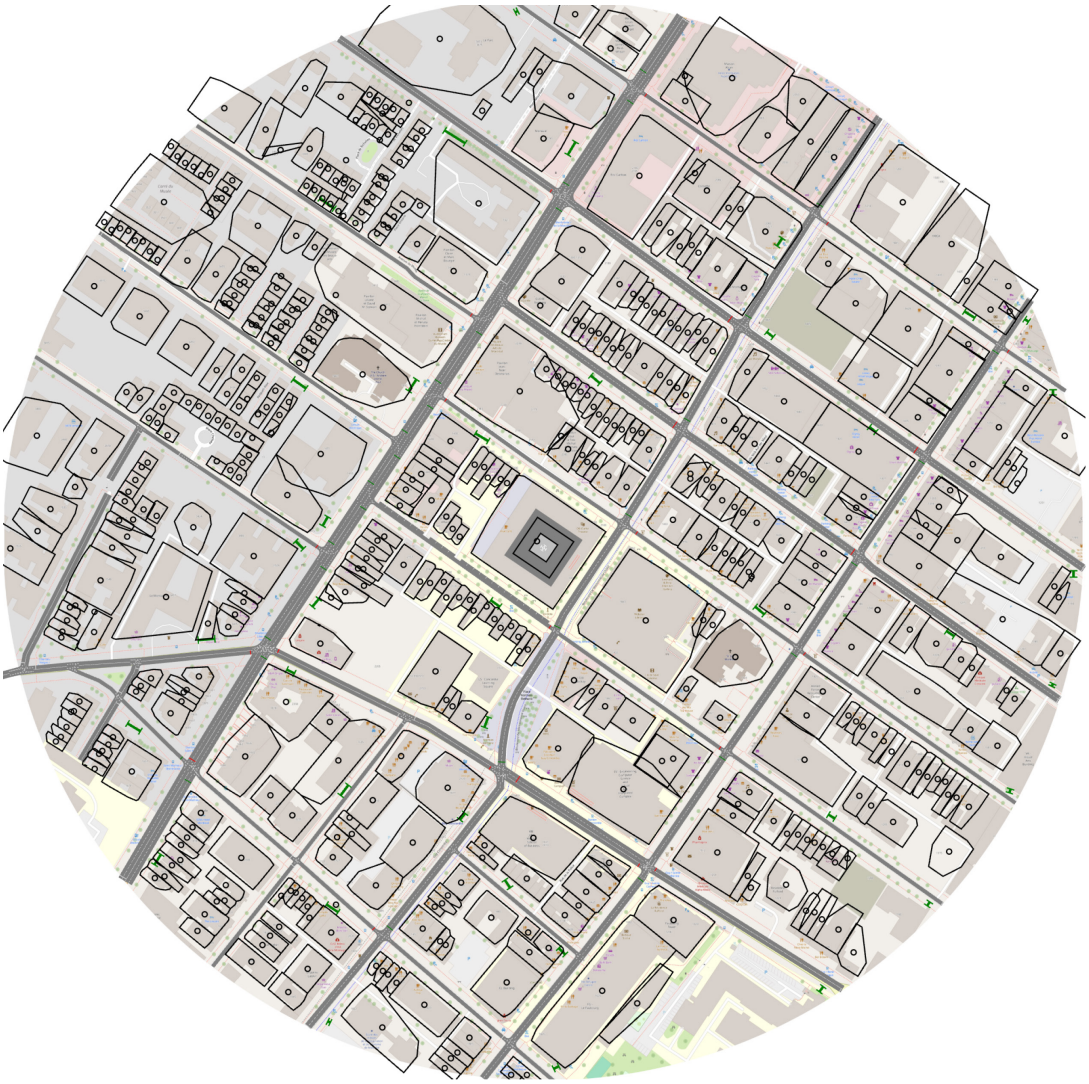

Figure 28 - Digital Twin 2d world view

4.5. Agent Creation

Depending on the architecture used by the selected mode of operation (Case Study vs Digital Twin Prototype), agents are created in different ways. The following subsections describe the methods used to accomplish the research objectives. The intent of the case study architecture is to be repeatable and validate that agent-based modeling can serve as a useful intermediary to decouple the real-world data and generated third-party metrics, whereas the intent of the Digital Twin Prototype architecture is to demonstrate a functional digital twin using available data. Due to this, agents are created differently in the two architectures.

4.5.1. Agent Creation in the Case Study

To validate that it is feasible to pre-select an approach that minimizes third-party impacts, it is important that the third-party agent positions are stochastic and change dynamically with time. Even though the population of agents are meant to be in a quasi-steady-state configuration during the case study, the

movements of individual agents can vary in the order or 10s of meters over the roughly 20 second time frame of the EVTOL approach flights in the case study. The third-party metrics are calculated using the same functions for agent data collection and analysis as used in the Digital Twin Prototype architecture. The metrics calculated at time zero of the case study run are output to a file and then cleared so that the metrics can be calculated again during the case study run and compared directly to the time zero metrics. The case study has multiple purposes: to find an optimal approach heading that minimizes third-party impacts, and to provide a means to compare a reference approach to alternatives. For this reason, in the case study, 64 simultaneous EVTOL agents are created simultaneously so that the 64 different approaches are guaranteed to use the exact same configuration of third-parties.

4.5.2. Agent Creation in the Digital Twin Prototype

To demonstrate that the system is able to represent the real-world data, it is important that the third-party agent positions are a configuration that satisfies the data and can be easily updated every time that the live data stream is called, and less that the agents move realistically between live data updates, although this possibility of prediction/interpolation between live updates is an interesting avenue to explore in further research. For this reason, the third-party agents are positioned statically to represent the data and then removed so that new agents can be created each time the data is updated.

Similarly, it is important that the EVTOL agent flies the optimal pre-calculated flight path, and it is unimportant to compare the results of multiple simultaneous approaches. When an EVTOL is created, the optimal predicted flight path heading is calculated using the current third-party state, followed by a representation of an EVTOL flying the approach. This air traffic management directing the flight based on third-party metrics is the feedback into the real-world that qualifies this as a functional Digital Twin Prototype.

4.5.3. Pedestrian agents

Agent-based human mobility modeling is an active field of study [108]. As such, the goal is to provide a representative agent distribution and movement (as necessary) using a standard human mobility library. Pedestrians are modelled using the Anylogic built-in pedestrian library which provides all the key functionalities required to achieve the objective of this research. The library supports agents that can find an optimal path between two locations, while being constrained by static factors such as walls and dynamic factors such as traffic lights.

When pedestrians are created, they are added to an agent population that contains all pedestrians in the simulation. This agent population is then used to reference individual agents as well as for calculating metrics or creating output files.

4.5.3.1. Pedestrian Modeling (Case Study Architecture)

For the case study architecture, pedestrian agents are created at various locations throughout the study area, and each is individually instructed to travel to a different location. Possible start and end locations are located throughout the study area to create any required coverage of pedestrians. They are prevented from travelling through buildings or crossing roads (aside from intersection) which the path finding algorithm built into the pedestrian library handles and allows them to travel realistically on sidewalks, crosswalks or through open areas. The intended pedestrian density is created by controlling the stochastic algorithms that

choose the start and end locations. These algorithms can be modified during the simulation to achieve the steady state, repeatable pedestrian density required for the study. For instance, the simulation can begin with a large initial pedestrian arrival rate in order to get quickly up to the amount of pedestrians and then can be decreased appropriately to achieve the desired steady state.

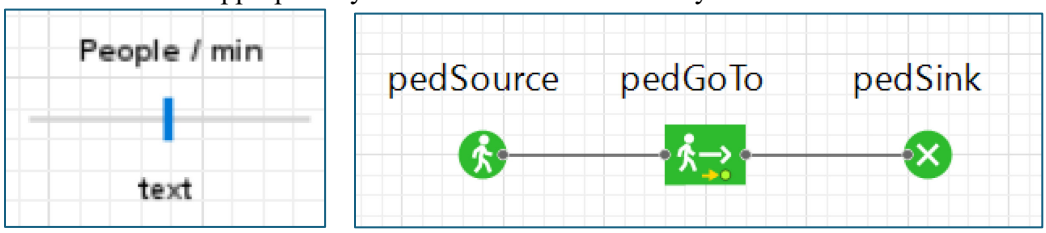

Figure 29 - Pedestrian agents. (a) manual agent addition controls, (b) Agent logic

Creation and control of the agents use a simple block diagram and some supporting functions. The block diagram shown in Figure 29b is as simple as possible, containing a Source, a Goto, and a Sink. The pedSource block creates the agents at a designated arrival rate at the specified starting location. For manual setup of a scenario, the people/min slider shown in Figure 29a is used to set the arrival rate. Changes are activated immediately using the set_rate method of the pedSource object. When running a scenario, events are used to change the arrival rate at a specific simulation times, for instance, after 50 seconds of simulation time, after 100 seconds of simulation time, etc. The pedGoTo block determines where each agent is to travel to. The pedSink block destroys the agents once they have reached the destination location given to them by the pedGoTo.

The list of possible start and end locations is stored in a collection object. The object contains all target lines used by pedestrians in the simulated world. Figure 30 shows the typical modeled spacing of the target lines in the case study area (shown as green lines). The spacing and locations were manually created to create reasonable results when running the scenarios. It is possible that a far more complicated case study area could be created where each building could have its own dynamically created target line facilitating more precise movements of pedestrians. This was not required for this research, so it was not pursued.

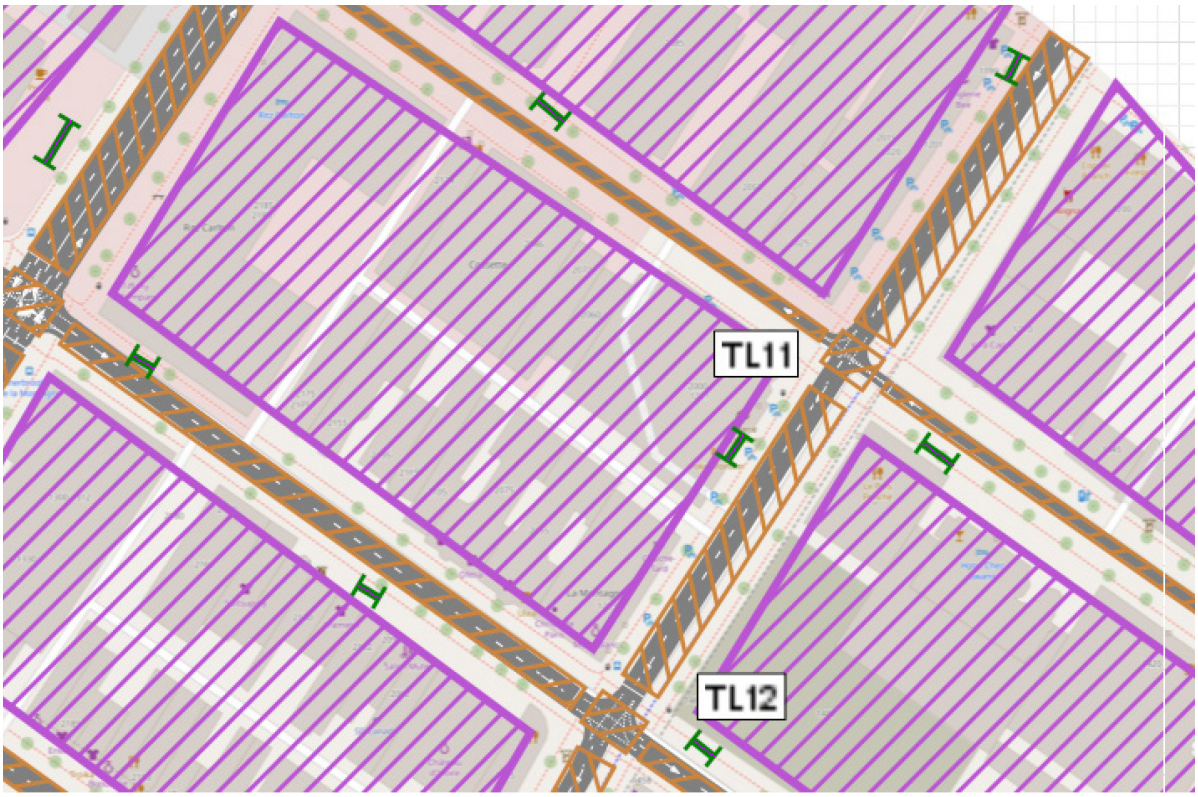

Figure 30 - Pedestrian target lines and pedestrian walls

The starting location TargetLine is assigned when the agent is created by the pedSource block which calls the helper function createPedestrianStartLocation(ped) where 'ped' is the agent. The function provides an algorithm where each TargetLine in the collection is given a statistical weighting. The collection of TargetLines has 2 parameters, the targetLine and the weighting where the sum of all weightings equals 100. Similar to the start location, the destination locations are set by the pedGoTo block using a helper function createPedestrianExitLocation(ped).

In order to create the required pedestrian density map, the statistical model is set so that start and end locations are more likely to create traffic between points where the required density is higher. The same methodology is used to ensure less pedestrian density.

To determine which target line will be chosen, a fitness proportionate selection algorithm is used. The function uniform_discr is used to generate a random integer between 0 and 100. Because the weighting parameters in the targetline collection sum to 100, the target lines are ordered, and their weightings are cumulatively summed. The chosen target line is the one where the cumulative sum of that item in the list is greater than the previous item in the list but less than cumulative sum of that item.

Pedestrian movements are limited to where they can travel by the use of wall objects. Walls are set up to bound the buildings as well as the portion of the road network where pedestrians should not cross. Figure 30 shows the bounding walls around buildings in pink, and the bounding walls around roads in orange. This prevents the pedestrians from passing through buildings and ensures that they cross roads only at pedestrian crossings, creating realistic movement patterns.

4.5.3.2. Pedestrian Modeling (Digital Twin Prototype)

As was described in 4.2.3 there was no live population density data available that met the criteria for the digital twin because the Telus Insights data does not provide real-time data [17]. The API is a subscription service and has restrictions on usage. The intended usage and specific research purpose is part of the subscription, so it was not tested for this thesis. Despite this, an approach to using this data was explored to see whether the data could conceptually be used, if it were able to provide live data without delays.

The case study area was segmented into a square grid. The grid cells were fed via shapefiles to the stubbed version of the Telus Insights API that returns the number of unique devices within each of the grid segments areas. Figure 31 shows some preliminary explorations into this approach where a specified number of pedestrian agents were placed into some of these boxes. Anylogic Rectangular nodes of 25m square were used as they act as a way to contain agents, but also because pedestrian agents can be programmed to spread out randomly within nodes, which is a useful function.

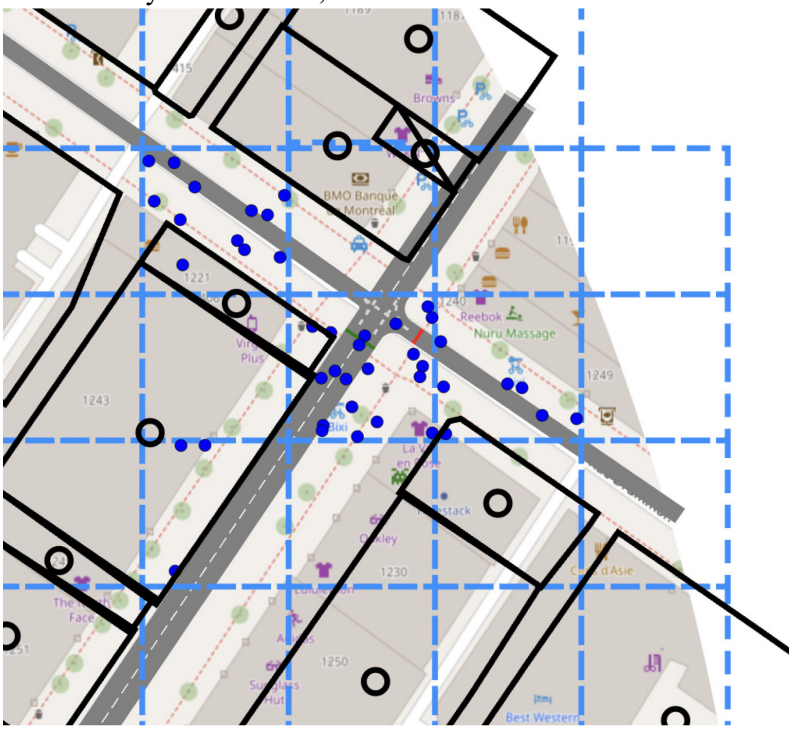

Figure 31 – Static placement exploration pedestrians in 25m square grid showing 4 grid cells with agents injected

After stubbing the reception of API data, the stub functions calls the inject method of the pedSource, shown in Figure 32 and adds agents to each of the applicable grid sections that have a provided number of "unique" people. Every time the Stubbed API is called, the PedWait freeAll() command is called to clear all existing agents so that new agents are created with the new API response data.

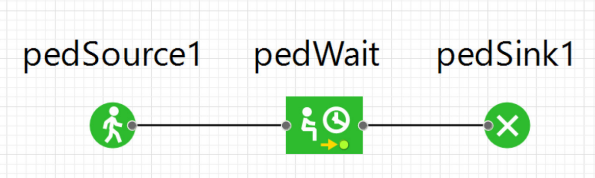

Figure 32 - agent lifecycle block diagram.

The Telus Insights API was not tested directly, so how well it would work, and any particularities in its usage or data is not known, but the approach is worth future exploration. While not explored for this thesis, the footprint of each building is saved in the building database, so the data returned by the stubbed API could be correlated with the building footprint to split the data for people inside buildings with the data for people outside buildings. Many assumptions went into this design, and it is something that could be explored further in future research. The grid approach of translating people/area data into static agents evenly distributed inside each grid segment was successfully implemented for preliminary testing purposes using stubbed (assumed) data. A fully functional implementation was not pursued. Figure 33a shows the user interface controls for the digital twin APIs. The population density option is greyed out to indicate that the feature is neither fully functional nor properly tested.

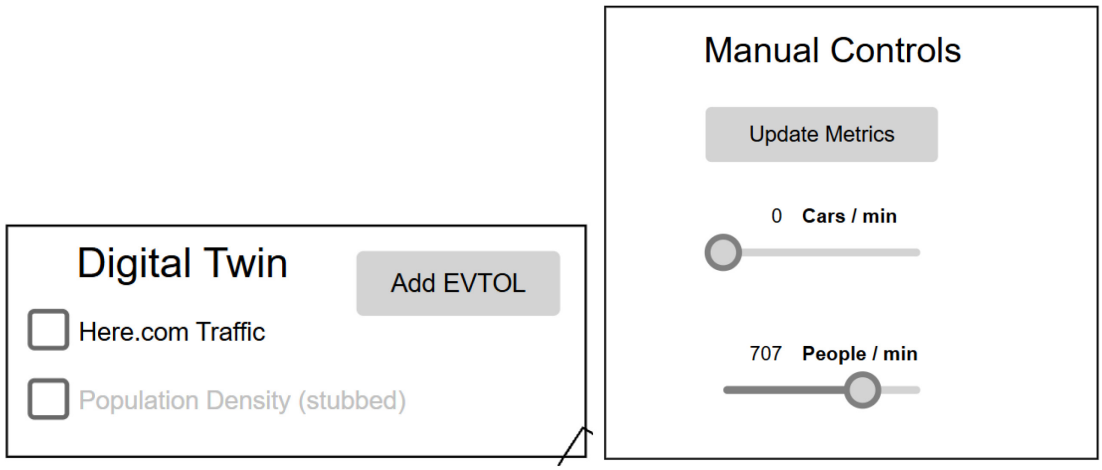

Figure 33 - Digital Twin Prototype a) API controls, b) manual agent controls

Based on this, the pedestrian agent positions in the Digital Twin Prototype free-play mode are only able to use synthetic data created via the manual controls shown in Figure 33b.

4.5.3.3. Pedestrian Agent Visualization

The pedestrian agents have a 2d and 3d visualization for presentation purposes. Each visualization is only shown in the visualization mode selected. The 2d visualization is a simple blue dot shown in Figure 34a. The 3d representation used is a standard 3d human figure from the library of included 3d shapes shown in Figure 34b.

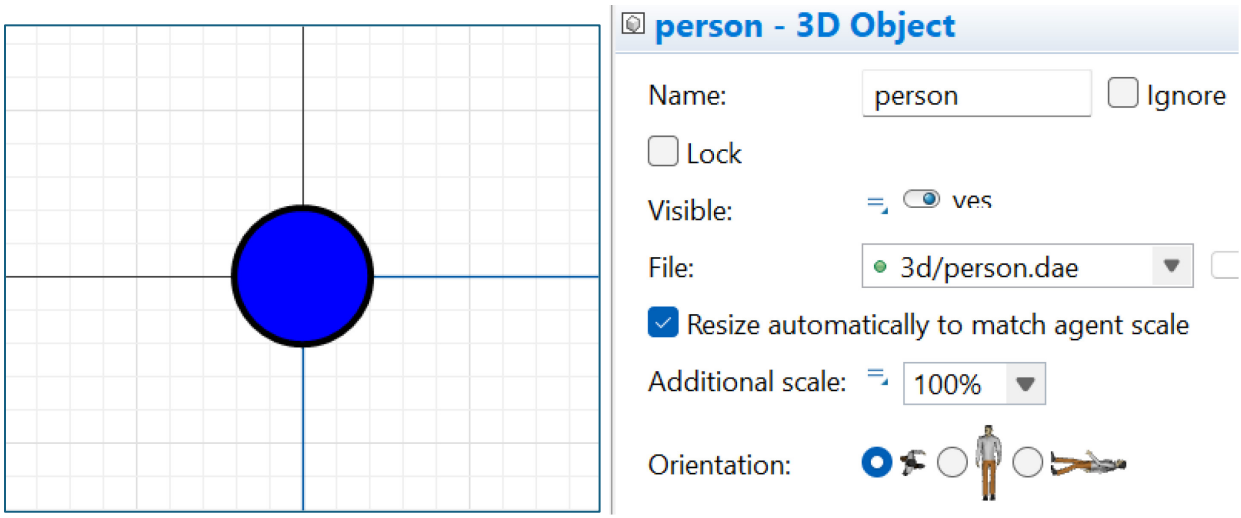

Figure 34 - Pedestrian Agent visualization. (a) 2d visualized as a blue dot, (b) 3d visualized using generic person 3d model

4.5.3.4. Pedestrian Agent variables

To record when a pedestrian is in a crash zone or falling object danger zone, each pedestrian agent saves its own statuses. Each agent has its own copy of a crashZone collection and FODZone collection as shown in Figure 35.

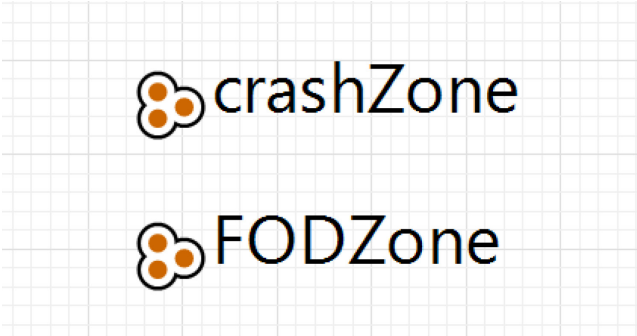

Figure 35 - pedestrian agent collections

To limit the amount of data saved to memory, instead of saving the states of all agents on a schedule these collections contain the simulation time that a pedestrian agent changes state between in a zone and not in a zone. These values are updated every 0.1 seconds by the recurrent events updateFODDangerZones and updateCrashZones, which run the functions updateAgentsFODZone() and updateAgentsCrashZone() respectively as shown in Figure 36 and Figure 37 and their code is available in Appendix A. The crashZone and FODZone collections are used by the metrics described in Chapter 5.

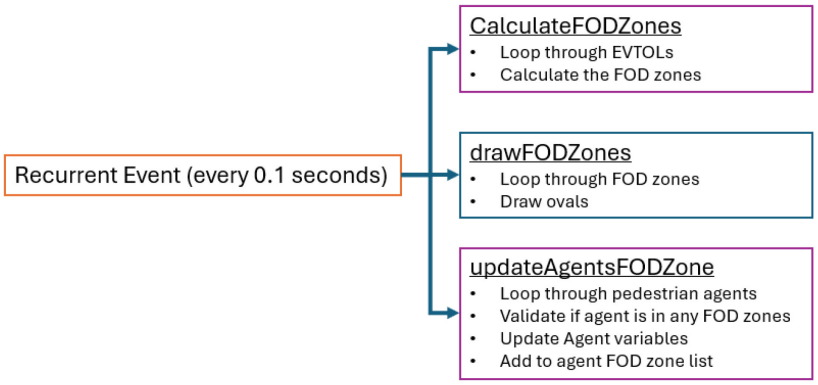

Figure 36 - Falling object zone calculations

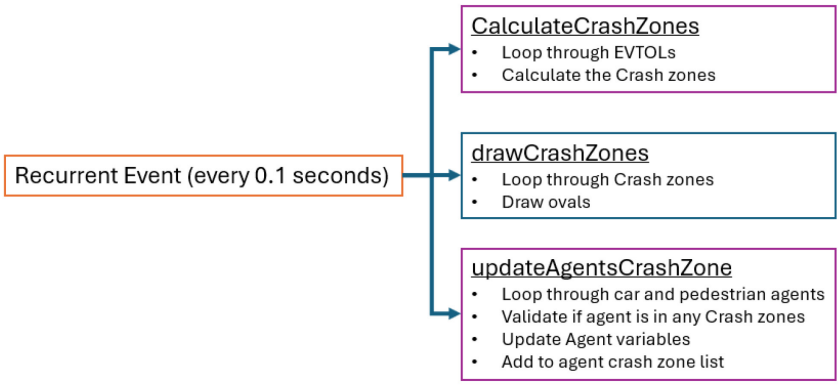

Figure 37 - Ground crash zone calculation

In order to more easily see pedestrian agents when using the Digital Twin interface, a feature is implemented to scale the visual representations larger on-demand. This 3x scaling is visible in both the 2D view and the 3D view. Figure 38 shows the interface controls as well as the resulting pedestrian visualizations.

Figure 38 - large pedestrian visualization control and mode

4.5.4. Road Vehicle Agents

Road vehicles are modelled using the built-in car library which provides all the key functionalities required to achieve the objective of this research. The library supports agents that can find an optimal path between

two locations on a road network, while following road rules including traffic lights, lanes, and avoiding other vehicles.

When road vehicles are created, they are added to an agent population that contains all cars in the simulation. This agent population is then used to reference individual agents as well as for calculating metrics or creating output files.

4.5.4.1. Vehicle Modeling (Case Study Architecture)

For the case study architecture, vehicle agents are created at the periphery of the study area, and each is individually instructed to travel to a different exit location in the study area. The road network is a faithful representation of the road network in the case study area. It was constructed using data from openstreetmap. The intended vehicle density is created by controlling the stochastic algorithm that chooses the start and end locations. This algorithm can be modified during the simulation to achieve the generally steady state, repeatable vehicle density required for the study. For instance, the simulation can begin with a large initial vehicle arrival rate in order to get quickly up to the number of vehicles and then can be scaled appropriately to achieve the desired quasi-steady-state.

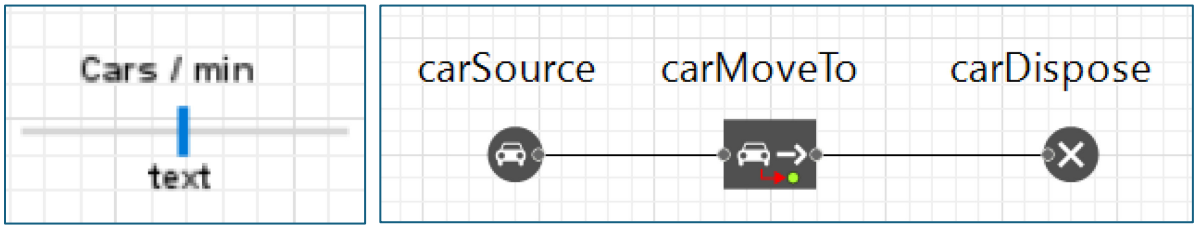

Figure 39 – Road vehicle agents. (a) manual agent addition controls, (b) Agent logic

Creation and control of the agents uses a simple block diagram and some supporting functions. The block diagram shown in Figure 39b is as simple as possible, containing a Source, a Goto, and a Sink. The carSource block creates the agents at a designated arrival rate at the chosen starting locations. For manual setup of a scenario, the cars/min slider is used to set the arrival rate. Changes are activated immediately using the set_rate method of the carSource object. When running a scenario, events are used to change the arrival rate at a specific simulation times, for instance, after 50 seconds of simulation time, after 100 seconds of simulation time, etc. The carMoveTo block determines where each agent is to travel to. The carDispose block destroys the agents once they have reached the destination location given to them by the carMoveTo.

Car type agents can be created either in parking spaces or on roads, where the forward or reverse direction lane needs to be specified. The case study area was created using a standard convention where the forward lane of all roads at the periphery of the virtual world were set as the lane direction into the case study area. This simplifies the setup of all start locations because the road segment reference is all that it required; the lane direction was hence not necessary. The moveTo block can direct vehicles to roads, stop lines, bus stops or parking spaces. It was chosen that all lanes exiting the case study area would have an associated stop line. This allowed the collection of end locations to be a collection of stop lines.

Figure 40 shows an example of a road segment at the periphery of the case study area with a small white arrow indicating the forward direction, and a stop line in the reverse direction.

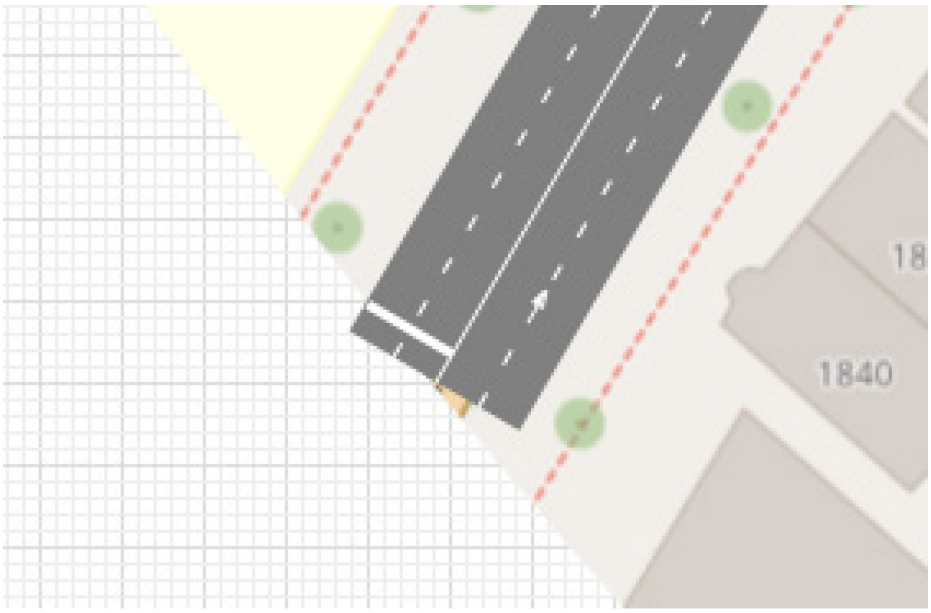

Figure 40 - Road segment at periphery of case study area

The starting location road segment is assigned when the agent is created by the carSource block which calls the helper function createVehicleStartLocation(car) where 'car' is the agent. The function provides an algorithm where each road segment at the periphery of the case study area in the collection is given a statistical weighting. The collection of roads has 2 parameters, the road and the weighting where the sum of all weightings equals 100. Similar to the start location, the destination locations are set by the carMoveTo block using a helper function createVehicleExitLocation(car). The difference being that the destination location collection contains stopline objects instead of road objects.

In order to create the required vehicle density map, the statistical model is set so that start and end locations are more likely to create traffic between points where the required density is higher. The same methodology is used to ensure less vehicle density.

To determine which target line will be chosen, a fitness proportionate selection algorithm is used. The function uniform_discr is used to generate a random integer between 0 and 100. Because the weighting parameters in the collections sum to 100, the locations are ordered and their weightings are cumulatively summed. The chosen location is the one where the cumulative sum of that item in the list is greater than the previous item in the list but less than cumulative sum of that item.

4.5.4.2. Vehicle Modeling (Digital Twin Prototype)

For the Digital Twin Prototype architecture, vehicle agents are created and positioned statically based on the HERE.com traffic API data. Figure 41 describes how the input data is transformed into an agent-based representation. The code for these functions and all related assisting functions is shown in Appendix A.

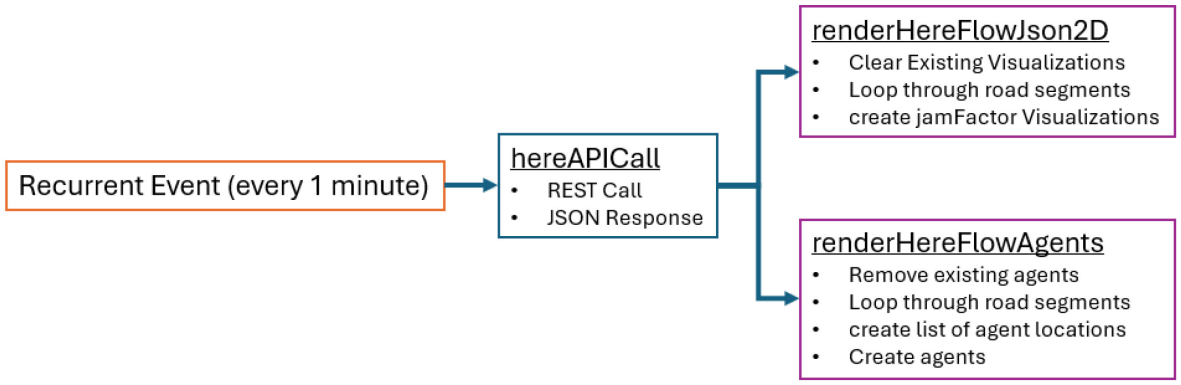

Figure 41 - Traffic API Call to vehicle agent position diagram

A REST call is made every 1 minute to the traffic API. The received JSON response is processed, looping through all road segments and extracting their jamFactors and uncapped speeds (speedlimits). The road segments are polylines described by individual lat-long points. Using the jamFactors and speed limits, a visualization is created and an estimated number of vehicles for each segment is calculated to generate a list of car agent locations, equally spaced at the appropriate x,y (lat-long) positions, are saved.

The documentation for the HERE.com Traffic API does not include any details that would help to convert the jamFactor, or other fields in the JSON response to a number of agents so a simple conversion model was created using the jamFactor and freeflow speed. A maximum distance of 150m was chosen to represent a jamFactor of 150m because it seemed reasonable that at any speed, 150m between vehicle would mean that they are not really impacting each other. A distance of 20m was chosen to represent a jamFactor of 1.0 because vehicle lengths generally range from about 4meters for a car to 23m meters for a tractor trailor [109]. The spacing is further scaled based on freeflow speed of the road segment where 25 km/h has a scaling factor of 0.5, 50km/h has a scaling factor of 1.0, and 100km/h has a scaling factor of 1.5. The calculation for spacing factor is as follows:

$$Spacing = JamEffect * SpeedEffect = \left(150.0 - \text{jamFactor} * (150.0 - 20.0)\right) * \left(\frac{\text{freeFlow}}{50.0}\right) \quad (1)$$

After estimating agent positions along the road segments, the existing vehicle agents from previous API calls are removed, and the new agents are created in their place as shown in Figure 42.

Figure 42 - Resulting agent positioning based on spacing calculation

To create the agents a simple diagram, shown in Figure 43, is used in Anylogic to create the agents, keep them in the model while the HERE data is valid, and then remove them.

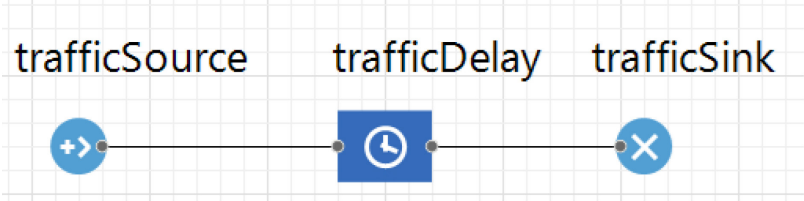

Figure 43 - Static Vehicle creation

The results of this method are acceptable to create a realistic traffic scenario. Figure 44 shows a comparison between the jamFactor visualization, and the created static car agent positions. The here.com traffic API data often has many roads without any data provided. As described in section 4.2.2.2 the data comes from connected vehicle or devices. If there are no devices transmitting road condition data for a given road then the API data excludes those road segments. It is difficult to estimate how many vehicles are likely on those roads, so the Digital Twin Prototype does not put any car agents on those roads. This is something that can be improved in the future using a few possible methods. 1) When better quality data is available there will be fewer road segments without data, 2) collecting and storing API data for a significant amount of time that can be analyzed and a machine learning algorithm applied to fill in the data gaps.

Figure 44 - Visualization of Traffic API jamFactor (colourization) and resulting agent positions a) regular cars, b) large cars

4.5.4.3. Vehicle Agent Visualization

The vehicle agents have a 2d and 3d visualization for presentation purposes. The 3d representation used is a standard 3d car from the library of included 3d shapes. The 2d visualization is a simple cyan rectangle with rounded corners.

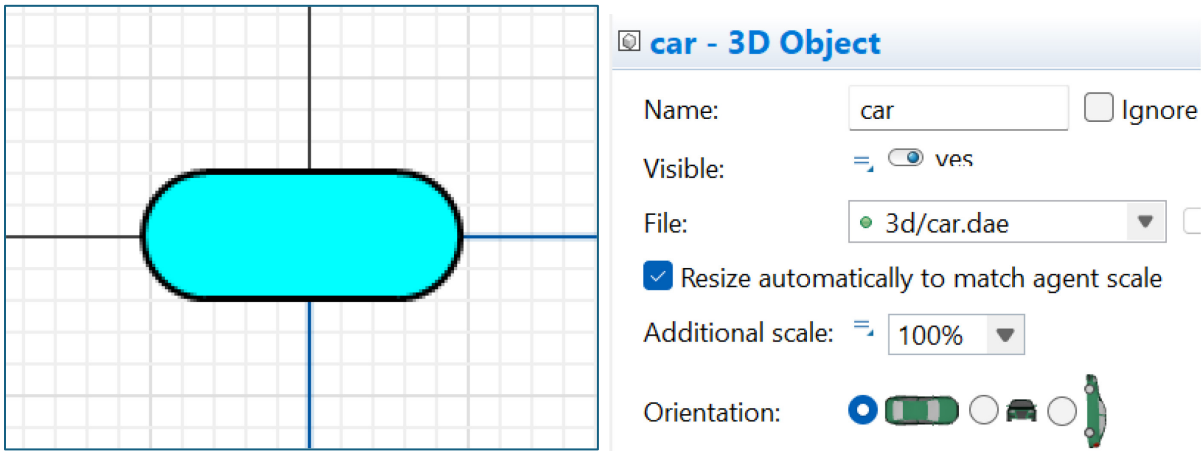

Figure 45 - Road vehicle Agent visualization. (a) 2d visualized as a blue rectangle, (b) 3d visualized using generic car 3d model

In order to record when a car is in a crash zone or falling object danger zone, each vehicle agent saves its own statuses. Each agent has its own copy of a crashZone collection and FODZone collection. These collections contain the simulation time that a vehicle agent changes state between in a zone or not in a zone. The metrics are described in sections 5.1 to 5.4

In order to more easily see car agents when using the Digital Twin interface, a feature is implemented to scale the visual representations larger on-demand. This 3x scaling is visible in both the 2D view and the 3D view. Figure 46 shows the interface controls as well as the resulting pedestrian visualizations.

Figure 46 - large car visualization control and mode

4.5.5. EVTOL Agents

EVTOL vehicles are modelled using the generic agent type which provides most of the necessary basic functionality for this research. The EVTOL agents can realistically move between locations, wait for a certain length of time or until a condition is met and move to other locations.

When EVTOL vehicles are created, they are added to an agent population that contains all EVTOLs in the simulation. This agent population is then used to reference individual agents as well as for calculating metrics or creating output files.

Each EVTOL agent contains variables for their unique identifier as well as their flight plan. These variables are set programmatically when the agent is created. EVTOL agents can be created at any location in 3d space. For the purposes of this research, they are created at the periphery of the case study area at a given height and then fly a given flight path, land on the Vertiport, wait for a specified amount of time and then fly straight to the periphery of the case study area before being removed.

The model supports individual or multiple simultaneous EVTOL agents. For the Digital Twin Prototype architecture where the user manually adds an agent when they want one, a single agent is created. As will be described in section 6.2, for the Case Study architecture an array of 64 EVTOLs are created by the simulation experiment and their flight paths are set based on the permutations of the 4 flight path types and 16 heading directions.

Complex movements (flight paths) can be created and controlled programmatically, but for the purpose of this research, simple flight paths described in section 6.2 with a maximum of two segments are performed. Figure 47 shows the block diagram consisting of two moveTo blocks where the first is programmatically set to the waypoint in the middle of the flight path and the second is the location of the vertiport.

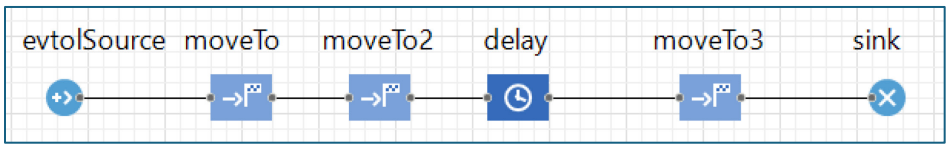

Figure 47 - EVTOL agent logic

As will be described in Chapter 6, the EVTOL vehicle manufacturers are still working towards having fully certified aircraft that can legally fly in Canada. For this reason, the Canadair CL-84 tilt-wing experimental VTOL was used as the visual representation as well as for its dimensions and characteristic in the design of the modeled vertiport. Its weight and dimensions are similar to modern designs. Figure 48 shows the

visual representation of the chosen EVTOL agent in 2D and 3D. The visual representation is scaled appropriately so that its size matches the scale of the rest of the virtual world.

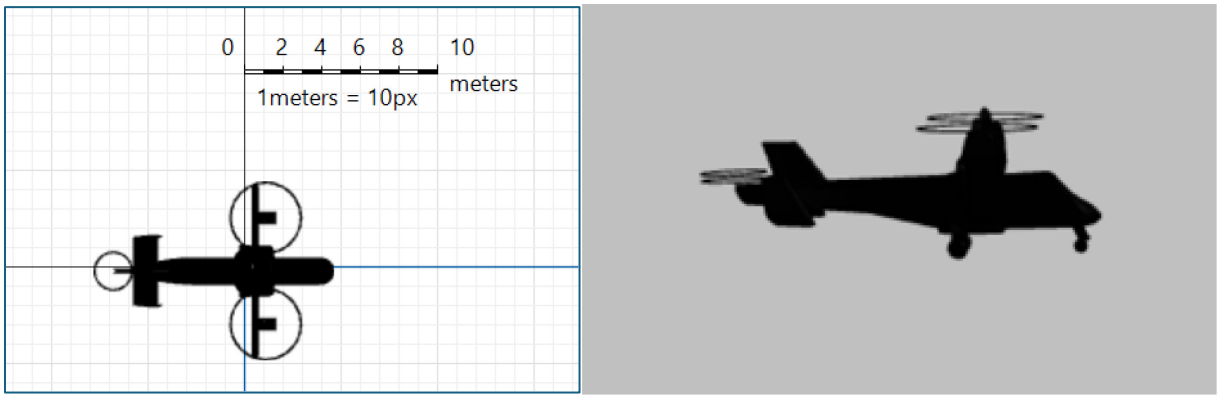

Figure 48 - EVTOL vehicle Agent visualization. (a) 2d top-view visualization, (b) 3d visualization using CL84 3d model

.

4.5.6. Vertiport Agents

Vertiports are modelled as agents Their specifications are modelled based on the reference aircraft from section 4.5.5 and the associated design suggestions from the FAA Engineering brief 105A [20] namely the vertiport dimensions including Touchdown and lift-off area (TLOF), Final Approach and Take-Off area (FATO), and Safety Areas. As per the design brief, these are all based on the dimensions of the chosen reference aircraft. Figure 49a is the 2D view of the vertiport as modelled in Anylogic showing the 3 areas and the standard symbology. The visual representation is scaled appropriately so that its size matches the scale of the rest of the virtual world. Figure 49b is the 3D view of a vertiport located on the roof of the Concordia Hall building.

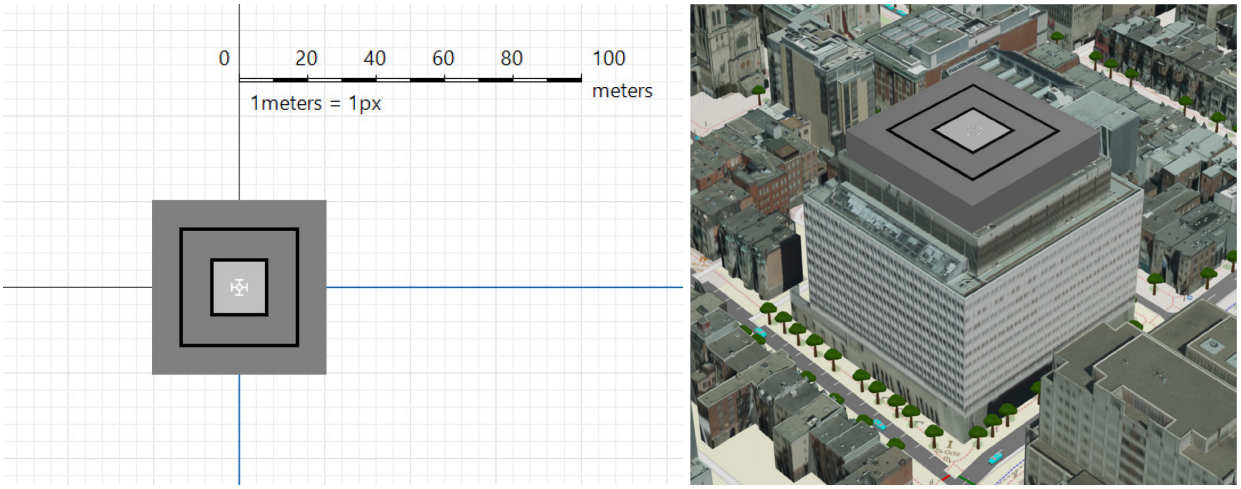

Figure 49 - Vertiport agent visualization. (a) 2D top-view visualization, (b)3D visualization on the Concordia Hall building

Design guidelines are described for ground-level vertiports as well as rooftop and elevated vertiports. As will be described in section 6.3, the engineering brief also provides suggested approach and departure flight paths. Some vertiport design details provided in the engineering brief were considered for the case study but not modelled visually. For instance, lighting and markings related to the approach heading were not

modelled visually, but descriptions of their characteristics helped in understanding the design intent of the reference vertiport.

Modelling the vertiport as an agent has other added benefits. It can communicate with other agents. While not required for the case study presented in Chapter 6, a queueing system and vertiport-based landing management system can be implemented to support a larger study. Further to this, since agents each have their own copy of parameters, variables, and states, multiple vertiports can be added easily to the virtual world at any location in 3d space and function independently.

4.5.7. Building Occupants

Occupants of building are not modelled individually as agents, instead they are modelled as a number of occupants per building based on the total (estimated) floor area compared to maximum occupancy regulations. The number of occupants in the buildings were estimated based on population density data and a technique similar to Choi et al. [111] where the occupants of buildings were estimated using a ground cover vs building cover formula that additionally uses the number of floors of the buildings. As described in section 5.4 the estimated building occupancy is used only for creating a reasonable weighting for the ground crash vs building crash metrics. Since the volume of each building compared to the volume of the other buildings is known, a relative impact measure is actually used.

Chapter 5

Third-party Impacts Evaluation Framework

As was discussed in section 3.1, transportation equity covers many different outcomes like affordability, safety, accessibility, and external costs. The term external costs relates to outcomes that are imposed by someone or something onto others and are normally negative in nature [27]. Outside of the transportation equity domain, the more common terminology used in literature is to describe these outcomes as third-party risk or impacts. In chapter 3, existing research into third-party safety and privacy impacts was reviewed and it was found that Target Level of Safety (TLOS) is a commonly employed calculation to compare to aviation safety standards. These TLOS try to estimate the likelihood of a safety issue happening. While the literature examples were used as inspiration for the metrics developed in this thesis, for this research a different approach is explored. The metrics developed assume that an incident or accident will occur, so the optimization of flight approach paths is presented as a way to minimize the impacts on third-parties. Moreover, metrics are developed using agent-based modeling of third-parties to create more person-centric assessments as opposed to population-level assessments.

The existing UAM-related literature exploring third-party safety impacts describes two interesting classifications of risks: risks associated with a vehicle crashing into a building or terrain, and risks associated with falling objects. Third-party privacy impacts are less studied. For this research, a geometric approach is utilized for modeling the chosen metrics due to the availability of high-quality geometric data used in the construction of the virtual world.

5.1. Ground Crash and Building Crash Safety Metrics

It remains to be seen how well traditional aviation crash modeling translates to EVTOL operations. Despite this unknown, this metric uses the weight-based approach described by Ale and Piers [58]. Since the virtual

world is constructed using highly detailed 3D models of the built environment, there is no need to generalize the crash area like they do for their built-up area formula, so the open area formula was used with building effects handled by the simulation model itself as required. The estimated impact area is modelled based solely on vehicle maximum takeoff weight (MTOW), using their open area formula converted to metric units, equation 2 is used to calculate the debris area:

$$A_{debris} = 0.275 * MTOW[kg] \quad (2)$$

For simplicity, the crash zone is modeled as a circle. The Digital Twin Prototype has the ability to visualize the crash zone as a colourized circular area of the appropriate diameter in the 2d and 3d scenes. Figure 50 displays the crash zone representations in 2D and 3D.

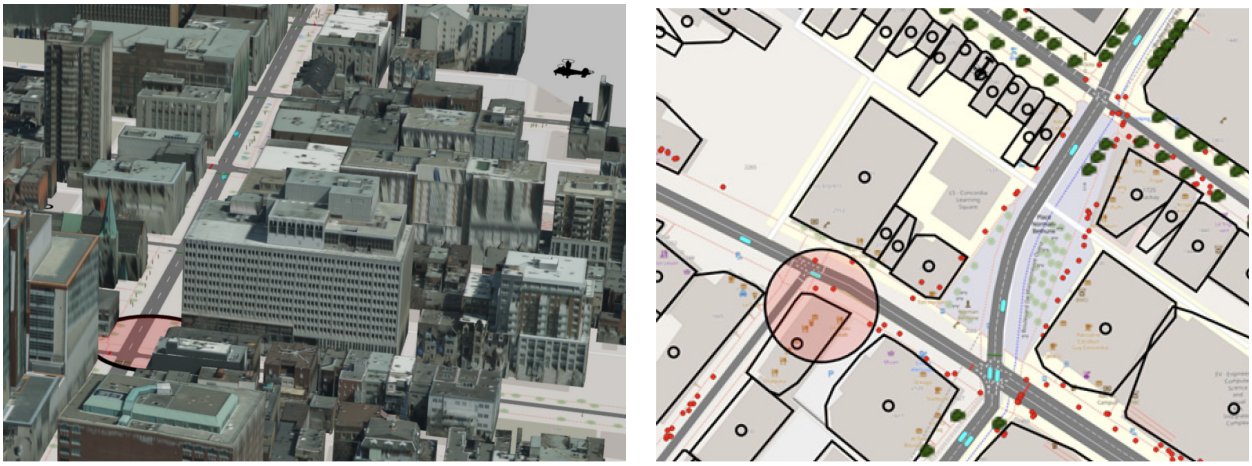

Figure 50 - Crash zone. (a) 3d visualization of crash zone, (b) 2d visualization of crash zone

The EVTOL crash is modeled as a sudden total power loss failure mode. The result is a ballistic free-fall trajectory path from the current position, heading, altitude and speed of the EVTOL to the ground or building. Figure 51 shows a simple illustration of the ballistic path the EVTOL crash calculation assumes.

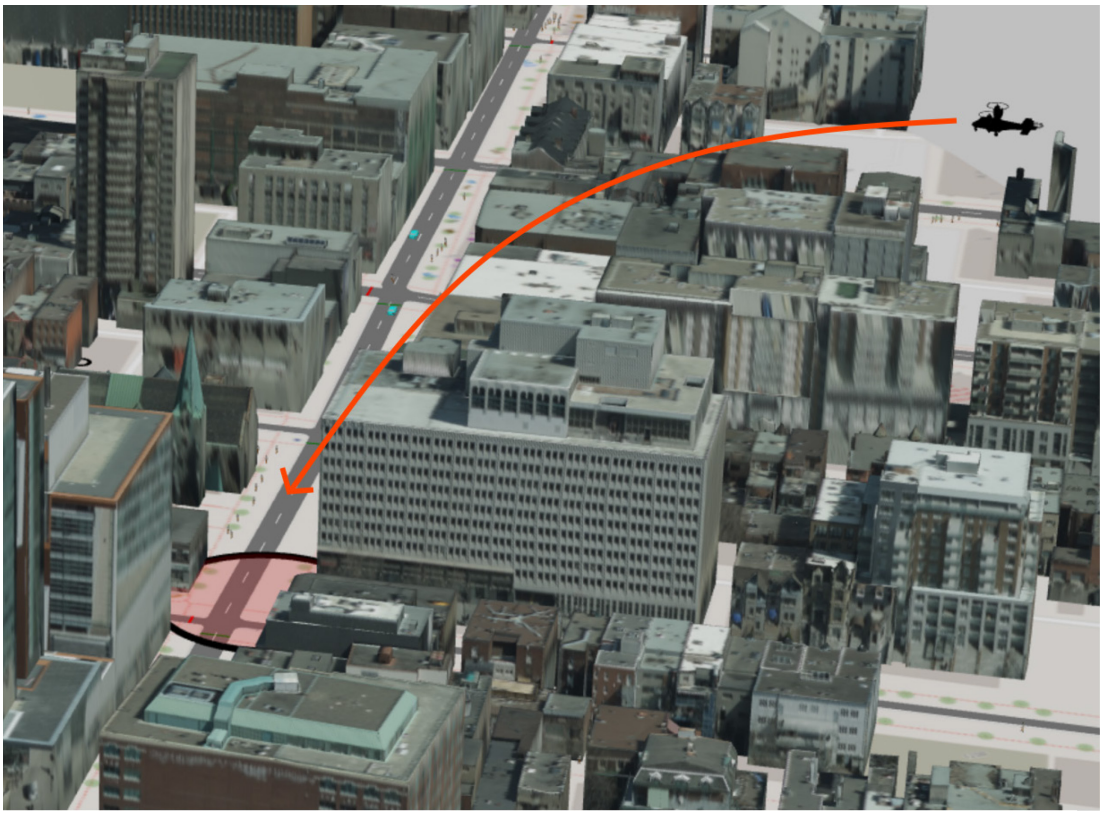

Figure 51 – EVTOL ballistic crash path illustration.

The outcome of this failure mode is an impact into the ground or into a building. Two crash safety metrics are generated based on determining the total number or agents or buildings that will be affected if the vehicle crashed somewhere along its route; the ground crash safety metric ($Safety_{gc}$), and the building crash safety metric ($Safety_{bc}$). In both cases the metrics are normalized by dividing by the total number of agents or buildings.

The ground crash safety metric calculates the impacts on vehicle and pedestrian agents by summing the total amount of agents and the time they each are located within the crash zone area along the flight path. The total agent time in the crash zone summation is then divided by the number of agents in the scenario as a way to normalize the metric so that it becomes a relative, not absolute, metric.

The ground crash safety metric is as follows:

$$Safety_{gc} = \frac{\Sigma t_{agentInCrashZone}}{N_{totalAgents}} \qquad (3)$$

Unlike the ground crash safety measure, the building crash safety measure does not take into account the length of time that each building is in the crash zone. The third dimension makes this determination much more difficult so for purposes of this research, a simplified model is used. The ballistic crash trajectory of the falling EVTOL vehicle is used to determine the buildings that are danger. The metric used was inspired by previous research, such as Choi et al. [111] that developed a method to estimate the fraction of third-parties exposed to harm by determining the building heights and percentage of ground coverage area of open areas compared to buildings. The fraction of third-parties exposed to harm was then used in conjunction with available live population density data to create a population-level metric for third-party

crash risk. For this thesis, a different approach was employed in order to arrive at a more localized metric. The building crash safety metric $Safety_{bc}$ shown in equation 4 does not itself represent the number of third-parties at risk, instead it represents the buildings at risk. When this metric is combined into the final integrated metric, the weighting factor takes into account building occupancy as a way to compare this metric to the ground crash safety metric. To create the metric, all buildings that are directly in the flight path of the EVTOL are determined, their volumes are retrieved from the building database, and the volume of all affected building is summed. This metric value is then normalized by dividing it by the total volume of all buildings in the virtual world. This effectively makes the metric relative for comparison purposes to alternative flight paths in the same case study area. The building safety crash

The building crash safety metric is as follows:

$$Safety_{bc} = \frac{\sum(Volume_{buildingInDanger})}{Volume_{allBuildings}} \quad (4)$$

5.2. Falling Object Safety Metric

Unlike a crash where there is a zone that will injure a third-party located within it, if an object falls from the EVTOL, it could land anywhere within the danger zone. The dangers to third-parties from objects falling or being ejected from EVTOLs are modeled probabilistically. Assumptions need to be made because there is limited vehicle-specific data available that could be used to model accurately the falling object characteristics. It will be vehicle specific whether windows could be manually opened or not, allowing for objects to be intentionally or unintentionally dropped. Propeller arrangement and tip speed could have an impact on how an object would be ejected from the vehicle if it comes into contact with a spinning rotor blade. Because of these factors and others, a simple ballistic model is used to create a circular danger zone around the EVTOL. It is modeled with a maximum outward horizontal velocity of 10m/s and then following a ballistic trajectory to the ground, which means that the danger zone diameter grows larger with altitude. Air effects were not modeled. Figure 52 shows the danger zone location with respect to the EVTOL in 2D and 3D. Figure 53 further illustrates the modeled path extent.

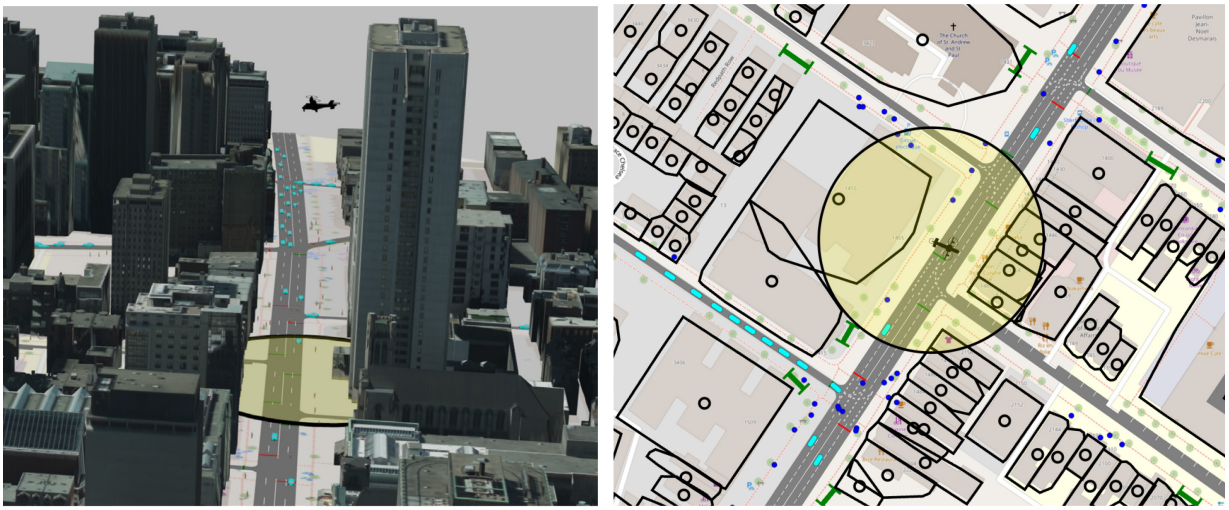

Figure 52 - Falling objects. (a) 3d visualization of danger zone, (b) 2d visualization of danger zone

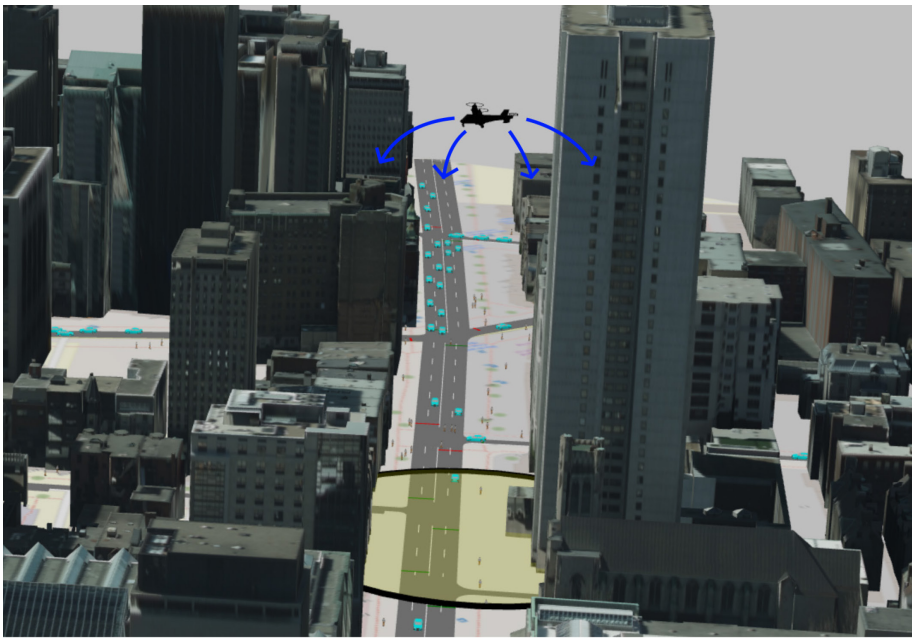

Figure 53 - ballistic falling object illustration

The safety of third-parties located in buildings and in vehicles is considered to be negligible so only pedestrian agents are used for the $Safety_{fo}$ metric. The $Safety_{fo}$ metric described in equation 5 uses the calculated danger zone area and assumes an even danger probability distribution across the zone. For every pedestrian agent, the space they occupy A_{agent} (assumed to be $1m^2$) is divided by the total danger zone area $A_{dangerZone}$ and then multiplied by the time each agent in location in the danger zone $t_{inDangerZone}$ during the approach flight. The summation of the impact to all affected pedestrians is calculated before being normalize by dividing by the total number of agents $N_{totalAgents}$ and the total approach time $t_{approach}$. This effectively normalizes the metric so that different flight paths can be compared equivalently, independent of the total number of pedestrians in the scenario.

The falling object safety metric is as follows:

$$Safety_{fo} = \frac{\Sigma(\frac{A_{agent}}{A_{dangerZone}} * t_{inDangerzone})}{N_{totalAgents} * t_{approach}} \quad (5)$$

5.3. Privacy Metric

Third-party privacy in aviation is not yet something that is thoroughly studied in the literature. There have been some studies published for UAV/drone related privacy impacts due to the growing ownership of private drones. One such study attempts to measure privacy impacts using distance from the drone to people, and the drone camera resolution to determine whether the person could be recognized [54]. This approach is interesting, but recognition based on camera resolution is not the same privacy impact as will be experienced for EVTOL flights. EVTOL vehicles will enable passengers to regularly occupy airspace that has not previously been heavily exploited. Passengers will be able to see new angles towards private residences and spaces including windows, rooftops and other public/private areas that were not feasible previously.

The 3D environment in the digital twin uses highly detailed 3d geometry for buildings but does not include other natural or cultural features such as foliage. For this reason, the metric chosen is a simple model that

determines the number of buildings within a certain distance from the EVTOL vehicle. The height and footprint of each building is used for this determination. As is shown in equation 6, the number of buildings located within this observation distance are summed and then divided by the total number of buildings in the scenario. This effectively normalizes the metric so that different flights can be compared, independent of the size of the case study area. The privacy metric in this case is a measure of the opportunities for a privacy incident, and not a measure of actual privacy incidents.

An opportunity exists for future research to improve this privacy metric by using highly detailed 3D photogrammetry or lidar data (which will include all natural and cultural features affecting sight lines) along a robust line of sight model using raytracing or other efficient techniques.

The privacy metric is as follows:

$$Privacy = \frac{\sum Building_{inObservationZone}}{N_{totalBuildings}} \qquad (6)$$

5.4. Integrated Third-Party Metric

Each of the safety and privacy metrics has its own unique impacts on third-parties, but there is a need to have an integrated metric that acts as a single assessment for optimization and flight path decisions. To create this integrated third-party impact metric $ThirdParty_{int}$ a weighted sum of the other metrics was created. Each metric has a different weight based on their relative impact and importance.

The first weight is the ground crash weight k_{gc}. It is used as the reference for the other metrics so its weighting is set to 1.0. The ground crash metric measure is based on serious injury or death to those in the crash zone, so the weighting of the other metrics is based on their relative severity to this benchmark.

$$k_{gc} = 1.0 \qquad (7)$$

The building crash metric is also a measure of serious injury or death to those affected by the crash, so its weighting k_{bc} will allow for a direct comparison to the number of affected people but since the building crash metric does not yet include a measure of building occupancy, it is included in the weighting instead. The weighting calculation is shown in equation 8. It includes the total volume of all buildings in the case study area, the number of agents, and some constants. The reasoning behind the choice of these values will be explained. The building crash metric is a relative measure of the total volume of affected buildings compared to the total value of all buildings in the case study area, reasons for which were discussed earlier. In order to convert this metric to something equivalent to the ground crash metric, first the total volume of all buildings in the case study area is used. This removes it from the calculation. The next step is to equate the volume of the effected buildings to the occupancy of those buildings. To accomplish this, the occupancy is estimated using 3 constants; the proportion of affected occupants affected by a crash of a small plane, the conversion of volume to estimated floor space, and finally the estimated occupancy load in area/person.

Past research has estimated the mortality rate for building impacts by aircraft to be 0.1-0.2 for small aircrafts[112, p. 49]. To remain conservative, a value of 0.1 is chosen. Converting from building volume to estimated floor space is accomplished by inferring the number of storeys in the building by dividing the volume by 3m/story. Finally, building occupancy is complicated and past research has suggested various

approaches [113], but it is something that cannot be accurately without accurate live data. Since live building occupancy data is not available, an estimation approach was used based on the building volumes. The occupancy for a given floorspace area can be estimated to be $10m^2$/person using the most conservative occupancy value from the National Building Code of Canada [114, pp. 3–36]. When these 3 constants are multiplied by the total volume of the affected buildings, it determines an estimated number of affected third-parties.

The final step is to take this estimated number of third-parties and divide by the total number of agents in the scenario as was done for the ground crash metric. Equation 6 shows the final form of the weighting k_{bc}. In retrospect, the building crash safety metric could have included this calculation, which would have resulted in a k_{bc} equal to 1.0. The resulting integrated metric would have been the equivalent.

$$k_{bc} = \frac{\left(\frac{Volume_{allbuildings}}{3*10*25}\right)*0.1}{N_{totalagents}} \quad (8)$$

The falling object metric is less severe than the crash metrics. Falling objects can vary substantially from small light objects that cause more of a nuisance, such as small birds that struck the EVTOL, to much heavier, multi-kilogram UAVs that would cause series injury or death if a third-party was struck by them. For this reason, a value of 0.25 was chosen for the weighting k_{fo} to show that it is generally less severe than the crash zone, but still important to consider.

$$k_{fo} = 0.25 \quad (9)$$

Finally, the privacy metric is considered to be the least important third-party impact. Privacy issues do not cause physical harm but can still have serious consequences. For this reason, a privacy weighting k_p of 0.1was chosen.

$$k_p = 0.1 \quad (10)$$

Calculating $Thirdparty_{int}$ involves more than simply taking the weighted sum of the 4 metrics. The units of each metric are unique and cannot be compared directly, so min-max normalization can be used prior to applying the weights. By inspection, it is easy to conclude that the metrics have a common global lower bound of 0, but a potentially infinite upper bound that is dependent on the specifics of the study area and settings so since the number of data points from each run are small, the min is chosen as global lower-bound of 0 while the upper bound is chosen to be the local max for the specific test run. While not a standard usage of min-max normalization, the resulting normalized metrics are then multiplied by each of their weights.

The integrated third-party impact metric is as follows:

$$ThirdParty_{Int} = k_{bc} * \frac{safety_{bc}}{safety_{MAXbc}} + k_{gc} * \frac{safety_{gc}}{safety_{MAXgc}} + k_{fo} * \frac{safety_{fo}}{safety_{MAXfo}} + k_p * \frac{privacy}{privacy_{MAX}} \quad (11)$$

Chapter 6

Montreal Case Study

6.1. Case Study Area Creation

The case study area was not determined based on any specific needs other than having a mix of residential, business, low-rise, and high-rise buildings as well as significant foot and vehicle traffic. Many locations could have satisfied these needs, so the case study location was chosen to be a relevant location for the author of the thesis. A circular area (shown in Figure 54), 425m in radius was chosen centered around WGS 84 coordinates 45.49732, -73.57888, which corresponds to the Concordia University Hall building. The size of the case study area was intentionally kept small to represent the final roughly 20 seconds of the approach flight, assuming a reasonable constant speed of 100km/h. The building is about 60m tall with a square footprint of about 50m. These characteristics were ideal for a vertiport as will be explained later.

Figure 54 - Case Study Area

The EVTOL vehicle chosen was the Canadair CL-84. The CL-84 was an early development project experimenting with Vertical Takeoff and Landing (VTOL) and Short Takeoff and Landing (STOL). It was developed in the 1960s by Canadair. Despite this age, its configuration (a tilt-wing VTOL aircraft) is a modern configuration of EVTOL aircraft [11].The main reason this vehicle was chosen for the case study is because there is as yet no commercial EVTOL vehicles that are fully certified for flight operations in Canada and there is no sponsorship by any OEM for this research. A comparison of some past, present, and future VTOL aircraft is provided in Table 4. The length, width, and Maximum Takeoff weight (MTOW) of the CL-84 is generally in the middle of the pack.

Table 4 - VTOL/EVTOL vehicles past, present, and future

Aircraft	*MTOW [kg]*	*length [m]*	*Width [m]*	*configuration*	*Reference*
Joby S4	2404	7.3	10.7	tilt rotor	[115]
Canadair CL-84	6577	16.3	10.6	Tilt wing	[116]
Archer Midnight	3175	15	15	Lift+cruise	[117]
Diamond Volocity	900	11.3	11.3	Multi-rotor	[118]
Wisk Cora	unknown	6.4	11	Lift+cruise	[119]
Eve Air Mobility Eve	unknown	10.3	15.2	Lift+cruise	[120]
Ehang AAV	620	5.73	5.73	Multi-rotor	[121]
Bell MV-75	14000	15.4	24.9	tilt rotor	[122]
Bell V-22 Raptor	23859	17.48	25.8	Tilt wing	[123]
Beta Alia 250	3175	15	15	Lift+cruise	[124]
Bell 429	3402	13.1	11	Helicopter	[125]

The vertiport design specifications from the FAA Engineering Brief are based on the reference aircraft that the individual vertiport is designed for, namely the controlling diameter (D) which is the diameter of the smallest circle that the aircraft's top-view projection can fit inside[20]. As shown in Table 4, the CL-84 has a vertiport controlling diameter of D=16.3m. This means that the Touchdown and Liftoff (TLOF) width will be D=16.3m, the Final Approach and Takeoff (FATO) width is 2D=32.6m, and the safety area is 2.5D=41m. The resulting configuration is shown in Figure 55. As the chosen location (the Concordia Hall Building) is a square about 50m in length, it is a valid placement for the vertiport.

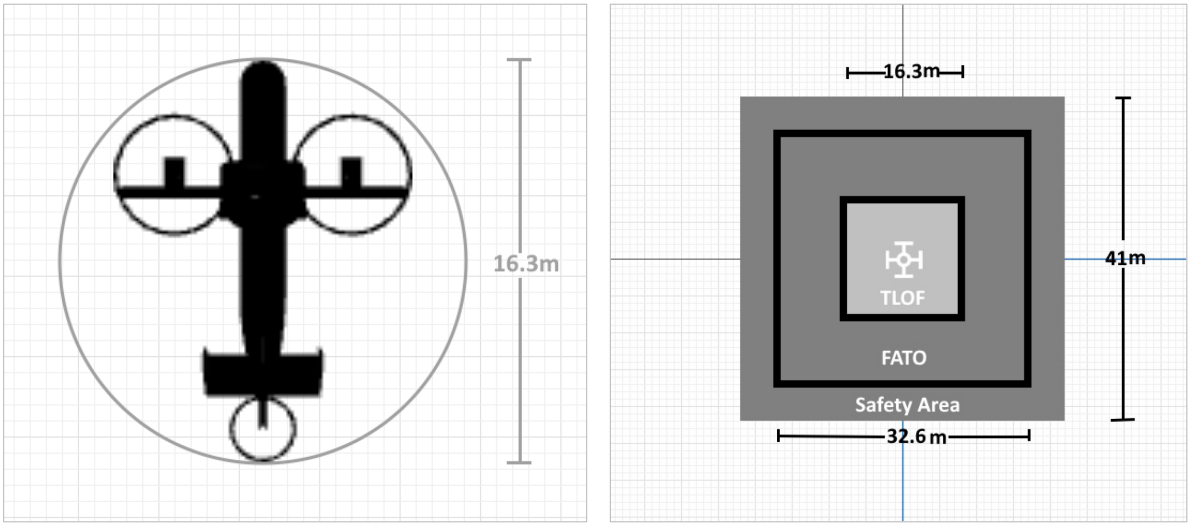

Figure 55 – The case study's a) CL-84's controlling diameter, b) Vertiport design dimensions

The crash metric described in section 5.1 is based on the MTOW in Kg. With a MTOW=5710kg for the CL-84, this results in a crash area of $A_{debris} = 0.275 * 5710 = 1570\, m^2$ as per equation 1. Assuming a circular crash area, the resulting crash area radius of 22.4m is used for the metric calculation.

6.2. Case Study Settings

Two settings were created for the case study to represent different realistic configurations of third-parties. Figure 56 shows the normal day settings and Figure 57 shows the special event settings with large agent visualizations and colorizations for traffic slowdowns and pedestrian heatmaps. As is described in sections 5.1 and 5.3 the metrics related to building occupancy are based on standard occupancy estimations, so the two settings differ only in the patterns of outdoor vehicles and pedestrians.

The normal day setting features distributed traffic and pedestrian patterns. It does not represent any specific historical date or time. The setting was created manually by modifying the stochastic algorithms described in sections 4.5.3.1 and 4.5.4.1.

The special event setting features pedestrian patterns that are meant to be representative of a special pedestrian event happening on St. Catherines street. It does not represent any specific historical date or time. The setting was created manually by modifying the stochastic algorithms described in sections 4.5.3.1 and 4.5.4.1.

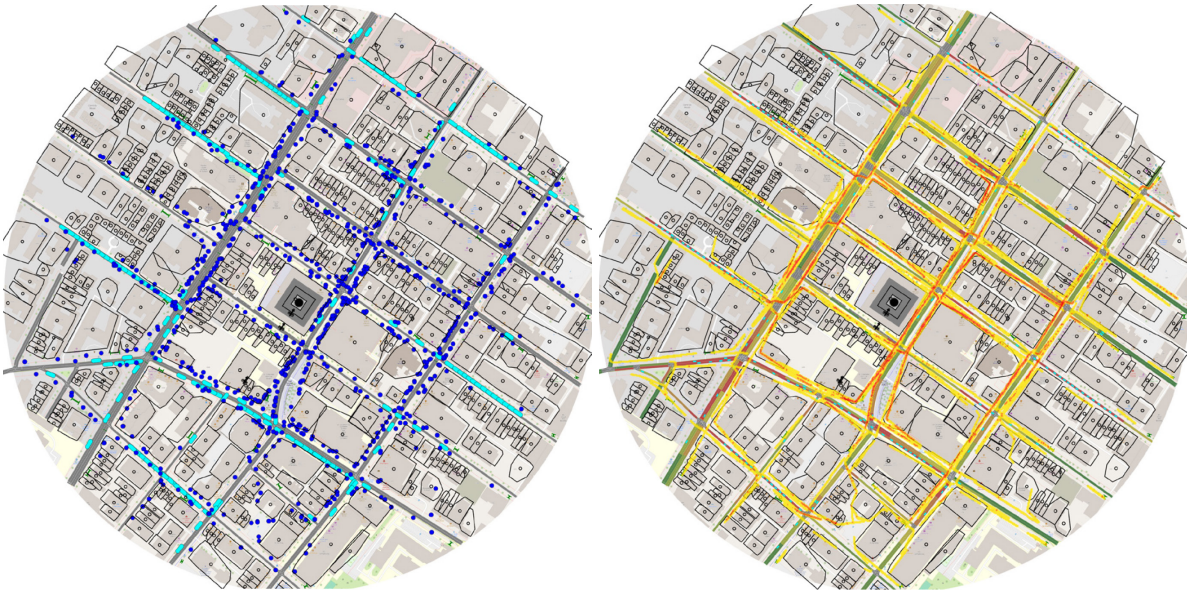

Figure 56 – Normal day Setting. (a) agents (b) pedestrian heatmap and traffic slowdowns

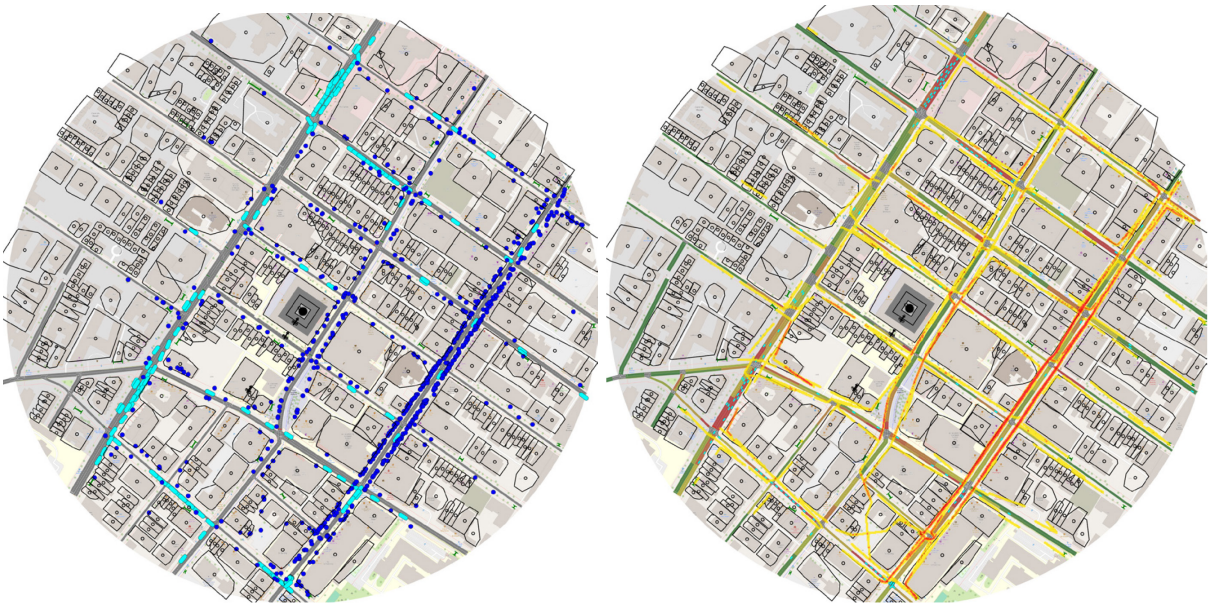

Figure 57 – Special event Setting. (a) agents (b) pedestrian heatmap and traffic slowdowns

6.3. Case Study Approach Flight Comparisons

There are two research questions that the case study will explore; whether realistic approach flight paths can be optimized to minimize third-party impacts, and whether the tool is useful to analyze policy decisions that describe fixed flight characteristics. The baseline flight path provided in the FAA Engineering Brief describes a preferred approach heading aligned with the prevailing wind direction in the area, and a preferred approach descent rate of 1:8 [20]. A descent rate of 1:8 is shown in Figure 58. There are many challenges for traditional aviation, including low visibility conditions which means that airports and heliports often are equipped with electronic guidance systems, lighting, and markings to help guide aircraft in for safe landings. With modern technology, there is no reason that similar systems could not be developed to guide air vehicles in via any approach path. The historical prevailing wind direction in Montreal is from WSW (247.5deg) [126], meaning that the preferred approach heading in the design of a vertiport is 67.5deg from North. Figure 59 shows in blue the preferred approach heading and descent rate.

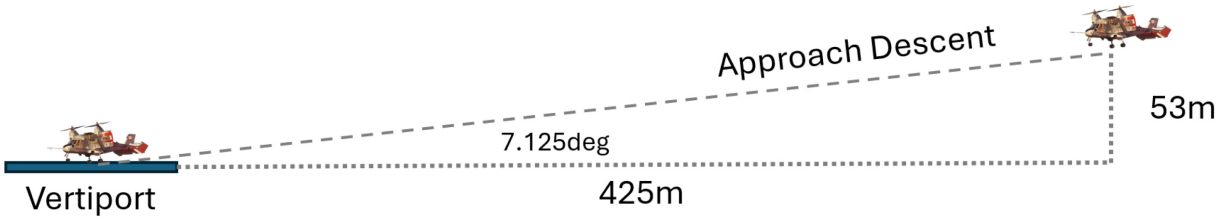

Figure 58 - Implementation of Baseline Approach from FAA Engineering Brief 105A [20]

The case study area is a circular area 425m in radius. EVTOL vehicles will likely (although this is a generalization and unproven) be more maneuverable than traditional aviation vehicles, so constant heading approaches were chosen instead of more complex flight paths using a series of lat/long waypoints. Future research could explore the use of more complicated flight paths and larger case study

areas. To find the optimal approach heading a selection of 16 evenly distributed headings (22.5deg apart) were chosen to cover the complete 360 degrees. It is possible to perform an analysis on as many approach headings as desired, but for a 425m flight, the starting positions of the 16 flight paths are only 166meters apart. It was deemed acceptable by the author that this level of precision would achieve the research objectives.

For each of the 16 approaches headings, 4 descent paths were compared. The baseline approach descent of 7.125deg, a 45deg approach descent, a low-altitude horizontal flight with a rapid descent at the end, and a higher-level horizontal approach flight with a rapid descent. These result in 64 scenarios that are tested simultaneously for analysis and comparison. Figure 59 show the 64 approach scenarios and highlights the baseline approach in blue.

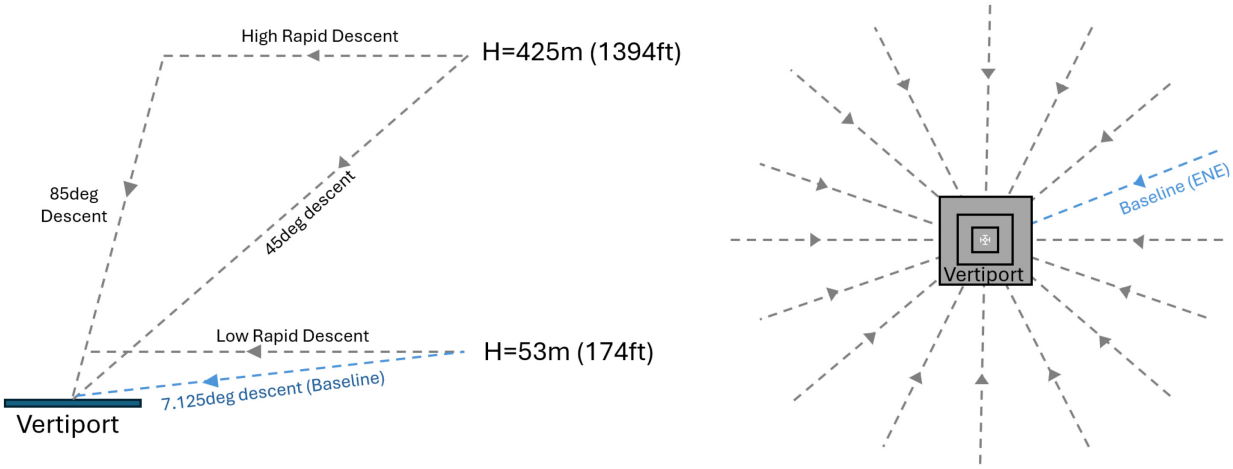

Figure 59 - 64 approach scenarios. (a)4 descent profiles, (b) 16 approach headings

6.4. Results

The case study settings were simulated, resulting in 128 total EVTOL approach scenarios. The third-party impact metrics were calculated. Visualizations of the metrics were generated using radar charts. Radar charts are not normally used to represent real angular positions, but in this case, they were used to visualize the metrics in a way that is easy to understand and superimpose on top of the circular case-study area. The radar charts show all 64 approaches for each setting, The 4 approach descents are shown as the data series of the chart, while the 16 approach headings are positioned on the spokes.

Each of the metrics is shown below and when applicable, the normal day setting is shown beside the special event setting for comparison purposes. The building crash and building privacy metrics are not affected by the case study settings so only a single instance of the resulting radar chart is shown.

6.4.1. Falling Object Safety metric

As described in section 5.2, the falling object metrics are based on a danger zone calculation that varies with altitude. The results of the falling object safety metric are shown in Figure 60. In both the normal day setting and special event setting, similar trends can be seen. In both settings, the metric increases with increased altitude. This can be explained because the falling object danger zone is larger for higher altitude flights. The directionality of the metric also decreases with altitude in both settings. This can also be explained by a natural averaging effect that happens when the danger zones are larger with significant overlaps.

The results show that the falling object safety metric in both settings can yield an optimized flight path, especially for low altitude flights.

Figure 60 - Falling Object Safety Metric. (a) Normal day setting, (b) Special event setting

6.4.2. Ground Crash Safety metric

As is described in section 5.1, the ground crash area is based solely on vehicle weight, meaning the same ground crash area will occur independent of initial altitude. What changes is the amount of time that agents are located in the potential crash debris areas. The results for both settings show some of the same trends. The metrics are nearly identical for the 2 approaches with consistent descents. This can be explained because the time in the crash zone for individual third-parties will be very similar in both cases. The metrics have variability when it comes to the rapid descent scenarios. When the vehicle performs its rapid descent, it has crash areas near the vertiport that exist for longer times. In both settings, there is a group of pedestrians to the east of the vertiport, and these pedestrians will be in the crash zone for a longer time than pedestrians earlier in the flight.

Both settings show significant directionality effects meaning the safety risk can be reduced significantly by optimizing the approach flight path heading.

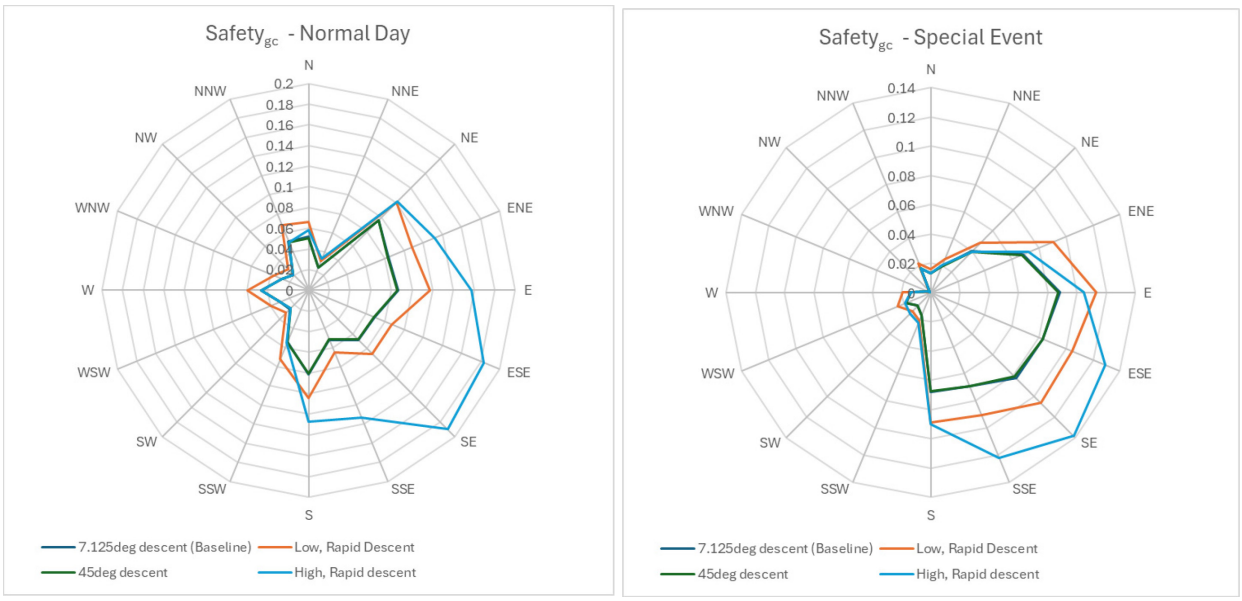

Figure 61- Ground Crash Safety Metric. (a) normal day setting, (b) special event setting

6.4.3. Building Crash Safety metric

As is described in section 5.1, the building crash metric is based on estimated building occupancy for all of the buildings that the flight path flies directly over. Unlike the metrics that are based on vehicle and pedestrian positions, the building crash safety metric uses a fixed calculation of building occupancy based on floor space and does not take time in the danger zone into account so the metric will not change with settings or descent characteristics. This simplification could be improved for future research. For this reason, only 16 datapoints are necessary to represent all 128 scenarios. The directionality of the metric is illustrated in Figure 62 and shows that the metric is useful for optimizing the approach heading.

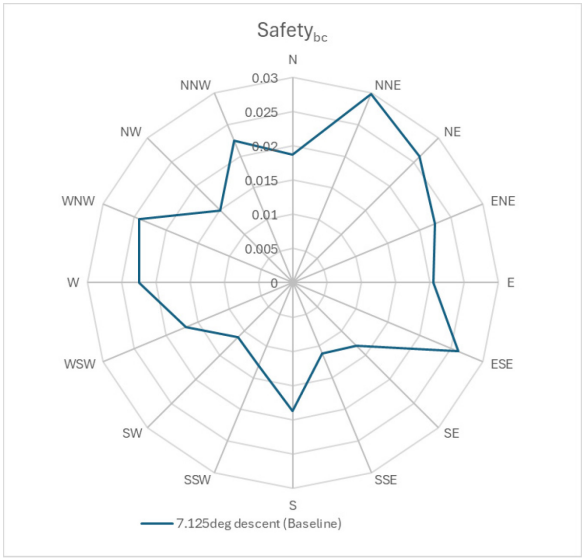

Figure 62 - Building Crash Safety Metric

6.4.4. Building Privacy metric

As is described in section 5.3 the building privacy metric is based on proximity to buildings. It uses the building locations and geometries (including their heights) to determine the minimum distance the EVTOL is to each building. Similar to the building crash safety metric, the building privacy metric is the same for both normal day and special event settings because it is not impacted by the vehicle and pedestrian configurations. Also similar to the building crash metric, time is not part of the metric calculation. The metric is a simple metric based on the proximity of the EVTOL to buildings along the approach path. Buildings are 3d objects with heights so there is a strong altitude component. Lower altitudes generally result in larger and more varied metric values. At low altitudes, the metrics show clear directionality demonstrating that the metric can be used for optimizing flight paths. Figure 63 shows the privacy metric for both settings. All 64 approaches have a unique metric value.

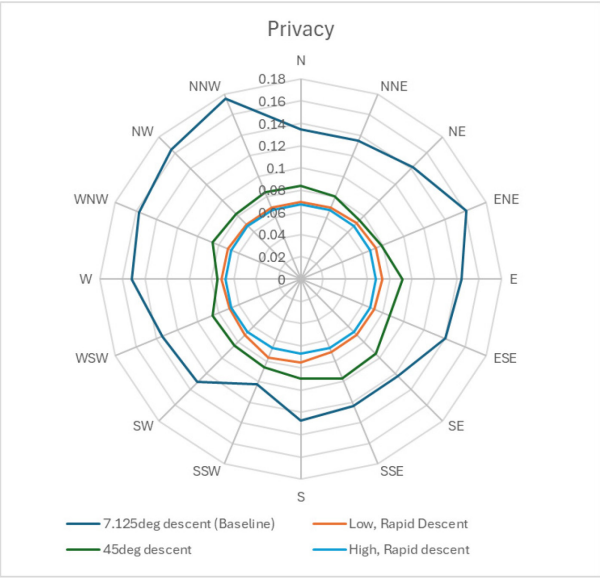

Figure 63 - Building Privacy Metric

6.4.5. Integrated Third-Party metric

The integrated third-party metric is the metric used for the digital twin prototype routing suggestion. As described in section 5.4, it is a weighted sum of the other 4 metrics. The weights are based on relative impacts. As can be seen in Figure 64 the metric has directional characteristics. The importance of this metric for routing can easily be seen in both settings. In each setting, the scenario with the greatest third-party impact, is over 4 times larger than the scenario with the least third-party impact. These results show that significant reductions to third-party impacts are possible with routing.

These results also show that the baseline approach is not the optimal approach. In both settings, the baseline approach (from ENE) is the second worst approach heading, and the baseline descent rate is also not optimal

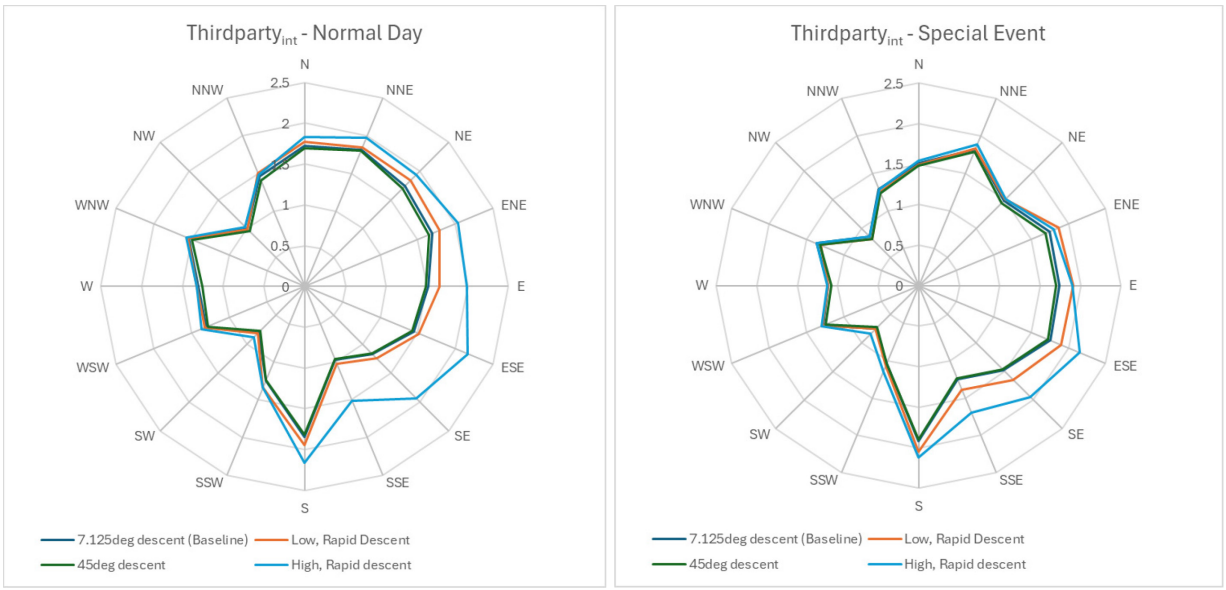

Figure 64 - Integrated third-party metric. (a) Normal day setting, (b) Special event setting

To provide a simple qualitative assessment of the results, metrics can also be super-imposed over the setting results as shown (for the integrated metric) in Figure 65. Upon initial inspection, the results for Normal day (Figure 65a) and Special Event (Figure 65b) look similar and produce the same optimal approach heading. Both exhibit a higher metric for the rapid descent cases than the consistent descent cases. In a rapid descent, more travel time is spent close to the vertiport as the EVTOL descends at a steep angle. In both settings, there is a large group of people just to the east of the vertiport, which will be in the crash/FOD zones for longer than the consistent descent approaches. This is seen clearly in the metric results. There are other clear similarities which can be explained mostly by the impacts that the building privacy and safety metrics (which remain constant for both settings) have on the integrated metric.

Figure 65 - Fusion of third-party metrics and associated setting results for a) Normal Day, b) Special Event

6.5. Result and Methodology Validation

To validate the methodology of using a digital twin (including agent-based representations of population data) to provide guidance on minimizing third-party impacts despite the dynamic nature of the agents during the test, three key elements are important to validate: input data representation, metrics sensitivity/predictability, and metrics repeatability. The case study covers only the last 425 horizontal meters of the approach which takes approximately 20 seconds to complete (depending on the descent profile) so there is not a long timeframe for the placement of agents to change appreciably, but there will be differences.

6.5.1. Input Data Representation Validation

The first key element, input data representation, refers to how the live input data from the Here.com Traffic API is implemented in the Digital Twin Prototype architecture and transformed into individual agents as described in sections 4.5.3.2 and 4.5.4.2. The raw data for individual pedestrians or vehicles is not available from the data providers. Unfortunately, this means that there is no way to compare the created agent positions with real-world data, so what will be investigated is whether the methodology of creating an equivalent agent-based representation can generate similar third-party metrics to achieve the objectives. Using the case study repeatability results from section 6.5.3, a visual comparison is performed showing the differences and similarities between multiple model runs and the resulting outputs.

Using the built-in Anylogic car library roadNetworkDescriptor class a visually understandable vehicle speed map can be generated; the default settings were used for visualization. Similarly, using the built-in pedestrian library heatmap functionality, a pedestrian heatmap can be generated; the default settings were used for visualization. Figure 66 shows a portion of the case study area at the end of runs #9 and #10. The vehicle and pedestrian agent positions can be seen in the left-hand images, and the resulting vehicle speed maps, and pedestrian heat maps can be seen in the right-hand images. The top row is run #9 and the bottom row is run #10.

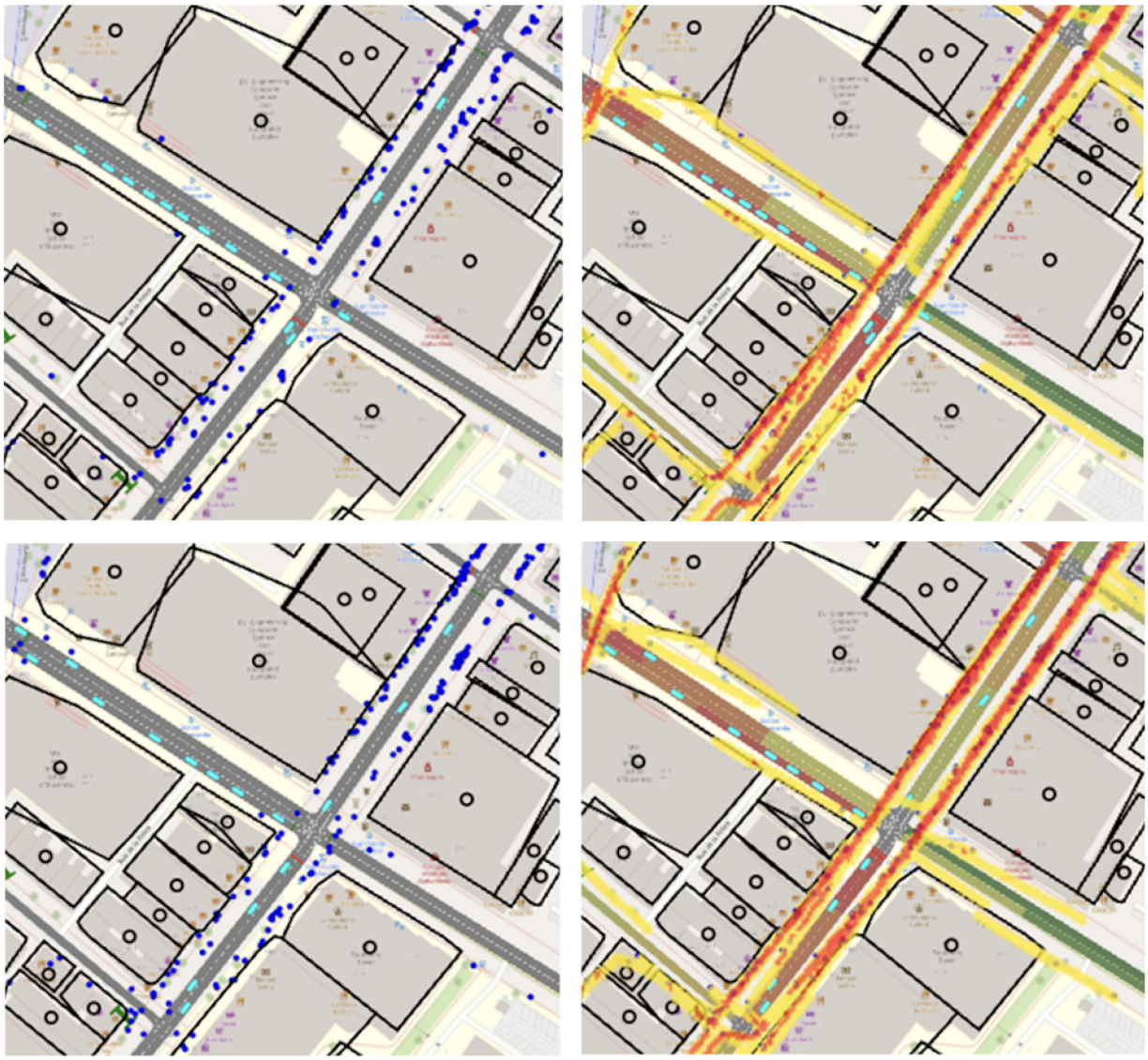

Figure 66 - Agent positions with resulting vehicle speed maps pedestrian heat maps for runs #9 (top) and #10 (bottom)

A visual comparison easily shows that the vehicle and pedestrian agent positions are different between runs, but the resulting speed maps and heat maps are quite similar. Further analysis is conducted in section 6.5.3 showing how much the resulting metrics are impacted by these differences between runs.

6.5.2. Predictability Validation

The first key element, predictability, refers to whether a decision could be made prior to the approach flight that remains valid once the flight has been completed. This premise is important because it validates that choosing an approach that should minimize third-party impacts, can. To test this, the Case Study architecture was used. The integrated third-party metric was tested because it is the final output for this case study. The special event setting was used because it exhibited more variation based on heading than the normal day setting. Predictions for the metric were calculated at time 0 (based on the position of all agents at t=0) for each of the 16 approach heading scenarios using the baseline descent profile. These predictions were compared to the output of the model run. The results are shown in Figure 67. The

predictions varied by a maximum of 12% and the predicted optimal approach heading that minimizes the metrics is shown to be the same as the final optimal approach.

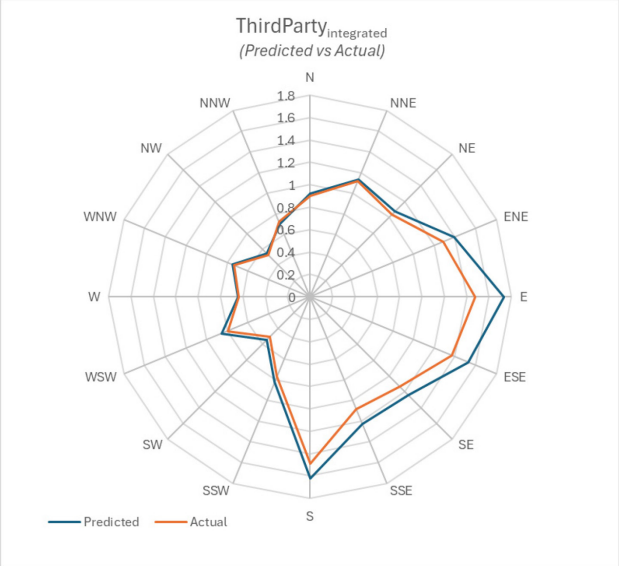

Figure 67 - Predicted vs Measured Integrated Third-Party metric for baseline descent profile using special event setting

6.5.3. Metric Sensitivity and Repeatability Validation

The third key element, sensitivity, refers to the whether metrics will yield the same recommendations for optimal flight path independent of the exact configuration of third-party agents in the test. The metrics were developed to be agent-centric. The input data sources for third-parties is available in the form of population density and traffic slowdown factor so by converting to an analogous agent-based representation, there are an infinite number of valid agent locations that will satisfy the data. Since there is no way to validate the true positions of people, it is important to validate the sensitivity to differences in agent positions.

The stochastic nature of the case study architecture agent creation algorithm means that, by design, each run of the model will yield slightly different results since the agents are not identical between runs. The Digital Twin Prototype architecture by contrast will always generate the same vehicle positions for the same input data. The ability to create an accurate (enough) representation of population data that produces repeatable third-party metrics usable for decision making is critically important to trust the methodology. The special event setting of the case study was run 10 times and a repeatability analysis was performed.

The results of all 10 runs were collected and the 7.25degree baseline approach descent rate was compared for all 16 headings. The results of the Integrated third-party metric are shown in Figure 68a. The stochastic nature of the agent generation algorithm means there is no guarantee that the number of agents will be identical each run, so to ensure that the metrics are scaled for comparison, the results were then normalized by dividing each datapoint on the radar chart by the summed total of the metric for each run as described in equation 12.

$$Integrated_{int\_i\_norm} = \frac{ThirdParty_{int\_i}}{\sum_{j=1}^{16} ThirdParty_{int\_j}} * 100\% \qquad (12)$$

The results are shown in Figure 68b in units of percent, which represents the percentage of each summed total that the metric value at each heading represents.

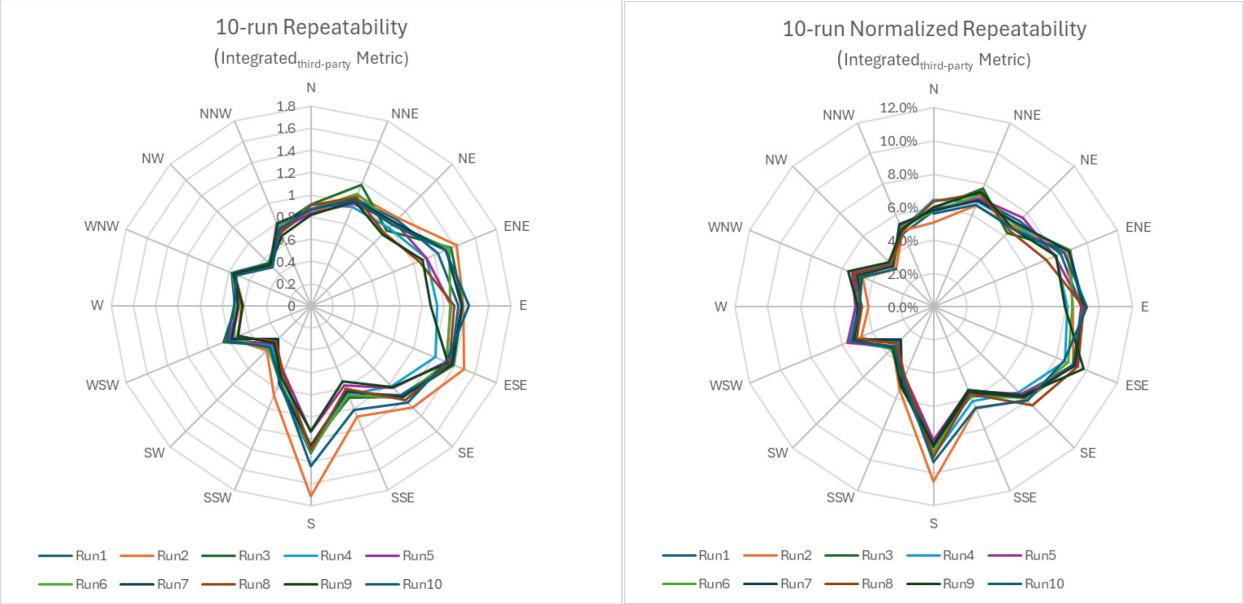

Figure 68 – Repeatability over 10 runs of 1) Integrated Third-Party metric, b) Normalized Integrated Third-Party metric for baseline descent profile using special event setting

Chapter 7

Conclusions and Future Directions

7.1. Conclusions

Carefully planning the integration of Urban Air Mobility into our communities is of critical importance because the technological readiness level of Electric Vertical Takeoff and Landing vehicles is advanced enough that some are expected to be certified for flight operations in Canada in the near future. The purpose of this research was to create a prototype Digital Twin that can use real-world data to create the virtual representation of the built environment as well as capable of using live population data, represent the data in a useful manner, evaluate some third-party impacts and demonstrate that EVTOL operations can be optimized to minimize third-party impacts. All of these were accomplished with the creation of a Digital Twin with 2 distinct architectures.

The first architecture is that of an interactive digital twin prototype where live data on third-parties is used and a resulting (static) agent-based representation is created. After the call to the live traffic API, an integrated metric for third-party impacts is calculated for 16 approach headings. The approach heading with the minimum integrated third-party metric value is identified. When commanded by the user, an EVTOL agent is created and performs the recommended flight path, for demonstration purposes. The user interface supports visualization of the virtual world in 2D and 3D, it supports visualizations of vehicle and pedestrians as well as their distributions, it supports manual control of pedestrian and vehicle agents, and finally it supports visualization of the output integrated third-party metric. The Digital Twin Prototype architecture successfully uses live Traffic API data from Here.com and represents the data as an equivalent agent-based model, on a refresh cycle of once per minute.

The second architecture was created to support the case study performed on an area in Montreal. A vertiport was virtually placed on the roof of the Concordia University Hall building. Two test settings were manually created with a different quasi-steady-state patterns of pedestrians and vehicles: a normal day scenario with a mostly even distribution of pedestrians and vehicles, and a special event scenario representing an outdoor event along St Catherines st. Vehicle and pedestrian agents were created using a stochastic function where the start positions, and end targets were controlled statistically via a fitness proportionate selection algorithm that can be customized to create the desired agent distributions. This method allowed realistic variation between case study runs, but with repeatable overall agent distributions. The method works very well for pedestrian agents because they generally do not interact with each other in a detrimental way, but it does not work as well for vehicle agents. Vehicle agents follow the rules of the road and are therefore subject to traffic light timings. The vehicles realistically group together at traffic lights, but, producing an expected distribution of vehicles requires a significant amount of manual tuning for light timings, which is outside of the scope of this thesis.

The results of the case study were collected, analyzed, and presented. For each of the two settings presented in the case study, 64 simultaneous approach flights were flown(16 evenly-spaced headings for each of the 4 different descent paths), and third-party metrics were calculated for ground crash safety, building crash safety, falling object safety, building privacy, and an integrated metric that combines the other 4 metrics in a weighted sum. The results demonstrate that the metrics can be used to suggest an optimal flight path that minimizes third-party impacts.

Previous studies used complex flight paths and cartesian coordinates to optimize UAV flights that minimize third-party impacts, but it was found that using polar coordinates (heading, altitude, distance) is an intuitive method for optimizing EVTOL flight paths and is aligned with traditional aviation. The use of radar charts is an easy to generate and intuitive method to visualize the metrics for a case study such as this where the flight approaches follow a consistent heading. Vertiports are the most discussed and studied infrastructure for EVTOL usage. Radar charts will prove to be a useful tool for vertiport operation planning and air traffic management.

The results of the case study highlight how important policy decisions are. When a regulator such as the FAA produces documents that guide the design of systems such as vertiports, details given may create non-optimal outcomes. Guidance in the FAA engineering brief 105A such as a suggested approach heading aligned with the prevailing winds allows for the appropriate additions of fixed visual markings, lighting, and electronic systems that help navigate vehicles safely in for landing but also can create outcomes that are less safe to third-parties. It was found that in the study area, the suggested flight paths were not optimal and resulted in over 2x more third-party impact compared to the optimized approach heading.

The case study architecture was also used to validate various aspects of the results and methodology. The first such validation is the assumption that third-party metrics could be used to determine an optimal flight path that minimizes the metrics was successful. The case study was designed to be a short, but critical, portion of the EVTOL flight: the final roughly 20 seconds of the approach during landing. In comparison to the metrics estimated prior to the EVTOL approach (flight time=0), the actual third-party metrics calculated as the EVTOLs flew their approach flight, the results were different (as expected) but not enough to significantly change which flight path heading was optimal. It was shown that using a representative dynamically changing agent-based third-party representation of vehicles and pedestrians can still predict a consistent optimal heading.

Real-time third-party data is pre-processed by the data providers to remove identifiable information. They use methods such as converting the raw data into population density or average traffic speed. This has the adverse effect of not being optimal for modelling using an agent-based simulation but also provides an opportunity for future research. For this thesis, since access to the raw data is not possible, there is no way to validate that the agent-based model appropriately matches the data or measures its accuracy. Despite this, the underlying methodology of converting to an agent-based model has potential, such as ease of simulating alternate scenarios.

The dynamically changing agent-based third-party representation was further validated for metric sensitivity and repeatability. Similar third-party metric values were obtained between multiple runs of the case study. This shows that the agent-based representation of the test settings remain valid despite the stochastic differences in the agent populations and validates that the sensitivity of the metrics is aligned with the variability of the agent positions. This shows that the third-party metrics are sensitive enough to be applied usefully for flight planning, but not too sensitive to result in large variations for different representations of the same input data.

Digital Twins will no doubt play an important role in maintaining safe and equitable Urban Air Mobility operations above our future communities.

7.2. Future Directions

The objectives of this research project were accomplished successfully but highlighted many future opportunities to expand this research in new directions. The Digital Twin Prototype developed for this thesis is easy to use, and easy to extend to support new research paths due in part to the use of Anylogic as the simulation and modeling engine which has proven to be an amazing and flexible tool for this type of research.

Access to real-time third-party data is essential to studies such as this. Data providers align their API offerings with what their customers request (and pay for) so there is an opportunity to determine the form of data that will best fit the needs of a project such as this one. Future research could explore an optimal data processing method that data providers could use to process the raw third-party position data that enables the creation of a verifiable agent-based representation and includes enough metadata to accurately represent people located in buildings or using various forms of transportation.

The use of machine learning to build models for third-party positions based on historical data could be created and combined with live data to fill in the data gaps that exist in the live data APIs in terms of data completeness or coverage in the areas of interest but also in terms of data update frequency. Furthermore, agentic AI techniques could be incorporated to build more robust models for third-parties as well as for EVTOL agent modeling and air traffic management.

Data sources and visualization technologies are always improving. As of the writing of this thesis, the satellite view of Montreal available in Google Maps is a 3D high-resolution scene created via photogrammetry. The visible details are impressive where not only buildings and streets are shown, but foliage and other cultural details are provided in high resolution. With the advances in machine learning for object identification, creation of the virtual world using technology such as this is exciting as it could greatly improve the third-party metrics for safety (via modelling detailed sheltering effects), and privacy by improved sight-line analysis.

With access to computing power becoming continually cheaper, and rendering technologies such as Unreal Engine continually improving, it is feasible to greatly scale up the Digital Twin geographic area. The case study area was meant to simulate EVTOL vehicle final approaches, so single heading approaches were viable for this analysis, but the entire flight path from vertiport to vertiport could be accomplished using techniques such as applying a min-cost A* algorithm to optimize longer flights in a way that makes sense for EVTOL flights, which have constraints of feasibility as well are vehicle capabilities and maneuverability. Further, a larger geographic area also presents the opportunity to explore more complex flight paths such as road-following, area avoidance and reactions to events that require rerouting.

References

[1] J. A. Stoop and J. P. Kahan, “Flying is the safest way to travel: How aviation was a pioneer in independent accident investigation,” *Eur. J. Transp. Infrastruct. Res.*, vol. 5, no. 2, Art. no. 2, June 2005, doi: 10.18757/ejtir.2005.5.2.4392.

[2] NASA, “Advanced Air Mobility Community Integration Considerations Playbook May 2023,” May 2023.

[3] “Vision and Mission to 2025.” Accessed: Aug. 22, 2025. [Online]. Available: https://www.icao.int/about-icao/vision-and-mission

[4] M. Hirst, *The Air Transport System*. 2008. Accessed: Nov. 18, 2025. [Online]. Available: https://www.sciencedirect.com/topics/engineering/civil-aviation-authorities

[5] *U-space ConOps*. [Online]. Available: https://corus-xuam.eu/wp-content/uploads/2022/11/CORUS-XUAM-D4.1-delivered_3.10.pdf

[6] FAA, “Urban Air Mobility (UAM) Concept of Operations 2.0.” Federal Aviation Administration, Apr. 2023. [Online]. Available: https://www.faa.gov/sites/faa.gov/files/Urban%20Air%20Mobility%20%28UAM%29%20Concept%20of%20Operations%202.0_0.pdf

[7] *Advanced Aerial Mobility: A National Blueprint*. Washington, D.C.: National Academies Press, 2020. doi: 10.17226/25646.

[8] “AC 21.17-3 - Type Certification of Very Light Airplanes Under FAR 21.17(b).” Accessed: Aug. 27, 2023. [Online]. Available: https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentid/22060

[9] “Performance Based Operations Aviation Rulemaking Committee (PARC) | Federal Aviation Administration.” Accessed: July 09, 2023. [Online]. Available: https://www.faa.gov/about/office_org/headquarters_offices/avs/offices/afx/afs/afs400/parc

[10] Federal Aviation Administration, “Engineering Brief No. 105, Vertiport Design.” Sept. 21, 2022. [Online]. Available: https://www.faa.gov/sites/faa.gov/files/eb-105-vertiports.pdf

[11] W. Johnson and C. Silva, “NASA Concept Vehicles and the Engineering of Advanced Air Mobility Aircraft,” *Aeronaut. J.*, vol. 126, no. 1295, Jan. 2022, Accessed: July 10, 2023. [Online]. Available: https://ntrs.nasa.gov/citations/20210026170

[12] X. Ren and C. Cheng, “Model of Third-Party Risk Index for Unmanned Aerial Vehicle Delivery in Urban Environment,” *Sustainability*, vol. 12, no. 20, Art. no. 20, Jan. 2020, doi: 10.3390/su12208318.

[13] M. Singh, E. Fuenmayor, E. Hinchy, Y. Qiao, N. Murray, and D. Devine, “Digital Twin: Origin to Future,” *Appl. Syst. Innov.*, vol. 4, no. 2, p. 36, May 2021, doi: 10.3390/asi4020036.

[14] S. Hu, Z. Huang, K. Wang, H. Lin, and M. Pei, “Modeling the adoption of urban air mobility based on technology acceptance and risk perception theories: A case study on flying cars,” *Multimodal Transp.*, vol. 4, no. 2, p. 100200, June 2025, doi: 10.1016/j.multra.2025.100200.

[15] NASA, *Advanced Air Mobility Community Integration Considerations Playbook*, May 2023.

[16] “How Google uses location information – Privacy & Terms – Google.” Accessed: Aug. 22, 2025. [Online]. Available: https://policies.google.com/technologies/location-data?hl=en-US

[17] “Telus Insights Location API,” Telus Insights Location API. Accessed: Aug. 22, 2025. [Online]. Available: https://docs.insights.telus.com

[18] “Facebook Statistics in Canada,” Made in CA. Accessed: Aug. 22, 2025. [Online]. Available: https://madeinca.ca/facebook-statistics-canada/

[19] L. S. Branch, “Consolidated federal laws of Canada, Canadian Aviation Regulations.” Accessed: Nov. 19, 2025. [Online]. Available: https://laws-lois.justice.gc.ca/eng/regulations/sor-96-433/page-61.html

[20] Federal Aviation Administration, “Engineering Brief No. 105.A, Vertiport Design, Supplemental Guidance to Advisory Circular 150/5390-2D, Heliport Design.” Dec. 27, 2024. [Online]. Available: https://www.faa.gov/airports/engineering/engineering_briefs/eb_105a_vertiports
[21] “Next-Generation Cities: An Encyclopedia | Next-Generation Cities Institute - Concordia University.” Accessed: Nov. 18, 2025. [Online]. Available: https://www.concordia.ca/research/cities-institute/initiatives/encyclopedia.html
[22] Oxford English Dictionary, “equity, n. meanings, etymology and more | Oxford English Dictionary.” Accessed: Dec. 17, 2023. [Online]. Available: https://www.oed.com/dictionary/equity_n
[23] K. Manaugh, M. G. Badami, and A. M. El-Geneidy, “Integrating social equity into urban transportation planning: A critical evaluation of equity objectives and measures in transportation plans in North America,” *Transp. Policy*, vol. 37, pp. 167–176, Jan. 2015, doi: 10.1016/j.tranpol.2014.09.013.
[24] E. O. Lewis, D. MacKenzie, and J. Kaminsky, “Exploring equity: How equity norms have been applied implicitly and explicitly in transportation research and practice,” *Transp. Res. Interdiscip. Perspect.*, vol. 9, p. 100332, Mar. 2021, doi: 10.1016/j.trip.2021.100332.
[25] L. Robbins, *An Essay on the Nature and Significance of Economic Science.* 1932.
[26] T. Litman, “Evaluating Transportation Equity,” Feb. 2005, Accessed: Jan. 07, 2024. [Online]. Available: https://web.archive.org/web/20050226053030/http://vtpi.org/equity.pdf
[27] T. Litman, “Evaluating Transportation Equity: Guidance for Incorporating Distributional Impacts in Transport Planning,” *Inst. Transp. Eng. ITE J.*, vol. 92, no. 4, pp. 43–49, Apr. 2022.
[28] K. Cantilina, S. R. Daly, M. P. Reed, and R. C. Hampshire, “Approaches and Barriers to Addressing Equity in Transportation: Experiences of Transportation Practitioners,” *Transp. Res. Rec.*, vol. 2675, no. 10, pp. 972–985, Oct. 2021, doi: 10.1177/03611981211014533.
[29] L. Van Dort, A. Guthrie, Y. Fan, and G. Baas, “Advancing Transportation Equity: Research and Practice,” Center for Transportation Studies, University of Minnesota, Report, Feb. 2019. Accessed: Dec. 17, 2023. [Online]. Available: http://conservancy.umn.edu/handle/11299/204694
[30] N. Thomopoulos, S. Grant-Muller, and M. R. Tight, “Incorporating equity considerations in transport infrastructure evaluation: Current practice and a proposed methodology,” *Eval. Program Plann.*, vol. 32, no. 4, pp. 351–359, Nov. 2009, doi: 10.1016/j.evalprogplan.2009.06.013.
[31] B. van Wee and N. Mouter, “Chapter Five - Evaluating transport equity,” in *Advances in Transport Policy and Planning*, vol. 7, N. Mouter, Ed., in New Methods, Reflections and Application Domains in Transport Appraisal, vol. 7. , Academic Press, 2021, pp. 103–126. doi: 10.1016/bs.atpp.2020.08.002.
[32] L. Ceriani and P. Verme, “The origins of the Gini index: extracts from Variabilità e Mutabilità (1912) by Corrado Gini,” *J. Econ. Inequal.*, vol. 10, no. 3, pp. 421–443, Sept. 2012, doi: 10.1007/s10888-011-9188-x.
[33] A. Delbosc and G. Currie, “Using Lorenz curves to assess public transport equity,” *J. Transp. Geogr.*, vol. 19, no. 6, pp. 1252–1259, Nov. 2011, doi: 10.1016/j.jtrangeo.2011.02.008.
[34] S. Jang, Y. An, C. Yi, and S. Lee, “Assessing the spatial equity of Seoul’s public transportation using the Gini coefficient based on its accessibility,” *Int. J. Urban Sci.*, vol. 21, no. 1, pp. 91–107, Jan. 2017, doi: 10.1080/12265934.2016.1235487.
[35] S. A. Hosein Mortazavi and M. Akbarzadeh, “A Framework for Measuring the Spatial Equity in the Distribution of Public Transportation Benefits,” *J. Public Transp.*, vol. 20, no. 1, pp. 44–62, Jan. 2017, doi: 10.5038/2375-0901.20.1.3.
[36] J. P. Pritchard, M. Stępniak, and K. T. Geurs, “5 - Equity analysis of dynamic bike-and-ride accessibility in the Netherlands,” in *Measuring Transport Equity*, K. Lucas, K. Martens, F. Di Ciommo, and A. Dupont-Kieffer, Eds., Elsevier, 2019, pp. 73–83. doi: 10.1016/B978-0-12-814818-1.00005-6.
[37] Y. Guo, Z. Chen, A. Stuart, X. Li, and Y. Zhang, “A systematic overview of transportation equity in terms of accessibility, traffic emissions, and safety outcomes: From conventional to emerging

technologies," *Transp. Res. Interdiscip. Perspect.*, vol. 4, p. 100091, Mar. 2020, doi: 10.1016/j.trip.2020.100091.
[38] Y. Guo, Z. Chen, A. Stuart, X. Li, and Y. Zhang, "A systematic overview of transportation equity in terms of accessibility, traffic emissions, and safety outcomes: From conventional to emerging technologies," *Transp. Res. Interdiscip. Perspect.*, vol. 4, p. 100091, Mar. 2020, doi: 10.1016/j.trip.2020.100091.
[39] D. S. Johnson, J. K. Lenstra, and A. H. G. R. Kan, "The complexity of the network design problem," *Networks*, vol. 8, no. 4, pp. 279–285, Dec. 1978, doi: 10.1002/net.3230080402.
[40] O. J. Ibarra-Rojas, F. Delgado, R. Giesen, and J. C. Muñoz, "Planning, operation, and control of bus transport systems: A literature review," *Transp. Res. Part B Methodol.*, vol. 77, pp. 38–75, July 2015, doi: 10.1016/j.trb.2015.03.002.
[41] R. Camporeale, L. Caggiani, and M. Ottomanelli, "Modeling horizontal and vertical equity in the public transport design problem: A case study," *Transp. Res. Part Policy Pract.*, vol. 125, pp. 184–206, July 2019, doi: 10.1016/j.tra.2018.04.006.
[42] H. Behbahani, S. Nazari, M. Jafari Kang, and T. Litman, "A conceptual framework to formulate transportation network design problem considering social equity criteria," *Transp. Res. Part Policy Pract.*, vol. 125, pp. 171–183, July 2019, doi: 10.1016/j.tra.2018.04.005.
[43] M. Kim, S.-Y. Kho, and D.-K. Kim, "A Transit Route Network Design Problem Considering Equity," *Sustainability*, vol. 11, no. 13, Art. no. 13, Jan. 2019, doi: 10.3390/su11133527.
[44] Q. Molloy, N. Garrick, and C. Atkinson-Palombo, "A New Approach to Understanding the Impact of Automobile Ownership on Transportation Equity," *Transp. Res. Rec.*, 2023, doi: 10.1177/03611981231174444.
[45] B. J. M. Ale and M. Piers, "The assessment and management of third party risk around a major airport," *J. Hazard. Mater.*, vol. 71, no. 1, pp. 1–16, Jan. 2000, doi: 10.1016/S0304-3894(99)00069-2.
[46] E. Ancel, T. Helsel, and C. M. Heinich, "Ground Risk Assessment Service Provider (GRASP) Development Effort as a Supplemental Data Service Provider (SDSP) for Urban Unmanned Aircraft System (UAS) Operations," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, Sept. 2019, pp. 1–8. doi: 10.1109/DASC43569.2019.9081659.
[47] E. Ancel, F. M. Capristan, J. V. Foster, and R. C. Condotta, "In-Time Non-Participant Casualty Risk Assessment to Support Onboard Decision Making for Autonomous Unmanned Aircraft," in *AIAA Aviation Forum 2019, no. AIAA 2019-3053*, June 2019. Accessed: Aug. 21, 2025. [Online]. Available: https://ntrs.nasa.gov/api/citations/20200002643/downloads/20200002643.pdf
[48] V. Celdran Martinez, H.-S. Shin, and A. Tsourdos, "Risk Assessment for sUAS in Urban Environments: A Comprehensive Analysis, Modelling and Validation for Safe Operations," in *AIAA SCITECH 2024 Forum*, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2024. doi: 10.2514/6.2024-0232.
[49] X. Ren and C. Cheng, "Model of Third-Party Risk Index for Unmanned Aerial Vehicle Delivery in Urban Environment," *Sustainability*, vol. 12, no. 20, Art. no. 20, Jan. 2020, doi: 10.3390/su12208318.
[50] X. Ren and C. Cheng, "Construction and application of third-party risk model for unmanned aerial vehicle operation in urban environment," *China Saf. Sci. J.*, vol. 31, no. 9, pp. 15–20, 2021, doi: 10.16265/j.cnki.issn1003-3033.2021.09.003.
[51] S. Choi, B. Kim, and H. Kim, "Third-Party Risk Assessment on the Ground for Urban Air Mobility Operations: A Case Study of Seoul Metropolitan City," *Int. J. Aeronaut. Space Sci.*, Sept. 2024, doi: 10.1007/s42405-024-00827-0.
[52] H. Tang, Q. Zhu, B. Qin, R. Song, and Z. Li, "UAV path planning based on third-party risk modeling," *Sci. Rep.*, vol. 13, no. 1, p. 22259, Dec. 2023, doi: 10.1038/s41598-023-49396-4.
[53] H. Tang, Q. Zhu, B. Qin, R. Song, and Z. Li, "UAV path planning based on third-party risk modeling," *Sci. Rep.*, vol. 13, no. 1, p. 22259, Dec. 2023, doi: 10.1038/s41598-023-49396-4.

[54] V. Celdran Martinez, H.-S. Shin, and A. Tsourdos, “Risk Assessment for sUAS in Urban Environments: A Comprehensive Analysis, Modelling and Validation for Safe Operations,” in *AIAA SCITECH 2024 Forum*, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2024. doi: 10.2514/6.2024-0232.
[55] W. Johnson and C. Silva, “NASA Concept Vehicles and the Engineering of Advanced Air Mobility Aircraft,” *Aeronaut. J.*, vol. 126, no. 1295, Jan. 2022, Accessed: July 10, 2023. [Online]. Available: https://ntrs.nasa.gov/citations/20210026170
[56] “Seoul Metropolitan City (2024) Seoul Open Data Plaza.” Accessed: Aug. 21, 2025. [Online]. Available: https://data.seoul.go.kr
[57] R. Melnyk, D. Schrage, V. Volovoi, and H. Jimenez, “A third-party casualty risk model for unmanned aircraft system operations,” *Reliab. Eng. Syst. Saf.*, vol. 124, pp. 105–116, Apr. 2014, doi: 10.1016/j.ress.2013.11.016.
[58] B. J. M. Ale and M. Piers, “The assessment and management of third party risk around a major airport,” *J. Hazard. Mater.*, vol. 71, no. 1, pp. 1–16, Jan. 2000, doi: 10.1016/S0304-3894(99)00069-2.
[59] S. H. S. B. Cardoso, M. V. R. de Oliveira, and J. R. S. Godoy, “eVTOL Certification in FAA and EASA Performance-Based Regulation Environments: A Bird Strike Study-Case,” *J. Aerosp. Technol. Manag.*, vol. 14, p. e2122, Nov. 2022, doi: 10.1590/jatm.v14.1271.
[60] M. H. Che Man and K. H. Low, “Damage Severity Prediction of Helicopter Windshields Caused by a Collision with a Small Unmanned Aerial Vehicle (sUAV),” presented at the AIAA Aviation and Aeronautics Forum and Exposition, AIAA AVIATION Forum 2021, 2021. doi: 10.2514/6.2021-3001.
[61] R. McKillip, A. Kaufman, and T. Quackenbush, “eVTOL accretion modeling for supporting algorithmic icing detection,” Oct. 2020. Accessed: Mar. 12, 2025. [Online]. Available: https://www-scopus-com.lib-ezproxy.concordia.ca/record/display.uri?eid=2-s2.0-85096893641&origin=resultslist&sort=plf-f&src=s&sot=b&sdt=b&s=TITLE-ABS-KEY%28evtol+AND+deicing%29&sessionSearchId=fce07081748248108b5effea7f178e6a
[62] A. Fitwi, Y. Chen, and S. Zhu, *No Peeking through My Windows: Conserving Privacy in Personal Drones*. 2019. doi: 10.48550/arXiv.1908.09935.
[63] T. Krstić Simić, E. Ganić, B. Mirković, M. Baena, I. LeGriffon, and C. Barrado, “U-Space Social and Environmental Performance Indicators,” *Drones*, vol. 8, no. 10, 2024, doi: 10.3390/drones8100580.
[64] M. Grieves and J. Vickers, “Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems,” in *Transdisciplinary Perspectives on Complex Systems*, Springer, Cham, 2017, pp. 85–113. doi: 10.1007/978-3-319-38756-7_4.
[65] M. Singh, E. Fuenmayor, E. Hinchy, Y. Qiao, N. Murray, and D. Devine, “Digital Twin: Origin to Future,” *Appl. Syst. Innov.*, vol. 4, no. 2, p. 36, May 2021, doi: 10.3390/asi4020036.
[66] D. Gelernter, *Mirror Worlds: or the Day Software Puts the Universe in a Shoebox...How It Will Happen and What It Will Mean*. Oxford, New York: Oxford University Press, 1993.
[67] X. Wang, N. Chevalot, G. Monnier, S. Ausejo, Á. Suescun, and J. Celigüeta, “Validation of a Model-based Motion Reconstruction Method Developed in the REALMAN Project,” presented at the 2005 Digital Human Modeling for Design and Engineering Symposium, June 2005, pp. 2005-01–2743. doi: 10.4271/2005-01-2743.
[68] J. Puig and J. Duran, “Digital Twins,” in *Proceedings of the 4th International Multi-Conference on Society, Cybernetics, and Informatics*, June 2010.
[69] E. J. Tuegel, A. R. Ingraffea, T. G. Eason, and S. M. Spottswood, “Reengineering Aircraft Structural Life Prediction Using a Digital Twin,” *Int. J. Aerosp. Eng.*, vol. 2011, no. 1, p. 154798, 2011, doi: 10.1155/2011/154798.
[70] L. Wright and S. Davidson, “How to tell the difference between a model and a digital twin,” *Adv. Model. Simul. Eng. Sci.*, vol. 7, no. 1, p. 13, Dec. 2020, doi: 10.1186/s40323-020-00147-4.

[71] P. Raj and C. Surianarayanan, “Chapter Twelve - Digital twin: The industry use cases,” in *Advances in Computers*, vol. 117, 1 vols., P. Raj and P. Evangeline, Eds., in The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases, vol. 117. , Elsevier, 2020, pp. 285–320. doi: 10.1016/bs.adcom.2019.09.006.
[72] M. Grieves and J. Vickers, “Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems,” in *Transdisciplinary Perspectives on Complex Systems*, Springer, Cham, 2017, pp. 85–113. doi: 10.1007/978-3-319-38756-7_4.
[73] T. John, K. Nath, and K. Guravaiah, “Exploring the Adoption and Innovation of Digital Twins in Healthcare,” *Procedia Comput. Sci.*, vol. 257, pp. 93–102, Jan. 2025, doi: 10.1016/j.procs.2025.03.015.
[74] A. A. Garanin, O. Yu. Aidumova, and A. V. Kontsevaya, “Clinical aspects of digital twins in medicine: a systematic review,” *Eur. Phys. J. Spec. Top.*, Feb. 2025, doi: 10.1140/epjs/s11734-025-01518-x.
[75] V. Karkaria, Y.-K. Tsai, Y.-P. Chen, and W. Chen, “An optimization-centric review on integrating artificial intelligence and digital twin technologies in manufacturing,” *Eng. Optim.*, vol. 57, no. 1, pp. 161–207, Jan. 2025, doi: 10.1080/0305215X.2024.2434201.
[76] E.-A. Roman, A.-S. Stere, E. Roșca, A.-V. Radu, D. Codroiu, and I. Anamaria, “State of the Art of Digital Twins in Improving Supply Chain Resilience,” *Logistics*, vol. 9, no. 1, Art. no. 1, Mar. 2025, doi: 10.3390/logistics9010022.
[77] W. A. Ali, M. Roccotelli, and M. P. Fanti, “Digital Twin in Intelligent Transportation Systems: a Review,” in *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, May 2022, pp. 576–581. doi: 10.1109/CoDIT55151.2022.9804017.
[78] M. Xiong and H. Wang, “Digital twin applications in aviation industry: A review,” *Int. J. Adv. Manuf. Technol.*, vol. 121, no. 9, pp. 5677–5692, Aug. 2022, doi: 10.1007/s00170-022-09717-9.
[79] T. Zhang, D. Grzelak, W. Zhao, M. A. Islam, H. Fricke, and U. Aßmann, “A Review on the Construction, Modeling, and Consistency of Digital Twins for Advanced Air Mobility Applications,” *Drones*, vol. 9, no. 6, Art. no. 6, June 2025, doi: 10.3390/drones9060394.
[80] E. C. Pinto Neto, D. M. Baum, J. R. de Almeida, J. B. Camargo, and P. S. Cugnasca, “A Trajectory Evaluation Platform for Urban Air Mobility (UAM),” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9136–9145, July 2022, doi: 10.1109/TITS.2021.3091411.
[81] T. Canada, “Canadian Aviation Regulations (SOR/96-433),” AARBH 14882767. Accessed: Aug. 22, 2025. [Online]. Available: https://tc.canada.ca/en/corporate-services/acts-regulations/list-regulations/canadian-aviation-regulations-sor-96-433
[82] “Certification Specifications (CSs) / Detailed Specifications (DSs) | EASA.” Accessed: Aug. 22, 2025. [Online]. Available: https://www.easa.europa.eu/en/document-library/certification-specifications
[83] *REQUISITOS GERAIS DE OPERAÇÃO PARA AERONAVES CIVIS – EMENDA 01*, 91, May 08, 2019. Accessed: Aug. 22, 2025. [Online]. Available: https://www.gov.br/anac/pt-br/acesso-a-informacao/participacao-social/consultas-publicas/audiencias/2019/15/ap152019quadrocomparativorbac91.pdf?utm_source=chatgpt.com
[84] A. P. Cohen, S. A. Shaheen, and E. M. Farrar, “Urban Air Mobility: History, Ecosystem, Market Potential, and Challenges,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 6074–6087, 2021, doi: 10.1109/TITS.2021.3082767.
[85] “https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_150_5390_2D_Heliports.pdf.” Accessed: Aug. 22, 2025. [Online]. Available: https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_150_5390_2D_Heliports.pdf
[86] S. H. S. B. Cardoso, M. V. R. de Oliveira, and J. R. S. Godoy, “eVTOL Certification in FAA and EASA Performance-Based Regulation Environments: A Bird Strike Study-Case,” *J. Aerosp. Technol. Manag.*, vol. 14, p. e2122, Nov. 2022, doi: 10.1590/jatm.v14.1271.

[87] D. Xue, X. M. Chen, and S. Yu, "Sustainable aviation for a greener future," *Commun. Earth Environ.*, vol. 6, no. 1, p. 233, Mar. 2025, doi: 10.1038/s43247-025-02222-3.
[88] S. Lee and N. Cho, "Optimal Location of Urban Air Mobility (UAM) Vertiport Using a Three-Stage Geospatial Analysis Framework," *Future Transp.*, vol. 5, no. 2, p. 58, June 2025, doi: 10.3390/futuretransp5020058.
[89] M. Chae, S. H. Kim, M. Kim, H.-T. Park, and S. H. Kim, "Potential market based policy considerations for urban air mobility," *J. Air Transp. Manag.*, vol. 119, p. 102654, Aug. 2024, doi: 10.1016/j.jairtraman.2024.102654.
[90] "AnyLogic: Simulation Modeling Software Tools & Solutions for Business." Accessed: Mar. 26, 2025. [Online]. Available: https://www.anylogic.com/
[91] "Welcome - Site web des données ouvertes de la Ville de Montréal." Accessed: June 23, 2025. [Online]. Available: https://donnees.montreal.ca/en/?
[92] "Bâtiments 3D 2016 (Maquette LOD2 avec textures) - Site web des données ouvertes de la Ville de Montréal." Accessed: Aug. 22, 2025. [Online]. Available: https://donnees.montreal.ca/dataset/batiment-3d-2016-maquette-citygml-lod2-avec-textures2
[93] "EPSG:2950 NAD83(CSRS) / MTM zone 8 -- Spatial Reference." Accessed: June 23, 2025. [Online]. Available: https://spatialreference.org/ref/epsg/2950/
[94] "Modèle numérique de terrain - Site web des données ouvertes de la Ville de Montréal." Accessed: Aug. 22, 2025. [Online]. Available: https://donnees.montreal.ca/dataset/modele-numerique-de-terrain-mnt
[95] I. A. I. webmaster, "KIT - IAI - Downloads - FZKViewer." Accessed: Nov. 19, 2025. [Online]. Available: https://www.iai.kit.edu/english/1648.php
[96] "Cartographie de base - Site web des données ouvertes de la Ville de Montréal." Accessed: Aug. 22, 2025. [Online]. Available: https://donnees.montreal.ca/dataset/cartographie-de-base
[97] "Géobase - réseau routier - Site web des données ouvertes de la Ville de Montréal." Accessed: Aug. 22, 2025. [Online]. Available: https://donnees.montreal.ca/fr/dataset/geobase
[98] "LiDAR aérien 2015 - Site web des données ouvertes de la Ville de Montréal." Accessed: Aug. 22, 2025. [Online]. Available: https://donnees.montreal.ca/fr/dataset/lidar-aerien-2015
[99] "Arbres publics sur le territoire de la Ville - Site web des données ouvertes de la Ville de Montréal." Accessed: Aug. 22, 2025. [Online]. Available: https://donnees.montreal.ca/fr/dataset/arbres
[100] "Arbres publics de Montréal." Accessed: Aug. 22, 2025. [Online]. Available: https://www.quebio.ca/fr/arbresmtl
[101] K. J. Niklas, "Size-dependent Allometry of Tree Height, Diameter and Trunk-taper," *Ann. Bot.*, vol. 75, no. 3, pp. 217–227, Mar. 1995, doi: 10.1006/anbo.1995.1015.
[102] "OpenStreetMap," OpenStreetMap. Accessed: Aug. 22, 2025. [Online]. Available: https://www.openstreetmap.org/
[103] "Open Data Commons Open Database License (ODbL) — Open Data Commons: legal tools for open data." Accessed: Aug. 22, 2025. [Online]. Available: https://opendatacommons.org/licenses/odbl/
[104] "OSM XML - OpenStreetMap Wiki." Accessed: Aug. 22, 2025. [Online]. Available: https://wiki.openstreetmap.org/wiki/OSM_XML
[105] "HERE Technologies | The world's #1 location platform." Accessed: Aug. 22, 2025. [Online]. Available: https://www.here.com
[106] "Introduction." Accessed: Aug. 22, 2025. [Online]. Available: https://www.here.com/docs/bundle/traffic-api-developer-guide-v6/page/topics/what-is.html
[107] "Google Maps Platform," Google for Developers. Accessed: Aug. 22, 2025. [Online]. Available: https://developers.google.com/maps
[108] A. Divasson-J., A. M. Macarulla, J. I. Garcia, and C. E. Borges, "Agent-based modeling in urban human mobility: A systematic review," *Cities*, vol. 158, p. 105697, Mar. 2025, doi: 10.1016/j.cities.2024.105697.
[109] *Road Vehicle Load and Size Limits Guide*.

[110] Federal Aviation Administration, “Engineering Brief No. 105, Vertiport Design.” Sept. 21, 2022. [Online]. Available: https://www.faa.gov/sites/faa.gov/files/eb-105-vertiports.pdf
[111] S. Choi, B. Kim, and H. Kim, “Third-Party Risk Assessment on the Ground for Urban Air Mobility Operations: A Case Study of Seoul Metropolitan City,” *Int. J. Aeronaut. Space Sci.*, Sept. 2024, doi: 10.1007/s42405-024-00827-0.
[112] S. D. Brady and R. Hillestad, “Modeling the External Risks of Airports for Policy Analysis,” RAND Corporation, Jan. 1995. Accessed: Apr. 08, 2025. [Online]. Available: https://www.rand.org/pubs/monograph_reports/MR605.html
[113] J. Caballero-Peña, G. Osma-Pinto, J. M. Rey, S. Nagarsheth, N. Henao, and K. Agbossou, “Analysis of the building occupancy estimation and prediction process: A systematic review,” *Energy Build.*, vol. 313, p. 114230, June 2024, doi: 10.1016/j.enbuild.2024.114230.
[114] NCR, *National Building Code of Canada*, 2015.
[115] “Joby Aviation S4 2.0 (pre-production prototype).” Accessed: Nov. 18, 2025. [Online]. Available: https://evtol.news/joby-s4
[116] Canada Aviation and Space Museum, “Canadair CL-84-1 Dynavert,” Canada Aviation and Space Museum. Accessed: June 27, 2025. [Online]. Available: https://ingenium.ca/aviation/en/collection-highlight/canadair-cl-84-1-dynavert/
[117] “Archer Midnight: Archer Aviation Midnight eVTOL Aircraft Overview,” Advanced Air Mobility Intl. Accessed: Nov. 18, 2025. [Online]. Available: https://www.aaminternational.com/projects/archer-midnight/
[118] “Diamond Aircraft VoloCity (prototype).” Accessed: Nov. 18, 2025. [Online]. Available: https://evtol.news/volocopter-volocity
[119] “Air Taxi by Wisk,” Boeing Future of Flight. Accessed: Nov. 18, 2025. [Online]. Available: https://www.boeingfutureofflight.com/wisk
[120] “EVE_Institutional+Presentation_November+24.pdf.” Accessed: Nov. 18, 2025. [Online]. Available: https://d1io3yog0oux5.cloudfront.net/_7865403a8ca662c6904cc525fc1fb5c8/eveairmobility/db/950/8930/pdf/EVE_Institutional+Presentation_November+24.pdf
[121] “EHang | UAM - Passenger Autonomous Aerial Vehicle (AAV).” Accessed: Nov. 18, 2025. [Online]. Available: https://www.ehang.com/ehangaav
[122] “Bell MV-75,” *Wikipedia*. Nov. 09, 2025. Accessed: Nov. 18, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Bell_MV-75&oldid=1321154673
[123] “V-22 Osprey.” Accessed: Nov. 18, 2025. [Online]. Available: https://www.boeing.com/content/theboeingcompany/us/en/defense/v-22-osprey
[124] “Aircraft.” Accessed: Nov. 18, 2025. [Online]. Available: https://beta.team/aircraft
[125] “bell-429-product-specifications.pdf.” Accessed: Nov. 18, 2025. [Online]. Available: https://www.bellflight.com/-/media/site-specific/bell-flight/documents/products/429/bell-429-product-specifications.pdf
[126] meteoblue.com, “Simulated historical climate & weather data for Montreal,” meteoblue. Accessed: Apr. 06, 2025. [Online]. Available: https://www.meteoblue.com/en/weather/historyclimate/climatemodelled/montreal_canada_6077243

Appendix A – Important Codebase

This appendix does not contain all java code used to create the Anylogic model, instead it contains the functions referenced in the text which represent the main functionality important to the research, namely the third-party metrics, the digital twin live data usage, and the virtual world building generation. It should be noted that function declarations are not included in the code below, because the user declares the function name, return type, and input variables inside the Anylogic graphical user interface, not along with the code. Following the Anylogic code, the python script developed to process the citygml files is included. In all cases, AI code assistance (Chatgpt) was used in the creation of the code.

```
//Function: void calculateFODZones()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)
foreignObjectDangerZones.clear();  // Clear previous data

double g = 9.81;  // Gravity (m/s²)
if(!useEVTOLArray)
{
                for (EVTOL evtol : eVTOLs) {
                    double x0 = evtol.getX();
                    double y0 = evtol.getY();
                    double h = evtol.getZ();  // eVTOL altitude
                    double theta = evtol.getRotation();  // eVTOL heading (radians)
                    double ve = evtol.getSpeed();  // eVTOL's speed (m/s)
                    double vb = 10;  // Blade speed (m/s)

                    // Compute time to fall
                    double t = Math.sqrt(2 * h / g);

                    // Compute major axis (A) and minor axis (B)
                    double A = vb * t;
                    double B = vb * t;

                    // Compute impact center
                    double X_impact = x0;
                    double Y_impact = y0;

                    // Save impact data (X_center, Y_center, Major Axis, Minor Axis, Orientation)
                    foreignObjectDangerZones.add(new Object[]{evtol.getId(), X_impact, Y_impact, A, B, theta});

                }
            }
  else
  {
            for (Object[] record : instantMetricsEVTOLArray)
            {
                double x0 = (double)record[1];
                double y0 = (double)record[2];
                double h = (double)record[3];  // eVTOL altitude
                double theta = 0.0;  // eVTOL heading (radians)
                double ve = (double)record[4];  // eVTOL's speed (m/s)
                double vb = 10;  // Blade speed (m/s)

                // Compute time to fall
                double t = Math.sqrt(2 * h / g);

                // Compute major axis (A) and minor axis (B)
                double A = vb * t;
                double B = vb * t;

                // Compute impact center
                double X_impact = x0;
                double Y_impact = y0;

                // Save impact data (X_center, Y_center, Major Axis, Minor Axis, Orientation)
                foreignObjectDangerZones.add(new Object[]{(int)record[0], X_impact, Y_impact, A, B, theta});
            }
  }
```

```
//Function: void drawFODZones()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

 // If ovals exist, update them instead of creating new ones
   if (!foreignObjectDangerZoneOvals.isEmpty()) {
      int i=0;
      for (Object[] record : foreignObjectDangerZones) {

         int evtolId = (int)record[0];
         double xImpact = (double)record[1];
         double yImpact = (double)record[2];
         double A = (double)record[3];  // Major axis
         double B = (double)record[4];  // Minor axis
         double theta = (double)record[5];  // Minor axis

         ShapeOval impactOval = foreignObjectDangerZoneOvals.get(i);  // Get the existing oval
         impactOval.setRadiusX(A);
         impactOval.setRadiusY(B);
         impactOval.setRotation(theta);
         impactOval.setY(yImpact);  // Centered Y yImpact - B / 2
         impactOval.setX(xImpact);  // Centered X xImpact - A / 2
         impactOval.setZ(0);  // Centered Y
         i++;
         impactOval.setVisible(checkbox1.isSelected());
      }
      return;  // Exit if ovals already exist
   }

   // If ovals don't exist, create them
      for (Object[] record : foreignObjectDangerZones) {

         int evtolId = (int)record[0];
         double xImpact = (double)record[1];
         double yImpact = (double)record[2];
         double A = (double)record[3];
         double B = (double)record[4];
         double theta = (double)record[5];

      // Create an oval at the calculated impact zone
      ShapeOval impactOval = new ShapeOval(
         SHAPE_DRAW_2D3D,
         true,
         xImpact,  // Center X
         yImpact,  // Center Y
         0,
         theta, //rotation
         BLACK,  // Border color
         new Color(255, 255, 0, 50),  // Red with transparency
         A,  // Width (Major Axis)
         B,  // Height (Minor Axis)
         1.0, //
         1.0,  // Line width
         LINE_STYLE_SOLID  // Line style
      );
                                 impactOval.setVisible(checkbox1.isSelected());
      presentation.add(impactOval);  // Add oval to the environment
      foreignObjectDangerZoneOvals.add(impactOval);  // Store for updates
      }
```

```
//Function: void updateAgentsFODZone()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

double time=useEVTOLArray? simulatedTime : time();

for (Pedestrian ped : pedestrians) {
   double agentX = ped.getX();
   double agentY = ped.getY();

   if (ped.FODZone.isEmpty())
   {
     for (Object[] record : foreignObjectDangerZones)
     {
      ped.FODZone.add(new Object[]{0, false});
     }
   }
   // Iterate through each crash zone
   int i=0;
   for (Object[] record : foreignObjectDangerZones)
   {
      int evtolId = (int) record[0];
      double xImpact = (double) record[1];
      double yImpact = (double) record[2];
      double radius = (double) record[3];

      // Calculate distance between agent and crash zone center
      double distance = Math.sqrt(Math.pow(agentX - xImpact, 2) + Math.pow(agentY - yImpact, 2));
                        boolean inZone=false;
      // Check if the agent is within the crash radius
      if (distance <= radius)
      {
            inZone=true;
      }
      if(inZone!=(boolean)ped.FODZone.get(i)[1])
      {

            AgentListFODZone.add(new Object[]{time,i,evtolId, ped.getId(), "ped", inZone, pedestrians.size(),radius});
      }
      ped.FODZone.set(i, new Object[]{evtolId, inZone});
      i++;
    }
}
```

```
//Function: void calculateFODMetric()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

// Map to store the final normalized occupancy per EVTOLIndex
double[] FODMetricArr = new double[64];
Map<Integer, Double> zoneTimePerEVTOL = new HashMap<>();

for (int evtolIndex = 0; evtolIndex < 64; evtolIndex++) {
    List<Object[]> filtered = new ArrayList<>();
    double earliestTime = Double.MAX_VALUE;
    double latestTime = Double.MIN_VALUE;
    double totalAgentsSum = 0.0;
    int totalAgentsCount = 0;

    // Filter and collect stats
    for (Object[] row : AgentListFODZone) {
        if ((int) row[1] == evtolIndex) {
            filtered.add(row);
            double time = (double) row[0];
            if (time < earliestTime) earliestTime = time;
            if (time > latestTime) latestTime = time;

            int totalAgents = (int) row[6];
            totalAgentsSum += totalAgents;
            totalAgentsCount++;
        }
    }

    if (filtered.isEmpty()) continue;

    double totalTime = latestTime - earliestTime;
    double avgTotalAgents = (totalAgentsCount > 0) ? totalAgentsSum / totalAgentsCount : 0.0;

    // Group by AgentID
    Map<String, List<Object[]>> agentEventsMap = new HashMap<>();
    for (Object[] row : filtered) {
        String agentID = String.valueOf(row[3]);
        agentEventsMap.computeIfAbsent(agentID, k -> new ArrayList<>()).add(row);
    }

    double numerator = 0.0;

    for (List<Object[]> agentEvents : agentEventsMap.values()) {
        agentEvents.sort(Comparator.comparingDouble(e -> (double) e[0]));

        for (int i = 0; i < agentEvents.size() - 1; i++) {
            boolean inZone = (boolean) agentEvents.get(i)[5];
            boolean nextInZone = (boolean) agentEvents.get(i + 1)[5];

            if (inZone && !nextInZone) {
                double t1 = (double) agentEvents.get(i)[0];
                double t2 = (double) agentEvents.get(i + 1)[0];
                double duration = t2 - t1;

                double r1 = (double) agentEvents.get(i)[7];
                double r2 = (double) agentEvents.get(i + 1)[7];
                double avgRadius = (r1 + r2) / 2.0;
                double area = Math.PI * avgRadius * avgRadius;

                if (area > 0 && duration > 0) {
                    numerator += duration / area;
                }
            }
        }
    }

    double denominator = avgTotalAgents * totalTime;
    double normalizedDensity = (denominator > 0) ? numerator / denominator : 0.0;
    zoneTimePerEVTOL.put(evtolIndex, normalizedDensity);
    FODMetrics.add(new Object[]{evtolIndex, normalizedDensity*1000000.0});

}
```

```
//Function: void calculateCrashZones()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

evtolCrashZones.clear();  // Clear previous data

   double g = 9.81;  // Gravity (m/s²)
            if(!useEVTOLArray)
            {
               for (EVTOL evtol : eVTOLs) {
                  double x0 = evtol.getX();
                  double y0 = evtol.getY();
                  double h = evtol.getZ();  // eVTOL altitude
                  double theta = evtol.getRotation();  // eVTOL heading (radians)
                  double ve = evtol.getSpeed();  // eVTOL's speed (m/s)

                  // Compute time to fall
                  double t = Math.sqrt(2 * h / g);
                                    double mtow = 5710.0;
                                    double adebris= 0.275*mtow;
                                    double impactRadius = sqrt(adebris/3.14159);

                  // Compute impact center
                  double X_impact = x0 + ve * Math.cos(theta) * t;
                  double Y_impact = y0 + ve * Math.sin(theta) * t;

                  // Save impact data (X_center, Y_center, Major Axis, Minor Axis, Orientation)
                  evtolCrashZones.add(new Object[]{evtol.getId(), X_impact, Y_impact, impactRadius, impactRadius, theta});
               }
            }
            else
   {
            for (Object[] record : instantMetricsEVTOLArray) {
                  double x0 = (double)record[1];
                  double y0 = (double)record[2];
                  double h = (double)record[3];  // eVTOL altitude
                  double theta = 0.0;  // eVTOL heading (radians)
                  double ve = (double)record[4];  // eVTOL's speed (m/s)

                  // Compute time to fall
                  double t = Math.sqrt(2 * h / g);
                                    double mtow = 5710.0;
                                    double adebris= 0.275*mtow;
                                    double impactRadius = sqrt(adebris/3.14159);

                  // Compute impact center
                  double X_impact = x0 + ve * Math.cos(theta) * t;
                  double Y_impact = y0 + ve * Math.sin(theta) * t;

                  // Save impact data (X_center, Y_center, Major Axis, Minor Axis, Orientation)
                  evtolCrashZones.add(new Object[]{(int)record[0], X_impact, Y_impact, impactRadius, impactRadius, theta});

            }
   }
```

```
//Function: void drawCrashZones()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)


// If ovals exist, update them instead of creating new ones
   if (!evtolCrashZoneOvals.isEmpty()) {
      int i=0;
      for (Object[] record : evtolCrashZones) {

         int evtolId = (int)record[0];
         double xImpact = (double)record[1];
         double yImpact = (double)record[2];
         double A = (double)record[3];  // Major axis
         double B = (double)record[4];  // Minor axis
         double theta = (double)record[5];  // Minor axis

         ShapeOval impactOval = evtolCrashZoneOvals.get(i);  // Get the existing oval
         impactOval.setRotation(theta);
         impactOval.setY(yImpact);  // Centered Y
         impactOval.setX(xImpact);  // Centered X
         impactOval.setZ(0);  // Centered Z
         i++;
         impactOval.setVisible(checkbox.isSelected());
      }
      return;  // Exit if ovals already exist
   }

   // If ovals don't exist, create them
      for (Object[] record : evtolCrashZones) {

         int evtolId = (int)record[0];
         double xImpact = (double)record[1];
         double yImpact = (double)record[2];
         double A = (double)record[3];
         double B = (double)record[4];
         double theta = (double)record[5];

      // Create an oval at the calculated impact zone
      ShapeOval impactOval = new ShapeOval(
         SHAPE_DRAW_2D3D,
         true,
         xImpact,  // Center X
         yImpact,  // Center Y
         0,
         theta, //rotation
         BLACK,  // Border color
         new Color(255, 0, 0, 50),  // Red with transparency
         A,  // Width (Major Axis)
         B,  // Height (Minor Axis)
         1.0,
         1.0,  // Line width
         LINE_STYLE_SOLID  // Line style
      );
      impactOval.setVisible(checkbox.isSelected());
      presentation.add(impactOval);  // Add oval to the environment
      evtolCrashZoneOvals.add(impactOval);  // Store for updates
      }
```

```
//Function: void updateAgentsCrashZone()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

double time=useEVTOLArray? simulatedTime : time();
for (Car car : cars) {
   double agentX = car.getX();
   double agentY = car.getY();

   if (car.crashZone.isEmpty())
   {
     for (Object[] record : evtolCrashZones)
     {
      car.crashZone.add(new Object[]{0, false});
     }
   }
   // Iterate through each fod zone
   int i=0;
   for (Object[] record : evtolCrashZones)
   {
      int evtolId = (int) record[0];
      double xImpact = (double) record[1];
      double yImpact = (double) record[2];
      double radius = (double) record[3];

      // Calculate distance between agent and crash zone center
      double distance = Math.sqrt(Math.pow(agentX - xImpact, 2) + Math.pow(agentY - yImpact, 2));
                        boolean inZone=false;
      // Check if the agent is within the crash radius
      if (distance <= radius)
      {
              inZone=true;
      }

      if(inZone!=(boolean)car.crashZone.get(i)[1])
      {
              AgentListCrashZone.add(new Object[]{time,i,evtolId, car.getId(), "car", inZone,pedestrians.size()+cars.size(),radius});
      }
      car.crashZone.set(i, new Object[]{evtolId, inZone});

      i++;
    }
}
for (Pedestrian ped : pedestrians) {
   double agentX = ped.getX();
   double agentY = ped.getY();

   if (ped.crashZone.isEmpty())
   {
     for (Object[] record : evtolCrashZones)
     {
      ped.crashZone.add(new Object[]{0, false});
     }
   }
   // Iterate through each crash zone
   int i=0;
   for (Object[] record : evtolCrashZones)
   {
      int evtolId = (int) record[0];
      double xImpact = (double) record[1];
      double yImpact = (double) record[2];
      double radius = (double) record[3];

      // Calculate distance between agent and crash zone center
      double distance = Math.sqrt(Math.pow(agentX - xImpact, 2) + Math.pow(agentY - yImpact, 2));
                        boolean inZone=false;
      // Check if the agent is within the crash radius
      if (distance <= radius)
      {
              inZone=true;
      }
      if(inZone!=(boolean)ped.crashZone.get(i)[1])
      {
              AgentListCrashZone.add(new Object[]{time,i,evtolId, ped.getId(), "ped", inZone,pedestrians.size()+cars.size(),radius});
      }
      ped.crashZone.set(i, new Object[]{evtolId, inZone});
      i++;
    }
}
```

```
//Function: void calculateBuildingCrashMetric()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

double[] buildingCrashMetricArr = new double[64];
int[] countArr = new int[64];
double[] totalVolumeArr = new double[64];

for(int j=0;j<64;j++)
{
                double startX = centreX-415-(double)evtolScenario.get(j)[0];
                double startY = centreY-415-(double)evtolScenario.get(j)[1];
                Point2D lineStart = new Point2D.Double(startX, startY);
                Point2D lineEnd = new Point2D.Double(centreX, centreY);

int count = 0;
double totalVolume=0.0;
double buildingCrashMetric=0.0;

  for (Object[] raw : buildingList) {
    List<Point2D> pts = new ArrayList<>();
    for (String pair : raw[4].toString().trim().split(" ")) {
      String[] xy = pair.split(",");
      pts.add(new Point2D.Double(Double.parseDouble(xy[0]), Double.parseDouble(xy[1])));
    }

    for (int i = 0; i < pts.size(); i++) {
      Point2D a1 = pts.get(i);
      Point2D a2 = pts.get((i + 1) % pts.size());

      double d1 = (lineEnd.getX() - lineStart.getX()) * (a1.getY() - lineStart.getY()) -
                  (lineEnd.getY() - lineStart.getY()) * (a1.getX() - lineStart.getX());
      double d2 = (lineEnd.getX() - lineStart.getX()) * (a2.getY() - lineStart.getY()) -
                  (lineEnd.getY() - lineStart.getY()) * (a2.getX() - lineStart.getX());
      double d3 = (a2.getX() - a1.getX()) * (lineStart.getY() - a1.getY()) -
                  (a2.getY() - a1.getY()) * (lineStart.getX() - a1.getX());
      double d4 = (a2.getX() - a1.getX()) * (lineEnd.getY() - a1.getY()) -
                  (a2.getY() - a1.getY()) * (lineEnd.getX() - a1.getX());

      if (d1 * d2 < 0 && d3 * d4 < 0) {
            count++;
        totalVolume+=Double.parseDouble(raw[2].toString());
        break;
      }
    }

  }
  buildingCrashMetric=count/(double)buildingList.size();
    buildingCrashMetricArr[j]=buildingCrashMetric;
    countArr[j]=count;
    totalVolumeArr[j]=totalVolume;
    buildingCrashMetrics.add(new Object[]{j, buildingCrashMetric,count,totalVolume});
  }
```

```
//Function: void calculateCrashMetric()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

// Map to store the final normalized occupancy per EVTOLIndex
Map<Integer, Double> zoneTimePerEVTOL = new HashMap<>();

for (int evtolIndex = 0; evtolIndex < 64; evtolIndex++) {
    List<Object[]> filtered = new ArrayList<>();
    double earliestTime = Double.MAX_VALUE;
    double latestTime = Double.MIN_VALUE;
    double totalAgentsSum = 0.0;
    int totalAgentsCount = 0;

    // Filter and collect stats
    for (Object[] row : AgentListCrashZone) {
        if ((int) row[1] == evtolIndex) {
            filtered.add(row);
            double time = (double) row[0];
            if (time < earliestTime) earliestTime = time;
            if (time > latestTime) latestTime = time;

            int totalAgents = (int) row[6];
            totalAgentsSum += totalAgents;
            totalAgentsCount++;
        }
    }

    if (filtered.isEmpty()) continue;

    double totalTime = latestTime - earliestTime;
    double avgTotalAgents = (totalAgentsCount > 0) ? totalAgentsSum / totalAgentsCount : 0.0;

    // Group by AgentID
    Map<String, List<Object[]>> agentEventsMap = new HashMap<>();
    for (Object[] row : filtered) {
        String agentID = String.valueOf(row[3]);
        agentEventsMap.computeIfAbsent(agentID, k -> new ArrayList<>()).add(row);
    }

    double numerator = 0.0;

    for (List<Object[]> agentEvents : agentEventsMap.values()) {
        agentEvents.sort(Comparator.comparingDouble(e -> (double) e[0]));

        for (int i = 0; i < agentEvents.size() - 1; i++) {
            boolean inZone = (boolean) agentEvents.get(i)[5];
            boolean nextInZone = (boolean) agentEvents.get(i + 1)[5];

            if (inZone && !nextInZone) {
                double t1 = (double) agentEvents.get(i)[0];
                double t2 = (double) agentEvents.get(i + 1)[0];
                double duration = t2 - t1;

                double r1 = (double) agentEvents.get(i)[7];
                double r2 = (double) agentEvents.get(i + 1)[7];
                double avgRadius = (r1 + r2) / 2.0;
                double area = Math.PI * avgRadius * avgRadius;

                if (area > 0 && duration > 0) {
                    numerator += duration ;//1.0;
                }
            }
        }
    }

    double denominator = avgTotalAgents ;
    double normalizedDensity = (denominator > 0) ? numerator / denominator : 0.0;
    zoneTimePerEVTOL.put(evtolIndex, normalizedDensity);
    crashMetrics.add(new Object[]{evtolIndex, normalizedDensity});
}
```

```
//Function: void calculateBuildingPrivacyList()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

double threshold=100.0;

if(!useEVTOLArray)
{
		int j=0;
		for (EVTOL evtol : eVTOLs) {
		  double buildingPrivacyMetric=0.0;

		  double x0 = evtol.getX();
		  double y0 = evtol.getY();
		  double h = evtol.getZ();  // eVTOL altitude
		  double theta = evtol.getRotation();  // eVTOL heading (radians)
		  double ve = evtol.getSpeed();  // eVTOL's speed (m/s)

				double currentX = centreX-415-x0;
				double currentY = centreY-415-y0;
				Point2D referencePoint = new Point2D.Double(currentX, currentY);
		 int count = 0;

		  for (Object[] raw : buildingList) {
		    List<Point2D> pts = new ArrayList<>();
		    for (String pair : raw[4].toString().trim().split(" ")) {
		      String[] xy = pair.split(",");
		      pts.add(new Point2D.Double(Double.parseDouble(xy[0]), Double.parseDouble(xy[1])));
		    }

		    boolean isClose = false;
		    for (Point2D p : pts) {
		    double distance = sqrt(referencePoint.distance(p)*referencePoint.distance(p)+ (h-Double.parseDouble(raw[1].toString()))*(h-
Double.parseDouble(raw[1].toString())));
		      if (distance <= threshold) {

		        isClose = true;
		        boolean existing=false;
		        for (Object[] record : buildingPrivacyList)
				{
				int i = (int) record[0];
				int evtolID = (int) record[1];
				String building = record[2].toString();
				if(i==j && evtolID==evtol.getId() && building==raw[0].toString())
				{
				   existing=true;
					break;
				}
		        }
		        if(!existing)
		        {
				buildingPrivacyList.add(new Object[]{j, evtol.getId(), raw[0].toString()});
		        }
		        break;
		      }
		    }

		    if (isClose) count++;
		  }

		    j++;
		  }
}
else
{
		int j=0;
		for (Object[] record : instantMetricsEVTOLArray) {
		   double x0 = (double)record[1];
		   double y0 = (double)record[2];
		   double h = (double)record[3];  // eVTOL altitude
		   double theta = 0.0;  // eVTOL heading (radians)
		   double ve = (double)record[4];  // eVTOL's speed (m/s)

				double currentX = centreX-415-x0;
				double currentY = centreY-415-y0;
				Point2D referencePoint = new Point2D.Double(currentX, currentY);


		 int count = 0;

		  for (Object[] raw : buildingList) {
```

```
                List<Point2D> pts = new ArrayList<>();
                for (String pair : raw[4].toString().trim().split(" ")) {
                  String[] xy = pair.split(",");
                  pts.add(new Point2D.Double(Double.parseDouble(xy[0]), Double.parseDouble(xy[1])));
                }

                boolean isClose = false;
                for (Point2D p : pts) {
                double distance = sqrt(referencePoint.distance(p)*referencePoint.distance(p)+ (h-Double.parseDouble(raw[1].toString()))*(h-
Double.parseDouble(raw[1].toString())));
                  if (distance <= threshold) {

                     isClose = true;
                     boolean existing=false;
                     for (Object[] plist : buildingPrivacyList)
                              {
                              int i = (int) plist[0];
                              int evtolID = (int) plist[1];
                              String building = plist[2].toString();
                              if(i==j && evtolID==(int)record[0] && building==raw[0].toString())
                              {
                                 existing=true;
                                          break;
                              }
                     }
                     if(!existing)
                     {
                              buildingPrivacyList.add(new Object[]{j, (int)record[0], raw[0].toString()});
                     }
                     break;
                  }
                }

                if (isClose) count++;
              }
               j++;
              }
}
```

```
//Function: void calculateBuildingPrivacyMetric()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)
double[] buildingPrivacyMetricArr = new double[64];
for(int j=0; j<64;j++)
{
		int count=0;
		double buildingPrivacyMetric=0.0;
		for (Object[] record : buildingPrivacyList)
		{
			int i = (int) record[0];
			if(i==j)
			{
			 count++;
			}
		}
		buildingPrivacyMetric=count/(double)buildingList.size();
		buildingPrivacyMetricArr[j]=buildingPrivacyMetric;
  // System.out.println(j+"," + buildingPrivacyMetric);
  buildingPrivacyMetrics.add(new Object[]{j, buildingPrivacyMetric});

}
```

```
//Function: void calculateIntegratedMetric()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)
double[] buildingCrashMetricArr = new double[64];
int[] buildingCountArr = new int[64];
double[] buildingTotalVolumeArr = new double[64];
double[] buildingPrivacyMetricArr = new double[64];
double[] FODMetricArr = new double[64];
double[] CrashMetricArr = new double[64];
double[] integratedMetricArr = new double[64];
double allBuildingsTotalVolume=7947524.944;
double maxCrashMetric=0.0;
double maxFODMetric=0.0;
double maxBuildingPrivacyMetric=0.0;
double maxBuildingCrashMetric=0.0;
double kgc=1.0;
double kfo=0.25;
double kp=0.10;
for(int j=0; j<64;j++)
{
		for (Object[] record : buildingPrivacyMetrics)
		{
				int i = (int) record[0];
				if(i==j)
				{
				 buildingPrivacyMetricArr[j]=(double)record[1];
				 if((double)record[1]>maxBuildingPrivacyMetric)maxBuildingPrivacyMetric=(double)record[1];
				}
		}
}
for(int j=0; j<64;j++)
{
		for (Object[] record : buildingCrashMetrics)
		{
				int i = (int) record[0];
				if(i==j)
				{
				 buildingCrashMetricArr[j]=(double)record[1];
				 buildingCountArr[j]=(int)record[2];
				 buildingTotalVolumeArr[j]=(double)record[3];
				 if((double)record[1]>maxBuildingCrashMetric)maxBuildingCrashMetric=(double)record[1];
				}
				}
}
for(int j=0; j<64;j++)
{
		for (Object[] record : crashMetrics)
		{
				int i = (int) record[0];
				if(i==j)
				{
				 CrashMetricArr[j]=(double)record[1];
				 if((double)record[1]>maxCrashMetric)maxCrashMetric=(double)record[1];
				}
				}
}
for(int j=0; j<64;j++)
{
		for (Object[] record : FODMetrics)
		{
				int i = (int) record[0];
				if(i==j)
				{
				 FODMetricArr[j]=(double)record[1];
				 if((double)record[1]>maxFODMetric)maxFODMetric=(double)record[1];
				}
				}
}
for(int j=0; j<64;j++)
{
		if(maxBuildingCrashMetric<0.0000001)maxBuildingCrashMetric=1.0;
		if(maxCrashMetric<0.0000001)maxCrashMetric=1.0;
		if(maxBuildingPrivacyMetric<0.0000001)maxBuildingPrivacyMetric=1.0;
		if(maxFODMetric<0.0000001)maxFODMetric=1.0;
		double intMetric =((buildingTotalVolumeArr[j]/(3.0*10.0))*0.1/1180.0)*(buildingCrashMetricArr[j]/maxBuildingCrashMetric)+
				kgc*(CrashMetricArr[j]/maxCrashMetric)+   kp*(buildingPrivacyMetricArr[j]/maxBuildingPrivacyMetric)+
					kfo*(FODMetricArr[j]/maxFODMetric);
				integratedMetricArr[j]=intMetric;
		integratedMetrics.add(new Object[]{j, intMetric});

}
```

```
//Function: void hereAPICall()
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

String url =
"https://data.traffic.hereapi.com/v7/flow?in=circle:"+refLat+","+refLon+";r="+radius+"&locationReferencing=shape&apiKey=GZryveizgT8xvsHZKf3bPzoNx43Oi
Keu6kIX0fySFls";
String json = httpGet(url);
hereAPIJsonResponse = json;
if (json != null && !json.isEmpty()) {
                    traceln(json);
                    //renderHereFlowJson(json);
                    if(jamFactorVisualization)
                    {
                                renderHereFlowJson2D(json);   // jamfactor colorization presentation
                    }

                   trafficDelay.stopDelayForAll();
                   renderHereFlowAgents(json);  //agent dots
                   calculateInstantMetrics();

                  } else {
                    traceln("Empty traffic response");
}

hereAPIUpdating=false;
```

```
//Function: Point2D latLonToXYLocal(double,double)
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

    double metersPerDegLat = 111_320.0;
    double metersPerDegLon = 111_320.0 * Math.cos(Math.toRadians(refLat));

    double x = (lon - refLon) * metersPerDegLon;    // east = right
    double y = -(lat - refLat) * metersPerDegLat;   // north = up → negative Y

    return new Point2D.Double(x, y);
```

```
//Function: void renderHereFlowJson2D (string)
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

try {
    clearTrafficLines();
    if (json == null || json.isEmpty()) return;

    // Draw a small center dot
    ShapeOval centerDot = new ShapeOval();
    centerDot.setRadius(3.0);
    centerDot.setFillColor(Color.BLACK);
    centerDot.setLineColor(null);
    centerDot.setPos(0, 0);
    traffic2DLayer.add(centerDot);
    hereTrafficShapes2D.add(centerDot);

    // Parse HERE JSON
    JSONObject root    = new JSONObject(json);
    JSONArray results  = root.getJSONArray("results");

    for (int i = 0; i < results.length(); i++) {
      JSONObject res      = results.getJSONObject(i);
      JSONObject location = res.getJSONObject("location");
      JSONObject current  = res.getJSONObject("currentFlow");

      double jamFactor = current.optDouble("jamFactor", Double.NaN);
      Color segColor   = jamToColor(jamFactor);

      JSONObject shape = location.getJSONObject("shape");
      JSONArray links  = shape.getJSONArray("links");

      for (int l = 0; l < links.length(); l++) {
        JSONObject link  = links.getJSONObject(l);
        JSONArray points = link.getJSONArray("points");
        if (points.length() < 2) continue;

        JSONObject p0 = points.getJSONObject(0);
        double lat0   = p0.getDouble("lat");
        double lon0   = p0.getDouble("lng");
        Point2D xy0   = latLonToXYLocal(lat0, lon0);
        double x0     = xy0.getX();
        double y0     = xy0.getY();

        for (int p = 1; p < points.length(); p++) {
          JSONObject pj = points.getJSONObject(p);
          double lat1   = pj.getDouble("lat");
          double lon1   = pj.getDouble("lng");
          Point2D xy1   = latLonToXYLocal(lat1, lon1);
          double x1     = xy1.getX();
          double y1     = xy1.getY();

          // Clip the segment to the circle in XY
          double[] clipped = clipSegmentToCircleXY(x0, y0, x1, y1, radius);
          if (clipped != null) {
            double cx0 = clipped[0];
            double cy0 = clipped[1];
            double cx1 = clipped[2];
            double cy1 = clipped[3];

            ShapePolyLine line = new ShapePolyLine();
            line.setLineColor(segColor);
            line.setLineWidth(6.0);
            line.setNPoints(2);
            line.setPoint(0, cx0, cy0);
            line.setPoint(1, cx1, cy1);

            traffic2DLayer.add(line);
            hereTrafficShapes2D.add(line);
          }

          // Next segment starts where this one ended
          x0 = x1;
          y0 = y1;
        }
      }
    }
  } catch (Exception e) {
    trace("renderHereFlowJson2D error: " + e.getMessage());
    e.printStackTrace();
  }
```

```
//Function: double[] clipSegmentToCircleXY(double,double,double,double,double)
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

double r2 = r * r;

    boolean in0 = x0 * x0 + y0 * y0 <= r2;
    boolean in1 = x1 * x1 + y1 * y1 <= r2;

    // If both inside, no clipping needed
    if (in0 && in1) return new double[]{ x0, y0, x1, y1 };

    double dx = x1 - x0;
    double dy = y1 - y0;
    double a  = dx * dx + dy * dy;
    if (a == 0) return null;  // degenerate

    double b = 2 * (x0 * dx + y0 * dy);
    double c = x0 * x0 + y0 * y0 - r2;

    double disc = b * b - 4 * a * c;
    if (disc < 0) return null;  // no intersection

    double sqrtD = Math.sqrt(disc);
    double t1 = (-b - sqrtD) / (2 * a);
    double t2 = (-b + sqrtD) / (2 * a);

    List<Double> ts = new ArrayList<>();
    if (t1 >= 0 && t1 <= 1) ts.add(t1);
    if (t2 >= 0 && t2 <= 1) ts.add(t2);
    if (ts.isEmpty()) return null;

    double cx0 = x0, cy0 = y0;
    double cx1 = x1, cy1 = y1;

    if (!in0) {
        double t = ts.get(0);
        cx0 = x0 + t * dx;
        cy0 = y0 + t * dy;
    }
    if (!in1) {
        double t = ts.size() == 1 ? ts.get(0) : ts.get(ts.size() - 1);
        cx1 = x0 + t * dx;
        cy1 = y0 + t * dy;
    }

    boolean newIn0 = cx0 * cx0 + cy0 * cy0 <= r2 + 1e-6;
    boolean newIn1 = cx1 * cx1 + cy1 * cy1 <= r2 + 1e-6;
    if (!newIn0 && !newIn1) return null;

    return new double[]{ cx0, cy0, cx1, cy1 };
```

```
//Function: Color jamToColor(double)
//The following code was generated with the assistance of an AI code assistant (ChatGPT)


if (Double.isNaN(jam)) return new Color(128,128,128,190); // unknown = gray
  if (jam < 2.0) return new Color(  0,180,  0,220);        // free
  if (jam < 4.0) return new Color(173,255, 47,220);        // light
  if (jam < 7.0) return new Color(255,165,  0,220);        // medium
  if (jam < 9.0) return new Color(255, 69,  0,220);        // heavy
  return new Color(178, 34, 34,220);                       // severe



//Function: double SpacingMeters(double,double)
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

 // defaults if not provided
    double freeKmh = Double.isNaN(freeFlow) ? 50.0 : freeFlow * 3.6;

    // base spacing: at jam=0 → 150m, at jam=10 → 20m
    double maxSpacing = 150.0;
    double minSpacing = 20.0;
    double j01 = Math.max(0.0, Math.min(1.0, jam / 10.0));
    double byJam = maxSpacing - j01 * (maxSpacing - minSpacing);

    // adjust by speed: 50 km/h → factor 1; 100 km/h → ~1.5; 25 → ~0.5
    double speedFactor = Math.max(0.5, Math.min(1.5, freeKmh / 50.0));

    return byJam * speedFactor;
```

```
//Function: void renderHereFlowAgents(string)
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

try {
   // Clear any old spawn points for this new batch
   trafficSpawnPoints.clear();
   if (json == null || json.length() == 0)
      return;
   JSONObject root    = new JSONObject( json );
   JSONArray results  = root.getJSONArray( "results" );

   for (int i = 0; i < results.length(); i++) {
      JSONObject res      = results.getJSONObject( i );
      JSONObject location = res.getJSONObject( "location" );
      JSONObject current  = res.getJSONObject( "currentFlow" );

      // HERE fields (m/s and 0–10)
      double jamFactor = current.optDouble( "jamFactor", Double.NaN );
      double freeFlow  = current.optDouble( "freeFlow",  Double.NaN );  // m/s

      // Decide spacing in meters for this road "location"
      double spacing = spacingMeters( freeFlow, jamFactor );
      if (spacing <= 0.0) spacing = 50.0;

      JSONObject shape = location.getJSONObject( "shape" );
      JSONArray links  = shape.getJSONArray( "links" );

      for (int l = 0; l < links.length(); l++) {
         JSONObject link  = links.getJSONObject( l );
         JSONArray points = link.getJSONArray( "points" );
         if (points.length() < 2)
            continue;

         // First point in this link
         JSONObject p0 = points.getJSONObject( 0 );
         double lat0   = p0.getDouble( "lat" );
         double lon0   = p0.getDouble( "lng" );
         Point2D xy0   = latLonToXYLocal( lat0, lon0 );
         double x0     = xy0.getX();
         double y0     = xy0.getY();

         for (int p = 1; p < points.length(); p++) {
            JSONObject pj = points.getJSONObject( p );
            double lat1   = pj.getDouble( "lat" );
            double lon1   = pj.getDouble( "lng" );
            Point2D xy1   = latLonToXYLocal( lat1, lon1 );
            double x1     = xy1.getX();
            double y1     = xy1.getY();

            // Clip this segment to the circle centered at (0,0) with radius 'radius'
            double[] clipped = clipSegmentToCircleXY( x0, y0, x1, y1, radius );
            if (clipped != null) {
               double cx0 = clipped[0];
               double cy0 = clipped[1];
               double cx1 = clipped[2];
               double cy1 = clipped[3];

               double dx  = cx1 - cx0;
               double dy  = cy1 - cy0;
               double len = Math.hypot( dx, dy );

               if (len > 1e-6) {
                  double step = spacing;

                  // Heading along this clipped segment (constant for all points on it)
                  double headingRad = Math.atan2( dy, dx );

                  // Start halfway along the first interval to avoid dots exactly at junctions
                  for (double d = step / 2.0; d < len; d += step) {
                     double t  = d / len;
                     double pxLocal = cx0 + t * dx;
                     double pyLocal = cy0 + t * dy;

                     // Apply your offset (same as before)
                     double px = pxLocal - 415.0;
                     double py = pyLocal - 415.0;

                     // Store x, y, headingRad
                     trafficSpawnPoints.add( new Object[] { px, py, headingRad } );
```

```
                    // Inject one agent for this spawn point
                    trafficSource.inject( 1 );
                }
            }
        }

        // Advance along polyline
        x0 = x1;
        y0 = y1;
    }
  }
 }

} catch (Exception e) {
  trace( "renderHereFlowAgents error: " + e.getMessage() );
  e.printStackTrace();
}
```

```
//Function: void drawBuildingCentroidsAndConvexHulls(string,double,double)
//The following code was generated with the assistance of an AI code assistant (ChatGPT)

try {
    BufferedReader br = new BufferedReader(new FileReader(filePath));
    String line;
    int rowIndex = 0;

    // Read through the file line by line
    while ((line = br.readLine()) != null) {
        rowIndex++;
        // Skip the header row if necessary
        if (rowIndex == 1) {
            continue;
        }

        // Parse the CSV line, taking into account quoted fields
        List<String> columns = new ArrayList<>();
        StringBuilder currentColumn = new StringBuilder();
        boolean inQuotes = false;

        for (int i = 0; i < line.length(); i++) {
            char c = line.charAt(i);

            if (inQuotes) {
                if (c == '\"') {
                    // Look ahead to see if this is a double quote for escaping
                    if (i + 1 < line.length() && line.charAt(i + 1) == '\"') {
                        currentColumn.append(c);
                        i++; // Skip the next character
                    } else {
                        inQuotes = false;
                    }
                } else {
                    currentColumn.append(c);
                }
            } else {
                if (c == '\"') {
                    inQuotes = true;
                } else if (c == ',') {
                    columns.add(currentColumn.toString());
                    currentColumn.setLength(0); // Clear the buffer
                } else {
                    currentColumn.append(c);
                }
            }
        }

        columns.add(currentColumn.toString()); // Add the last column

        // Extract the centroid from the 6th column
        if (columns.size() > 5) { // Ensure there are enough columns
            String pointStr = columns.get(5);

            // Assuming the point is formatted as "x,y"
            String[] point = pointStr.split(",");
            double centroidX = (Double.parseDouble(point[0].trim()) - xOffset);
            double centroidY = -1.0 * (Double.parseDouble(point[1].trim()) - yOffset);

            // Check if the centroid is within the specified range
            if (Math.sqrt(centroidX*centroidX + centroidY*centroidY)<areaRadius/2.0){//(centroidX >= minimumX && centroidX <= maximumX && centroidY >=
minimumY && centroidY <= maximumY) {

                                                    //add building to buildingList collection
                                                    buildingList.add(new Object[]{columns.get(0), columns.get(1), columns.get(2), columns.get(3), columns.get(4),
columns.get(5), columns.get(6), columns.get(7), columns.get(8)});

                // Process the 5th column for polylines
                if (columns.size() > 4) { // Ensure there are enough columns
                    String pointsStr = columns.get(4);

                    // Split the points string by spaces to get individual points
                    String[] pointsArray = pointsStr.split(" ");

                    double[] dx = new double[pointsArray.length];
                    double[] dy = new double[pointsArray.length];

                    double startX = 0;
                    double startY = 0;
```

```
// Process each point in the array
for (int i = 0; i < pointsArray.length; i++) {
    // Split each point by comma to get x and y coordinates
    String[] coords = pointsArray[i].split(",");
    double x = Double.parseDouble(coords[0].trim());
    double y = Double.parseDouble(coords[1].trim());

    if (i == 0) {
        // Store the starting point with offset applied
        startX = (x - xOffset);
        startY = -1.0 * (y - yOffset);
    }

    // Apply the offset and make the coordinates relative to the starting point
    dx[i] = (x - xOffset) - startX;
    dy[i] = -1.0 * (y - yOffset) - startY;
}

// Create a polyline from the points

ShapePolyLine polyline = new ShapePolyLine(
    true,       // ispublic
    startX-areaRadius/2.0,      // x coordinate of the start point with offset
    startY-areaRadius/2.0,      // y coordinate of the start point with offset
    black, // line color
    null,       // fill color (no fill)
    pointsArray.length, // number of points
    dx,         // x coordinates relative to the start point
    dy,         // y coordinates relative to the start point
    true,       // closed polyline
    1.0,        // line width
    LINE_STYLE_SOLID // line style
);

presentation.add(polyline); // Add the polyline to the environment

// Create a new Wall
Wall Thiswall = new Wall();

// Start drawing from the first point
Thiswall.startDrawing(startX + dx[0] - areaRadius / 2.0, startY + dy[0] - areaRadius / 2.0, 0); // z = 0 for ground level

// Iterate through pointsArray to add each segment, but exclude the last point if it repeats the first one
or (int i = 1; i < pointsArray.length; i++) {
 double x = startX + dx[i] - areaRadius / 2.0;
double y = startY + dy[i] - areaRadius / 2.0;

// Check if this is the last point and if it's the same as the first point
if (i == pointsArray.length - 1 && x == (startX + dx[0] - areaRadius / 2.0) && y == (startY + dy[0] - areaRadius / 2.0)) {
  continue; // Skip the last point if it's the same as the first
}

 Thiswall.lineTo(x, y, 0); // z = 0 to keep the wall at ground level
}

// Manually close the wall by linking the last point to the first
Thiswall.lineTo(startX + dx[0] - areaRadius / 2.0, startY + dy[0] - areaRadius / 2.0, 0);

// Set additional wall properties
Thiswall.setZHeight(3.0); // Wall height in meters
Thiswall.setColor(Color.BLACK); // Set wall color to black
Thiswall.setVisible(true);
Thiswall.setOwner(this);
Thiswall.setLineWidth(1);
Thiswall.setLevel(wall61.getLevel());
Thiswall.initializeInternal();

presentation.add(Thiswall);
Thiswall.setVisible(false);


// Create an oval at the centroid position
ShapeOval oval = new ShapeOval();
oval.setX(centroidX-areaRadius/2.0);
oval.setY(centroidY-areaRadius/2.0);
oval.setRadius(2); // Set the desired radius
presentation.add(oval); // Add the oval to the environment

// Print a message to the console with the centroid position
System.out.println("Added polyline and centroid at position: (" + centroidX + ", " + centroidY + ")");
```

```
                                                    System.out.println(load3DBuildings);
                                                if(load3DBuildings)
                                                {
                                // Check if the 3D object file exists in the "3d" subfolder
                                if (columns.size() > 8) { // Ensure there is a first column
                                   String objectName = columns.get(0).trim();
                                   String objectFilePath = "3d/" + objectName + ".dae";
                                   double buildingwidthfloat = Math.ceil(Double.parseDouble(columns.get(8)));
                                   double buildingheightfloat = Math.ceil(Double.parseDouble(columns.get(7)));

                                   double buildingtopx = -1.0* Math.ceil(buildingwidthfloat/2);
                                   double buildingtopy = -1.0* Math.ceil(buildingheightfloat/2);

                                   File objectFile = new File(objectFilePath);

                                   if (objectFile.exists()) {
                                      // Load the 3D object and place it at the centroid position

                                      Shape3DObject building  = new Shape3DObject(
                                 Main.this, SHAPE_DRAW_3D, true, centroidX-areaRadius/2.0, centroidY-areaRadius/2.0, 0.0, 1.5707963267948966,
                                                                                                                    1.0, false, "/casestudy1/",
                                                                                                                          objectFilePath,
OBJECT_3D_YZX_AXIS_ORDER, Object3DInternalLighting.OBJECT_3D_INTERNAL_LIGHTING_OFF, false, buildingtopx, buildingtopy,
                                                                                                                          buildingwidthfloat, buildingheightfloat, null,true);

                                      presentation.add(building); // Add the 3D object to the environment

                                      // Print a message to the console
                                      System.out.println("Added 3D object from file: " + objectFilePath);
                                   } else {
                                      // Print a message indicating the 3D file was not found
                                      System.out.println("3D object file not found: " + objectFilePath);
                                   }
                                }
                             }
                          }
                       }
                    }
                 }

                 br.close();

              } catch (Exception e) {
                 e.printStackTrace();
              }
```

```
# Script: citygmltocsv_all.py
# The following code was generated with the assistance of an AI code assistant (ChatGPT)

import csv
import numpy as np
from scipy.spatial import ConvexHull
from lxml import etree


def parse_citygml(file_path, csvwriter):
    ns = {
        'gml': 'http://www.opengis.net/gml',
        'bldg': 'http://www.opengis.net/citygml/building/2.0',
        'gen': 'http://www.opengis.net/citygml/generics/2.0'
    }

    print(f"Processing file: {file_path}...")

    tree = etree.parse(file_path)
    root = tree.getroot()

    def corrected_polygon_centroid(hull_points):
        x = hull_points[:, 0]
        y = hull_points[:, 1]
        A = 0.0
        Cx = 0.0
        Cy = 0.0
        for i in range(-1, len(hull_points) - 1):
            common_factor = (x[i] * y[i + 1]) - (x[i + 1] * y[i])
            A += common_factor
            Cx += (x[i] + x[i + 1]) * common_factor
            Cy += (y[i] + y[i + 1]) * common_factor
        A *= 0.5
        Cx *= (1.0 / (6.0 * A))
        Cy *= (1.0 / (6.0 * A))
        return np.array([Cx, Cy])

    for building in root.findall('.//bldg:Building', namespaces=ns):
        building_id = building.attrib.get('{http://www.opengis.net/gml}id')

        # Extract the building volume
        volume_element = building.find('.//gen:doubleAttribute[@name="Volume"]/gen:value', namespaces=ns)
        building_volume = volume_element.text if volume_element is not None else 'N/A'

        # Extract ground surface area
        ground_surfaces = building.findall('.//bldg:GroundSurface', namespaces=ns)
        ground_surface_area = 'N/A'
        all_points = []
        point_list = []

        for ground_surface in ground_surfaces:
            area_element = ground_surface.find('.//gen:doubleAttribute[@name="Area"]/gen:value', namespaces=ns)
            if area_element is not None:
                ground_surface_area = area_element.text

            pos_lists = ground_surface.findall('.//gml:posList', namespaces=ns)

            for pos_list in pos_lists:
                coords = pos_list.text.strip().split()
                points = []
                # Loop through the coordinates, ignoring the last triplet (X, Y, Z)
                for i in range(0, len(coords) - 3, 3):
                    x = float(coords[i])
                    y = float(coords[i + 1])
                    points.append(f"{x},{y}")
                    point_list.append([x, y])

                all_points.extend(points)

        # Extract building height using the bounding box (Envelope)
        lower_corner = building.find('.//gml:Envelope/gml:lowerCorner', namespaces=ns)
        upper_corner = building.find('.//gml:Envelope/gml:upperCorner', namespaces=ns)
        building_height = 'N/A'

        if lower_corner is not None and upper_corner is not None:
            lower_z = float(lower_corner.text.split()[-1])
            upper_z = float(upper_corner.text.split()[-1])
            building_height = round(upper_z - lower_z, 1)  # Round to 1 decimal place

        # Perform Convex Hull, Centroid, and Width/Height calculations
```

```
        if point_list:
            points_array = np.array(point_list)
            hull = ConvexHull(points_array)
            hull_points = points_array[hull.vertices]
            centroid = corrected_polygon_centroid(hull_points)

            # Calculate width and height
            min_x = np.min(points_array[:, 0])
            max_x = np.max(points_array[:, 0])
            min_y = np.min(points_array[:, 1])
            max_y = np.max(points_array[:, 1])
            width = round(max_x - min_x, 2)
            height = round(max_y - min_y, 2)

            # Format hull points and centroid for CSV output
            hull_points_str = " ".join([f"{x},{y}" for x, y in hull_points])
            centroid_str = f"{centroid[0]},{centroid[1]}"
        else:
            hull_points_str = 'N/A'
            centroid_str = 'N/A'
            width = 'N/A'
            height = 'N/A'

        if all_points:
            csvwriter.writerow(
                [building_id, building_height, building_volume, ground_surface_area, hull_points_str, centroid_str,
                 " ".join(all_points), width, height])

    print(f"Finished processing file: {file_path}")


if __name__ == "__main__":
    output_file = 'output_file.csv'

    with open(output_file, 'w', newline='') as csvfile:
        csvwriter = csv.writer(csvfile)
        csvwriter.writerow(['BuildingID', 'Height', 'Volume', 'GroundSurfaceArea', 'ConvexHullPoints', 'Centroid',
                            'GroundSurfacePoints', 'Width', 'Height'])

        # Loop through files numbered 01 to 18
        for i in range(1, 19):
            input_file = f'VM{i:02d}_2016.gml'  # File names like 'VM01_2016.gml', 'VM02_2016.gml', etc.
            print(f"Starting to process {input_file}")
            parse_citygml(input_file, csvwriter)
        print("All files processed successfully.")
```