SimUAM: A Comprehensive Microsimulation Toolchain to Evaluate the Impact of Urban Air Mobility in Metropolitan Areas

Pavan Yedavalli, Urban Analytics Lab / Cal Unmanned Lab, Institute of Transportation Studies, University of California, Berkeley

Emin Burak Onat, Cal Unmanned Lab, Institute of Transportation Studies, University of California, Berkeley

Xin Peng, Cal Unmanned Lab, Institute of Transportation Studies, University of California, Berkeley

Raja Sengupta, Cal Unmanned Lab, Institute of Transportation Studies, University of California, Berkeley

Paul Waddell, Cal Unmanned Lab, Institute of Transportation Studies, University of California, Berkeley

Vishwanath Bulusu, Crown Consulting Inc., NASA Ames Research Center

Min Xue, NASA Ames Research Center

August, 2021

UCB-ITS-WP-2021-02

SimUAM: A Comprehensive Microsimulation Toolchain to Evaluate the Impact of Urban Air Mobility in Metropolitan Areas

Pavan Yedavalli*, Emin Burak Onat†, Xin Peng ‡, Raja Sengupta§, Paul Waddell¶
University of California, Berkeley, Berkeley, CA, 94720

Vishwanath Bulusu‖, Min Xue**
NASA Ames Research Center, Moffett Field, CA 94035

Over the past several years, Urban Air Mobility (UAM) has galvanized enthusiasm from investors and researchers, marrying expertise in aircraft design, transportation, logistics, artificial intelligence, battery chemistry, and broader policymaking. However, two significant questions remain unexplored: (1) What is the value of UAM in a region's transportation network?, and (2) How can UAM be effectively deployed to realize and maximize this value to all stakeholders, including riders and local economies? To adequately understand the value proposition of UAM for metropolitan areas, we develop a holistic multi-modal toolchain, SimUAM, to model and simulate UAM and its impacts on travel behavior. This toolchain has several components: (1) MANTA: A fast, high-fidelity regional-scale traffic microsimulator, (2) VertiSim: A granular, discrete-event vertiport and pedestrian, (3) FE^3: A high-fidelity, trajectory-based aerial microsimulation. SimUAM, rooted in granular, GPU-based microsimulation, models millions of trips and their exact movements in the street network and in the air, producing interpretable and actionable performance metrics for UAM designs and deployments. The modularity, extensibility, and speed of the platform will allow for rapid scenario planning and sensitivity analysis, effectively acting as a detailed performance assessment tool. As a result, stakeholders in UAM can understand the impacts of critical infrastructure, and subsequently define policies, requirements, and investments needed to support UAM as a viable transportation mode.

I. Introduction

When a new mode is introduced into a metropolitan transportation network, its impact on individual travel behavior is often not readily understood [1]. To proponents, UAM directly tackles one of the known parameters of mode choice, travel time, a theoretically and empirically known 'bad', or a factor that increases disutility [2, 3]. Assuming UAM operations will use an on-demand model that includes aircraft flying along high-volume travel corridors between a network of vertiports, analyses have attempted to estimate the potential impacts of UAM passenger services on travel behavior, using predetermined vertiport locations and simulating different modes of travel in different metropolitan areas [4–8]. However, these existing models assume static parameters in either ground traffic, vertiport transfer times, aerial flight, or a combination of the three. For UAM, since its purported value is in travel time savings, simulating the margins at a granular level, incorporating exact dynamics and scheduling in each leg of the trip, is of utmost importance.

This paper proposes SimUAM, a holistic multi-modal toolchain that integrates ground traffic microsimulation, ground-air interfacing at the vertiport, and aerial microsimulation, producing exact travel times for both multi-modal UAM trips as well as other driving trips across the region. Specifically, the MANTA component of SimUAM models road congestion patterns of all driving trips, including the access and egress legs of the multi-modal UAM trips. VertiSim models the exact ground-air interface at the vertiport to move the traveler from the ground to a specific aircraft

*Ph.D Candidate, Department of City and Regional Planning/Civil and Environmental Engineering, 230 Wurster Hall, Berkeley, CA 94720
†Ph.D Student, Department of Civil and Environmental Engineering, 100 McLaughlin Hall, Berkeley, CA 94720
‡Ph.D Candidate, Department of Civil and Environmental Engineering, 100 McLaughlin Hall, Berkeley, CA 94720
§Professor, Department of Civil and Environmental Engineering, 100 McLaughlin Hall
¶Professor, Department of City and Regional Planning/Civil and Environmental Engineering, 230 Wurster Hall, Berkeley, CA 94720
‖Aerospace Research Scientist, Crown Consulting Inc., Aviation Systems Division, NASA Ames Research Center, Moffett Field, CA 94035, USA
**Lead Researcher

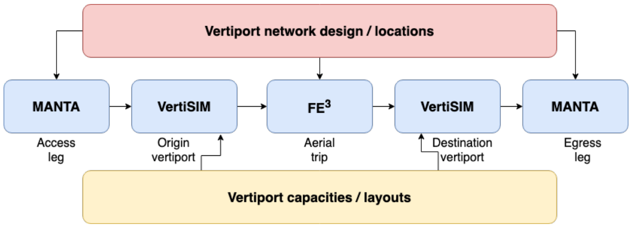


Fig. 1 Simulation components for an individual multi-modal UAM trip

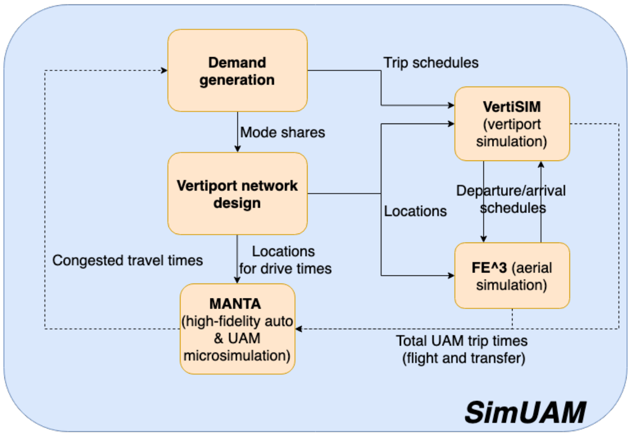

Demand generation
Trip schedules
VertiSIM (vertiport simulation)
Mode shares
Vertiport network design
Departure/arrival schedules
Locations
Congested travel times
Locations for drive times
FE^3 (aerial simulation)
MANTA (high-fidelity auto & UAM microsimulation)
Total UAM trip times (flight and transfer)
SimUAM

Fig. 2 The full SimUAM model pipeline

for the multi-modal UAM trips. The FE^3 component models the exact flight trajectories of these aircraft, incorporating conflict resolution and real UAV mission profiles in the sky. For each multi-modal UAM trip, the *interaction* among the three legs is exceedingly important in realizing the true value of UAM in a regional transportation network. As a result, SimUAM is the first holistic toolchain, to the best of these authors' knowledge, that models all these interactions together dynamically and granularly, as shown in Fig. 1.

This paper is organized into the following sections. Section II details MANTA, the GPU-based regional-scale road traffic microsimulator. Section III explains VertiSim, the granular, discrete-event vertiport and pedestrian simulator. Section IV discusses FE^3, a NASA-developed, GPU-based high-fidelity regional-scale microsimulator for air traffic. These three components comprise the SimUAM architecture, shown in Fig. 2. Section VI describes the performance metrics measured by SimUAM and experiments showing SimUAM's benefit. Finally, Section VII and Section VIII highlight the limitations, future work, and conclusions of this research.

II. MANTA: Microsimulation Analysis for Network Traffic Assignment

The SimUAM framework relies on accurate ground traffic microsimulation in order to produce congested travel times for all of the travel demand in the investigated region. In order to do so, we leverage Microsimulation Analysis for Network Traffic Assignment (MANTA), an ultra-fast, highly-parallelized GPU-based microsimulation platform, developed by this author [9]. Existing simulators have typically revealed a tradeoff among accuracy, computational speed, and geographic scale of simulation [9–13]. However, MANTA exhibits performance benefits in all three of these areas, enabling it to be used for agile scenario planning, particularly with an emergent mode such as UAM, whose deployment is still in its inchoate stages. MANTA's ability to track the microscopic movements of vehicles on the street network, namely lane changes, acceleration, and braking with respect to other vehicles, enables a granularity that remains vital in travel time computations [9].

A. Inputs

MANTA takes two major inputs: (1) Geographic network, comprised of edges and nodes in a metropolitan geography's street network, and (2) Origin-destination (OD) demand showing traveler movements in the selected

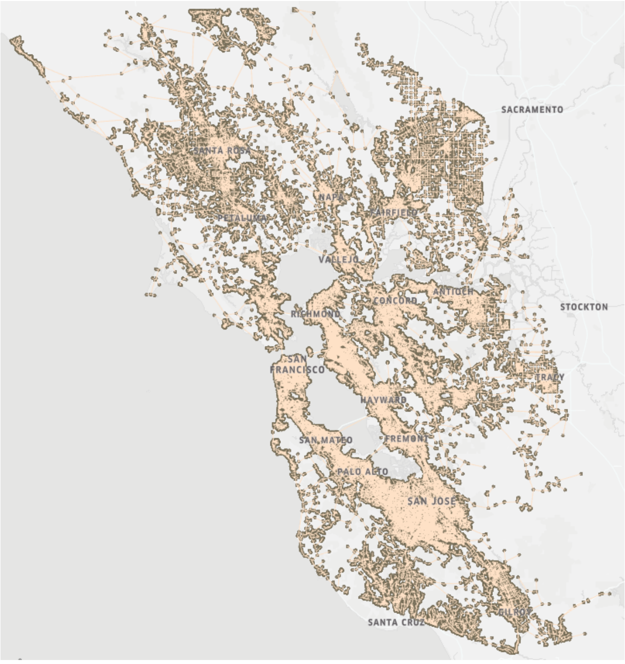


Fig. 3 The San Francisco Bay Area region's edges and nodes, defined by the polygonal hull of its nine counties

geography.

The geographic network, as described in [9], is derived from OpenStreetMap. We use the network library OSMnx to determine the nodes that exist within the polygonal hull of a metropolitan region, determined from widely available shapefiles for the region of investigation [14]. The San Francisco (SF) Bay Area is shown in Fig. 3 as an example output of this process, and is also used as the area of investigation in Section VI of this paper. The SF Bay Area is chosen as the case study due to wide travel data availability and massive geographic scale that reflects useful bidirectional commuting patterns.

The OD demand is generated from the local metropolitan planning organization (MPO), which typically calibrates a travel demand model based on estimates from census and household travel data [15]. For the SF Bay Area, the data are derived from the Bay Area Metropolitan Transportation Commission (MTC)'s Travel Demand One model.

Once each trip is assigned a mode, these trips are filtered to only driving trips, which include driving alone, shared trips, and driving to transit. In the SF Bay Area, this totals 25M trips for the full day. This work focuses on the morning commute, from 5am-12pm, and is filtered down to approximately 3M trips. Importantly, these trips may include multiple passengers in the car, but these extra passengers are not considered in the number of trips.

The origin and destination of each trip is at a traffic analysis zone (TAZ) granularity, which is a population density-based geographic unit used by MPOs across the nation, with each TAZ area typically larger than a block but smaller than a zip code. This size is too coarse for MANTA, which operates at the node level. As a result, once the origin and destination TAZs are known, we randomly assign nodes to the origin and destination within their respective TAZs, sampled from a uniform distribution, producing each trip as one from an origin node to a destination node.

B. Functionality

In MANTA, there are two major components: (1) routing and (2) microsimulation. The initial routing algorithm developed in MANTA used a parallelized Dijkstra priority queue implementation, which greedily selects the closest vertex that has not yet been processed. This single source shortest path (SSSP) implementation could compute 3M routes, the number of driving trips from 5 AM to 12 PM in the SF Bay Area, in approximately 1 hour.

However, an update was made to integrate an ultra-fast, parallelized Open Source Routing Machine framework using Pandana, based on the contraction hierarchy scheme [16]. The time it takes to compute the same 3M routes in the SF Bay Area is now under one minute.

MANTA is a time-based microsimulator that accounts for parallel changes that occur due to the dependence of cars' behaviors on one another. The vehicular movement on an edge is dictated by conventional car following, lane changing, and gap acceptance algorithms [9, 17]. The well-known Intelligent Driver Model (IDM), as shown in 1, is used to control the vehicle dynamics through the network [18].

$$\dot{v} = a(1 - (\frac{v}{v_o})^{\delta} - (\frac{s_o + Tv + \frac{v\Delta v}{2\sqrt{ab}}}{s})^2) \tag{1}$$

where $\dot{v}$ is the current acceleration of the vehicle, a is the maximum possible acceleration of the vehicle, v is the current speed of the vehicle, v_o is the speed limit of the edge, δ is the acceleration exponent, s is the gap between the vehicle and the leading vehicle, s_0 is the minimum spacing allowed between vehicles when they are at a standstill, T is the desired time headway, and b is the braking deceleration of the vehicle [9, 19]. The exact position of each vehicle at the current timestep is computed using this calculated acceleration value $\dot{v}$.

In addition to car following, vehicles also follow lane changing rules, based on whether the vehicle is making a mandatory or discretionary lane change. Mandatory lane changes occur when the vehicle must take an exit from the edge, while discretionary lane changes occur during overtaking a slower vehicle [9]. The lane changing model states that the vehicle has an exponential probability of switching from a discretionary lane change to a mandatory lane change as it proceeds through the edge, as shown in 2.

$$m_i = \begin{cases} e^{-(x_i - x_0)^2} & x_i \geq x_0 \\ 1 & x_i \leq x_0 \end{cases} \tag{2}$$

where m_i is the probability of a mandatory lane change for vehicle i, x_i is the distance of vehicle i to an exit or intersection, and x_0 is the distance of a critical location to the exit or intersection [20, 21].

Once a vehicle decides to change lanes, the maneuver is performed if the lead and lag gaps of the cars in the lane to which it is changing are acceptable. The critical lead or lag gap for a successful lane change is defined as the minimum distance to the following or lagging vehicle that allows for a lane change, as shown in 3.

$$g_{lead} = \max(g_a, g_a + \alpha_{a1} v_i + \alpha_{a2}(v_i - v_a)) + \epsilon_a \tag{3}$$

$$g_{lag} = \max(g_b, g_b + \alpha_{b1} v_i + \alpha_{b2}(v_i - v_b)) + \epsilon_b \tag{4}$$

where g_{lead} is the critical lead gap for a lane change, g_{lag} is the critical lag gap for a lane change, g_a is the desired lead gap for a lane change, g_b is the desired lag gap for a lane change, α is a system parameter (typically [0.05,0.40]) that controls the gap based on speed, v_i is the speed of the vehicle, v_a is the speed of the lead vehicle, v_b is the speed of the lag vehicle, and ϵ_a and ϵ_b are the random components [9].

MANTA is a parallelized, time-based simulator, rather than an event-based simulator, and thus accounts for changes that occur for all vehicles in every timestep. A time-based approach makes parallelization easier to implement, as synchronization protocols for event-driven simulators is challenging. In addition, no overhead exists with time-based simulators, which contrasts with event-based simulators, where millions to billions of events must be constantly generated. Without this overhead, this allows for the ability to model more granular movements in edges, not just events such as a vehicle entering or exiting an edge, which often constitutes an event. The MANTA architecture is shown in Fig 4.

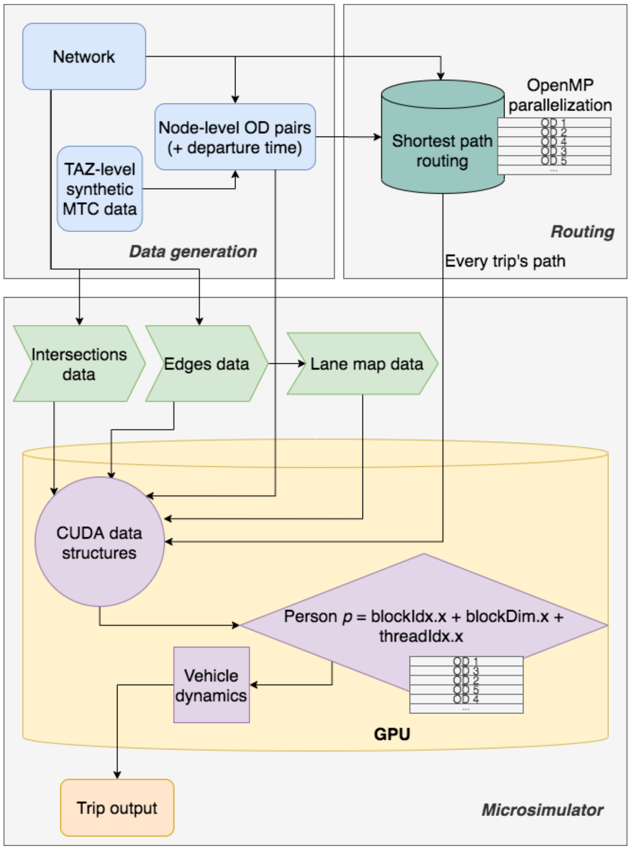


Fig. 4 The MANTA architecture

C. Calibration and Validation

MANTA has been calibrated against Uber Movement data on a per-edge basis for the San Francisco Bay Area. First, 17% of edges in the San Francisco Bay Area network are matched with the Uber Movement data [9]. Then, a mini-batch gradient descent algorithm is used to iteratively converge to the optimal a, b, T, and s_0 parameters of the Intelligent Driver Model, shown in 1. In addition, each driver is outfitted with a different driver profile per edge, based on a Gaussian distribution, with the speed limit of the edge as the mean and the standard deviation of the Uber speeds on the edge as the standard deviation. This produces closely matched edge speeds for the 17% of edges, as shown in Fig. 5 [9]. Average travel times and speeds are then closely validated with California Household Travel Survey (CHTS) data as well as Uber Movement data [9]. As an example, Fig. 5 shows the mean edge speeds across all edges whose speed limits are 45 mph, for both MANTA and Uber Movement. The average speeds differ by 3 mph and the standard deviations by below 1 mph, suggesting successful calibration.

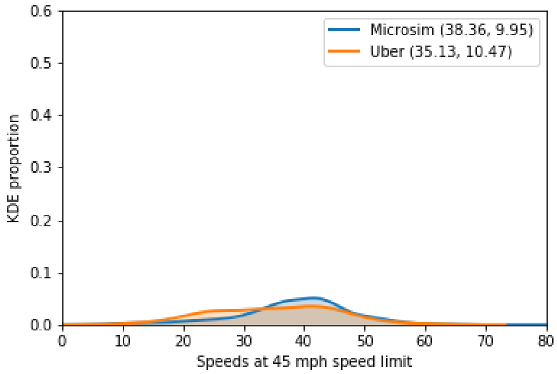


Fig. 5 Kernel density estimator plot comparing MANTA microsim edge speeds vs. Uber edge speeds on edges whose speed limits are 45 mph

From a performance perspective, MANTA can run one full simulation with 3M trips, including routing, in 4.6 minutes on a machine with Intel i9-7940X CPU, 3.10 GHz clock frequency and 28 cores. The GPU that runs the microsimulation component is an NVIDIA GP104 (GeForce GTX 1080) with a 33 MHz clock.

MANTA provides the simulation backbone to the entire SimUAM platform, as all trips, both driving and multi-modal UAM, must originate and end through MANTA. More information can be found in [9]. However, as shown in the next section, multi-modal UAM trips must be modeled by multiple interacting components, one of which is the ground-air interface modeled by VertiSim.

III. VertiSim: Vertiport Ground-Air Interface Modeling

Aircraft need infrastructure to land, taxi-in, park, load and unload passenger/cargo, charge, repair, taxi-out, and take-off. Vertiports have to provide this infrastructure to handle passengers and aircraft within limited land use. Thus, they have limited throughput capacity. This limitation makes vertiports the bottlenecks of the UAM system [22].

Vertiports are the transfer centers for passengers to change from one mode to another. Therefore, transfers at the origin and destination vertiports must be simulated in order to accurately calculate their transfer time. Additionally, aircraft turnaround time and taxiway conflicts must be simulated for accurate throughput calculation.

In order to understand passenger waiting times and aircraft throughput, VertiSim has been developed by the second author. VertiSim is an agent-based discrete-event simulator capable of simulating millions of passengers and aircraft.

A. Inputs

VertiSim takes as input vertiport locations and surface layouts (topologies). These are comprised of TLOF pads, parking spaces, number of charging stations, number of security lines/turnstiles at the vertiport entrance, distributions for stochastic walking times for passengers, passenger arrival schedule with passengers' destination vertiport, aircraft arrival schedule. It also takes several inputs for vertiport surface management strategies.

B. Functionality

The three major components of VertiSim are the (1) vertiport layout designer, (2) flight generator, and (3) vertiport manager. The responsibility of the vertiport layout designer is to create the node-link model of the input file. The flight generator then generates flights, as the number of passengers accumulate to fly to the same direction, and assigns passengers to available aircraft. If there is no available aircraft, the generator puts them in the departure queue. It also generates flights for the passengers who waited more than T_{max}, the maximum waiting time threshold, specified intuitively as 10 minutes, without waiting for the aircraft to be full. When there is a flight queue for the departing passengers in the waiting room, they wait until their turn. The flight generator can assign new passengers to these underbooked flights if passengers who are flying to the same direction arrive to the waiting room before passengers leave the waiting room. The vertiport manager oversees the usage of the resources of the vertiport for the agents. That is, it assigns a parking space and TLOF pad for the aircraft and creates the taxi route for the aircraft. Ultimately, VertiSim is able to simulate the departing passenger simulation, arriving passenger simulation and aircraft simulation at the vertiport surface.

VertiSim is capable of simulating a variety of vertiport topologies. Every vertiport in the given vertiport network can have a unique topology. Departing passenger flow in VertiSim is as follows: (1) car to vertiport entrance, (2) security check/turnstile pass, (3) security check/turnstile to waiting room, (4) waiting room to boarding gate, (5) boarding gate to aircraft, (6) taxi-out, (7) take-off. Arriving passengers then disembark, go to the exit gate at the vertiport surface, exit the vertiport at the ground level, and enter a car. From the perspective of the aircraft, arriving aircraft request an available parking space. If there are no available parking spaces, they loiter until one is reserved. They then land at the TLOF pad assigned by the vertiport manager, taxi-in, unload the passengers, and finally charge. These aircraft then become available for the departing passengers. When a parked aircraft is assigned to a flight, they load passengers, reserve an available TLOF pad, taxi-out, and take-off.

VertiSim outputs aircraft departure schedules, aircraft turnaround time statistics, passenger transfer time statistics, and vertiport efficiency metrics. The VertiSim architecture is shown in Fig. 6.

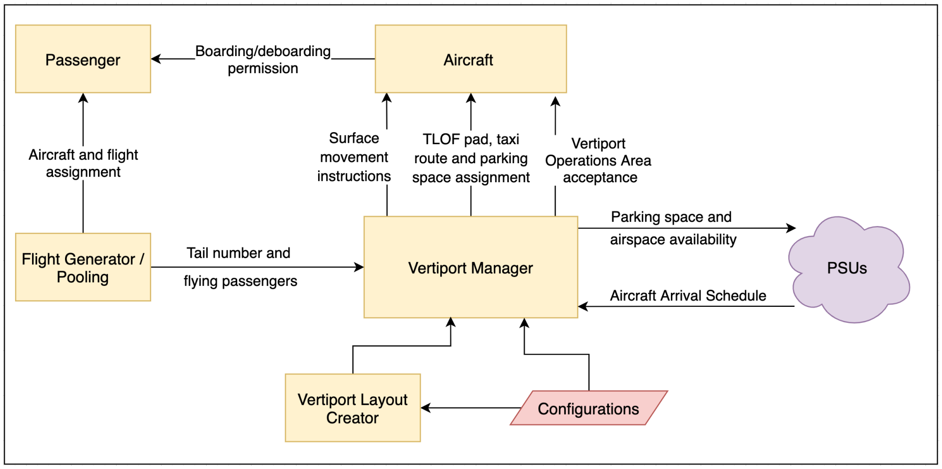

Passenger
Boarding/deboarding permission
Aircraft
Aircraft and flight assignment
Surface movement instructions
TLOF pad, taxi route and parking space assignment
Vertiport Operations Area acceptance
Flight Generator / Pooling
Tail number and flying passengers
Vertiport Manager
Parking space and airspace availability
Aircraft Arrival Schedule
PSUs
Vertiport Layout Creator
Configurations

Fig. 6 The ground/air interface at the vertiport

1. Steady-state data collection

The first passengers have priority to fly when the minimum number of passengers required to fly to the same direction has been met, or if they have waited more than T_{max}. Therefore, the first agents wait less than the other agents. In order to collect accurate data, we remove the data from the transient-state. Although there is no general guideline to identify transient-state from steady-state, several methods exists. The first option is to choose a large enough simulation time to eliminate the effects of the transient state. The second option is to calculate across independent simulation runs the beginning time of the transient-state [23]. We have derived a formulation to detect the beginning of the steady-state region based on our observations.

$$N_A = N_S * N_{TLOF} * 2 \tag{5}$$

where N_A represents number of pseudo-agents need to be added on top of the original demand of each vertiport, N_S is aircraft passenger capacity, and N_{TLOF} represents number of TLOF pads at the vertiport. For instance, on a 4 seat aircraft and 4 TLOF pad scenario, VertiSim creates 32 passengers before the actual demand arrives. Pseudo-agents are randomly sampled from the original demand file and removed at the end of the simulation.

C. Calibration and Validation

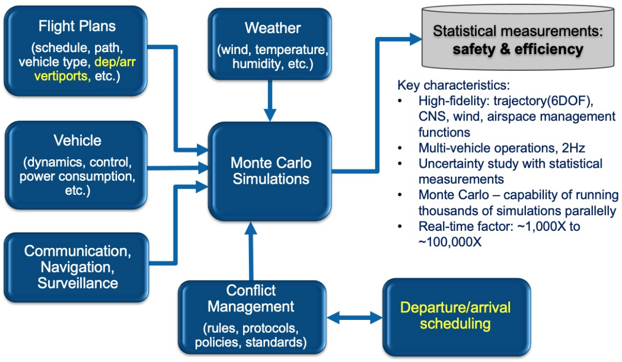
The parameters of VertiSim such as average human walk speed, aircraft ground speed are determined from the literature and by subject-matter experts [24, 25]. Comparing the location-time graphs of the passenger and aircraft agents to these studies confirm the proper parameters.

IV. FE^3: High-fidelity Aerial Simulation

In order to model UAM accurately, we must also simulate aerial component of the trip. Leveraging the FE^3 GPU-based microsimulation platform, developed at NASA, a mission profile is created for every UAM trip, with a departure time, ascent and descent characteristics, cruise altitude, and velocity/directional changes depending on wind patterns and collision avoidance [26]. The FE^3 platform is able to model these in-flight dynamics and any subsequent flight trajectory shifts that contribute to changes in total flight time, making it ideal for simulating UAM in congested airspace [26]. FE^3 is a dynamic, high-fidelity microsimulator leveraging a parallelized GPU architecture, similar to MANTA. Due to the similarities in fidelity, the loose coupling is both seamless and interpretable.

A. Inputs

FE^3 takes several structures as input. As described earlier, MANTA outputs the arrival times of the multi-modal UAM trips to their respective origin vertiports, and VertiSim consolidates these passengers into aircraft based on pedestrian walking times, aircraft arrival times, and vertiport capacities. Once these passengers are assigned aircraft and then board, the aircraft must go through the takeoff and landing process before it departs. As a result, FE^3 takes as input the aircraft departure times output from VertiSim. The passenger IDs are abstracted away from FE^3, which proceeds based on the assumption that all aircraft contain at least 1 passenger. In addition to the aircraft departure times, FE^3 defines simulation related parameters, such as the number of Monte Carlo simulations, temperature and humidity, and the type of avoidance algorithm. It also has the ability to overwrite certain vehicle or device model parameters, as shown in Fig. 7.


Flight Plans
(schedule, path, vehicle type, dep/arr vertiports, etc.)
Weather
(wind, temperature, humidity, etc.)
Statistical measurements:
safety & efficiency
Vehicle
(dynamics, control, power consumption, etc.)
Monte Carlo Simulations
Communication, Navigation, Surveillance
Conflict Management
(rules, protocols, policies, standards)
Departure/arrival scheduling
Key characteristics:
• High-fidelity: trajectory(6DOF), CNS, wind, airspace management functions
• Multi-vehicle operations, 2Hz
• Uncertainty study with statistical measurements
• Monte Carlo – capability of running thousands of simulations parallelly
• Real-time factor: ~1,000X to ~100,000X

*Fe^3 - Flexible engine for Fast-time evaluation of Flight environments

Fig. 7 FE^3 aerial microsimulator architecture

B. Functionality

Fe^3 is highly-parallelized using the CUDA programming language on graphics processing units (GPUs). It is deployed on the Amazon Web Service (AWS) cloud for scalability, such that the number of GPU instances can be dynamically deployed based on simulation needs. Within the simulator, each aircraft's flight in FE^3 is governed by several models, specifically trajectory, energy consumption, collision avoidance, vehicle communication and sensors, and wind models [26].

The trajectory model consists of simulating the vehicle trajectory (in the inertial frame), forces (in the body frame), kinematics (in the body frame), and the moments (in the body frame) [26]. For multi-copters such as VTOL, the forces include: aerodynamic drags caused by vehicle's motion and wind, motor- generated forces that are always in z-direction of the body frame, and the gravity force, which aligns with the z-direction of the inertial frame [26].

The energy consumption model contains two different methods, depending on the fidelity required: (1) simple, in which the power is the product of torque multiplied by the rotational speed, or (2) complex, in which DC motor power consumption is calculated as a product of the input voltage and current. For a given motor, both models show that the consumed energy is primarily a function of the motor's rotational speed, which is driven by the desired thrust and moments [26].

The collision avoidance model takes in the aircraft's position, speed, and intent, and intruder's position, speed, and intent, although some many require information from multiple intruders. FE^3 uses a trajectory projection based method, which is one that predicts the intruder's trajectory based on the intruder's current states (such as position and velocity) and ranks resolutions with predefined maneuver options and rules [26]. Models included are Detect and Avoid Alerting Logic for Unmanned Systems (DAIDALUS) and the Generic Resolution Advisor and Conflict Evaluator (GRACE) [26].

The communication and sensor models are restricted by the vehicle power and size. High transmission power increases the communication channel load and the signal interference, and essentially reduces the signal reception probability [26]. Since the high-density operation leaves a limited space for vehicles to avoid the potential conflicts, and the accuracy and update frequency of the intruder's states greatly affect the conflict avoidance algorithm's performance, the communication and detection capability plays a critical role in the analysis of the UTM-type traffic system [26].

The wind model used by FE^3 is The High Resolution Rapid Refresh (HRRR) system, developed by the National Oceanic and Atmospheric Administration (NOAA) [26]. This provides wind information at low-altitude airspace with great spatial and temporal resolutions. The HRRR wind data cover two altitudes at 10m and 80m [26]. The spatial resolution is 3 km by 3 km and its temporal resolution is 15 minutes [26]. When implemented in FE^3, this model uses a spatially discretized database with turbulence intensity/uncertainty associated with every location [26].

C. Calibration and Validation

Since FE^3's trajectory and energy consumption models are governed by first-principles physical equations, their results are well-understood and reasonable. In addition, FE^3's models for collision avoidance, communication, and wind, such as the DAIDALUS, GRACE, and HRRR, are well-known and fully calibrated and validated [26].

V. Simulation

The SimUAM toolchain is now applied to the San Francisco Bay Area as a case study. This toolchain combines network and origin-destination trip generation, ground traffic microsimulation, vertiport simulation, aerial traffic simulation, and UAM trip assignment and modeling. Once these steps are complete, the final step is to simulate both UAM and non-UAM trips[9, 26].

The first step is a standard MANTA simulation without UAM as a mobility option and with all trips as driving-only trips. These results are used for later comparison.

The next step involves running a MANTA simulation with UAM as a mobility option. In this scenario, there are several sequential steps. The first requires simulating in MANTA the access leg of each multi-modal UAM trip, with the rest of the trips in the demand assigned to driving.

After the toolchain runs, two experiments are presented, (1) S_1, Simulated road traffic in MANTA, and static vertiport transfer times and aerial flight times for the multi-modal UAM trips, and (2) D_1, Simulated road traffic in MANTA, simulated vertiport transfer times in VertiSim, and simulated aerial flight times in FE^3.

In the static experiment, S_1, Once we have these initial times, we then simulate with static transfer and aerial times. A 2 minute transfer time is assumed at the origin vertiport. Then, each traveler is assumed to have access to an aircraft after 2 minutes at the origin vertiport, and they embark on their flight. This flight also has static characteristics, specifically 138 mph (120 kts) cruise speed, 2000 feet cruise altitude, ascent/descent angles of 10 degrees, and 100 fpm vertical climb. The flight path assumes a Haversine distance between the origin and destination vertiports. After the flight lands at the destination vertiport, a 2 minute transfer time at the destination vertiport is used.

In the dynamic experiment, D_1, instead of assuming static transfer and aerial times, travelers are simulated within the vertiport with VertiSim and are consolidated into aircraft. These aircraft's trajectories are then simulated in FE^3. Once the flight lands at the destination vertiport, VertiSim, simulates the traveler's movement from the aircraft to their egress vehicle, as shown in Fig. 1.

In both experiments, once the traveler reaches the egress vehicle, MANTA takes control and simulates the full demand with all access and egress trips for UAM as well as all the non-UAM driving trips.

VI. Performance Metrics

SimUAM's value is derived from its ability to produce a comprehensive set of performance metrics for all trips at both the microscopic person scale and the macroscopic regional scale. Once a network design is input into SimUAM, it is able to simulate all the demand in the region, assigning specific trips to multi-modal UAM and preserving the rest of the demand as driving trips.

In S_1, as mentioned earlier, the transfer times at the vertiports are statically assigned, and the aerial flights follow standard mission profile assumptions that depart on demand for each traveler. We evaluate the number of trips that benefit from taking multi-modal UAM, which is defined as a trip whose travel time when taking multi-modal UAM is shorter than the travel time of the same trip if it was driving only. Shown in Fig. 8, we see that as the transfer times increase, the number of benefited trips decreases across all k. We can also see that the number of benefited trips increases as the number of vertiports increases, with the largest increase occurring between 10 and 30 vertiports, with more marginal benefit occurring as k increases further, ranging between 10K to 250K trips. This serves as a useful upper bound of the total addressable market of urban air mobility.

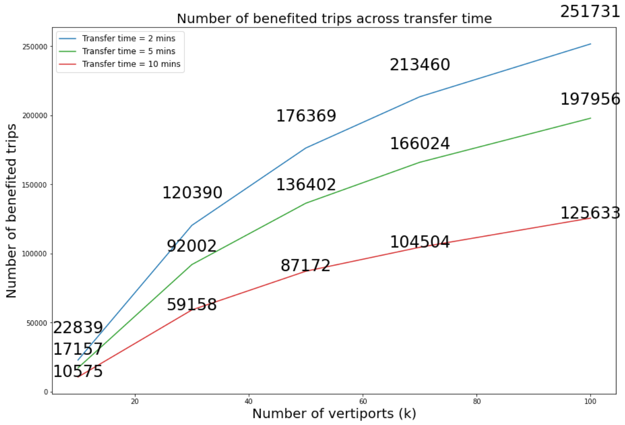


Fig. 8 The number of benefited trips when the ground-air interface is not modeled

A. Regional impact and addressable market

One of the major limitations of static assumptions is the inaccurate representation of aircraft being on-demand (i.e., being available to fly immediately when the traveler requires it). In experiment D_1, we now model the ground-air interface at the vertiport in which the wait times and consolidation times of passengers into aircraft, as well as the aircraft and vertiport capacities. Using the number of benefited from a transfer time of 2 minutes above, we now see after putting that demand through the full SimUAM toolchain that the number of benefited trips at $k = 10$, $k = 30$, $k = 50$, and $k = 70$ decrease significantly. In Fig. **??**, we see that the 23K trips at $k = 10$ to 5.3K trips, 125K trips at $k = 30$ to 11K trips, 171K trips at $k = 50$ to 18K trips, and 212K trips $k = 70$ to 23K trips. This drop in the number of benefited immediately shows the impact of real world constraints that will exist for UAM at the vertiports themselves. Specifically, the number of takeoff and landing areas (TOLAs), number of parking spaces for the aircraft, and the number of passengers per aircraft will dictate the efficiency of multi-modal UAM as a beneficial transportation mode for travelers. $k = 100$ is not calculated due to memory constraints for FE^3.

In Fig. 11, the number of benefited trips in the static S_1 experiment shows a logarithmic growth over k, but once the ground-air interface is modeled in the loosely coupled SimUAM toolchain in D_1, we see that the number of benefited trips increases relatively linearly across k, showing that the waiting times and consolidation times at the vertiports significantly decreases the potential of multi-modal UAM for networks of 30 vertiports or fewer.

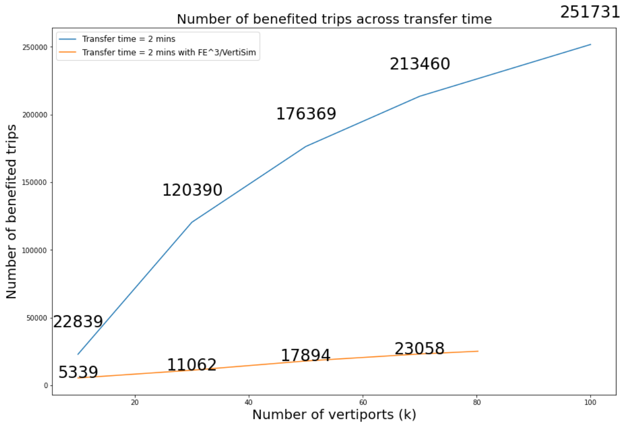


Fig. 10 The number of benefited trips once the ground-air interface is simulated in SimUAM significantly decreases across all k

The trips that benefit also change. Specifically, when considering the ground-air interface, the trips that benefit are those that have even higher travel times than when we did not consider the ground-air interface. For instance, at $k = 10$, the average driving time of trips that benefit when not considering the ground-air interface is 94 minutes, but when considering the ground-air interface, this increases to 101 minutes. The multi-modal UAM times for these same trips also increase when considering the ground-air interface, going from 64 minutes to 77 minutes. This is unsurprising given the increase in transfer times due to the ground-air interface bottleneck. This dynamic across k is shown in Fig. **??**.

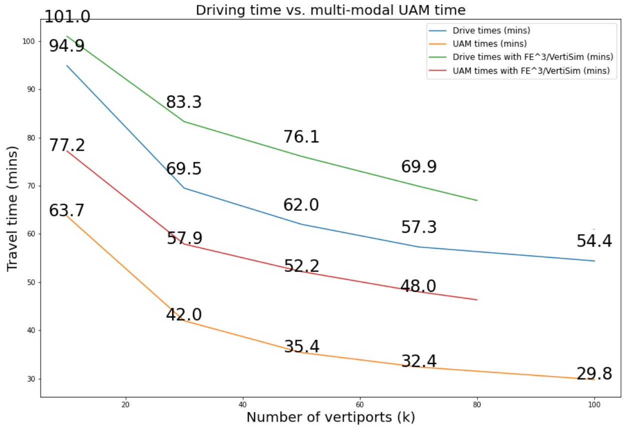


Fig. 11 The average travel time of benefited trips once the ground-air interface is simulated in SimUAM significantly increases across all k (red vs. orange)

B. Vertiports

Per the existing literature, we assume a throughput of 240 flights per hour, including both departures and arrivals [26]. At $k = 10$, the average transfer time once the ground-air interface is considered dramatically increases to 49 minutes, as shown in Fig. 12. With a configuration of 4 TLOF pads, 16 parking spaces, and up to 4 seater aircraft, using the initial total demand of 23K at a transfer time of 2 minutes, we see that the real transfer times vary considerably across vertiports. The maximum average waiting time occurs at Vertiport 6, whose traffic of over 4K passengers forces the average transfer time to be over 2 hours. At Vertiport 4, since the demand is below 1K, the average transfer time is only 3.85 minutes.

This shows that demand balancing may be important to decrease average waiting times at each vertiport. However, balancing the demand in a network with few vertiports may also increase the access and egress driving times of these multi-modal UAM trips.

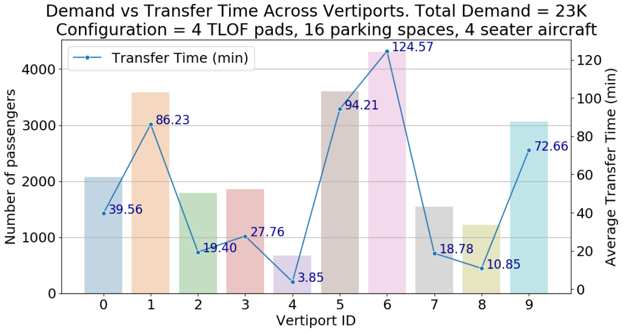


Fig. 12 The average transfer time at $k = 10$ once the ground-air interface is considered

To investigate the effect of demand level on passenger transfer times, we have created 7 simulation scenarios with the vertiport configuration of 4 TLOF pads and 16 parking spaces per vertiport. 30 seconds of aircraft arrival per vertiport (arrival of 2 minutes per TLOF) is used for all cases. We use the same total demand, but now increase the number of vertiports in the UAM network. Thus, the demand per vertiport decreases. Fig. 13 shows that as we decrease the demand, the average transfer time decreases down to an inflection point. The decrease in the average transfer time is due to the increased passenger inter-arrival time, which results in shorter queues for the flights. Decreasing the demand further actually results in an increase in the average transfer time because the accumulation time for the required number of passengers for a flight (4 passengers) increases. We also see that as we decrease the demand, the average aircraft occupancy decreases, due to the ratio of the passengers that wait more than T_{max} (10 minutes) increases and the flight generator is triggered for those passengers. Ultimately, waiting times decrease as k increases, but at 30 vertiports, the consolidation time increases as well, due to the demand being distributed across all vertiports rather than at certain vertiports.

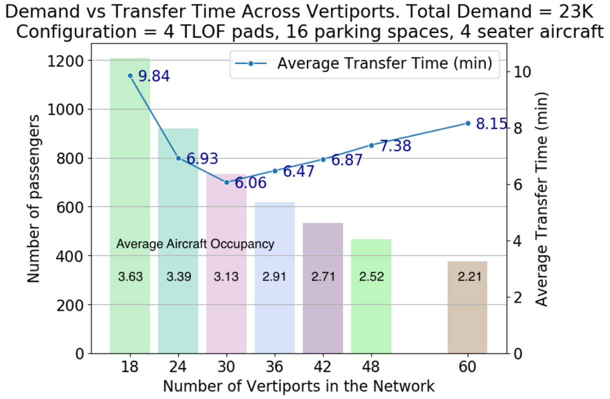


Fig. 13 The average transfer time and average aircraft occupancy with different demand levels

C. Airspace

Once we simulate the aircraft in FE^3, we see that there are now many aircraft congesting the airspace, with 7,352 flights for $k = 30$ vertiports, shown in Fig. 14. Note that these aircraft trajectories do not consider airspace restrictions, which will further affect the number of benefited trips if the airspace-restricted flight times are higher than the airspace-unrestricted flight times. FE^3 finds that with federated scheduling and no conflict resolution, the average delay is 21.5 seconds per flight.

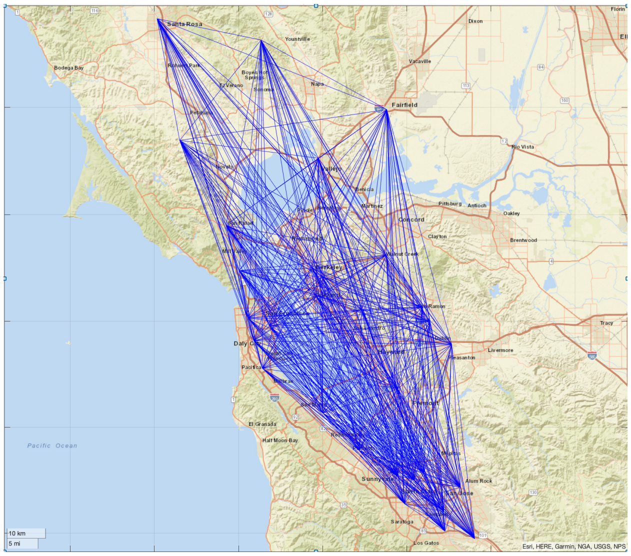


Fig. 14 The congested airspace of the San Francisco Bay Area at $k = 30$ vertiports

VII. Limitations and Future Research

The SimUAM toolchain has one notable limitation: the inability to do iterative convergence across a loosely coupled pipeline [**?**]. Due to the complex data input and output of iterative convergence, an ongoing solution is to cross-compile both GPU-based microsimulators, MANTA and FE^3, and convert VertiSim into a GPU-based simulator. By synchronizing the time, the convergence issue will be eliminated.

Due to the long transfer times at the vertiports, future research should involve optimal network design to incorporate balanced vertiport networks. In addition, incorporating waiting time and consolidation time into the network design will be able to decrease the average transfer times across vertiports. Additional work involves refining the current components and adding new components. MANTA is currently being enhanced to carry out dynamic assignment, such that vehicles can adjust their routes based on the current congestion. FE^3 is currently being refactored to account for larger demand scales and vertiport networks, such as at $k = 100$, and also to consider local airspace restrictions. In addition, VertiSim is being restructured as a parallel programming architecture.

VIII. Conclusion

This paper proposes a high-level multi-component architecture to integrate and model the effects of introducing an emerging mobility technology, urban air mobility, into the transportation network. Scholarship hitherto has tackled different components of this pipeline independently; however, in this research, we develop a full toolchain, SimUAM, to model the ground traffic, vertiport, and aerial traffic with granular, agent-based, high-fidelity simulation. Initial results show that once the ground-air interface is modeled, the market for UAM decreases across all network designs relative to models with static assumptions about transfer times. However, we also find that improvements can be made to balance the demand and optimize the networks for transfer time, likely increasing the number of benefited trips closer to the total addressable market once more. SimUAM is the first of its kind to incorporate fast, high-fidelity, regional-scale microsimulation on the ground and in the air with detailed ground-air interface modeling at the vertiport, and can be used as a powerful decision support system for UAM stakeholders.

References

[1] Giuliano, G., and Hanson, S., *The Geography of Urban Transportation*, ????

[2] Stopher, P., and Stanley, J., *Introduction to Transport Policy: A Public Policy View*, ????

[3] Garrow, L. A., Ilbeigi, M., and Chen, Z., "Forecasting Demand for On Demand Mobility," *17th AIAA Aviation Technology,*

Integration, and Operations Conference, AIAA AVIATION Forum, American Institute of Aeronautics and Astronautics, ???? https://doi.org/10.2514/6.2017-3280, URL https://arc.aiaa.org/doi/10.2514/6.2017-3280.

[4] Rothfeld, R., Fu, M., Balac, M., and Antoniou, C., "Potential Urban Air Mobility Travel Time Savings: An Exploratory Analysis of Munich, Paris, and San Francisco," ????

[5] Urban Air Mobility, N., "Uam-Market-Study-Executive-Summary-Pr.Pdf," , ???? URL https://www.nasa.gov/sites/default/files/atoms/files/uam-market-study-executive-summary-pr.pdf.

[6] Bulusu, V., Burak Onat, E., Yedavalli, P., Sengupta, R., and Macfarlane, J., "A Traffic Demand Analysis Method for Urban Air Mobility," ????

[7] Yedavalli, P., "Designing and Simulating Urban Air Mobility Vertiport Networks under Land Use Constraints," ????

[8] Lim, E., and Hwang, H., "The Selection of Vertiport Location for On-Demand Mobility and Its Application to Seoul Metro Area," Vol. 20, No. 1, ????, pp. 260–272. https://doi.org/10.1007/s42405-018-0117-0, URL http://link.springer.com/10.1007/s42405-018-0117-0.

[9] Yedavalli, P., Kumar, K., and Waddell, P., "Microsimulation Analysis for Network Traffic Assignment (MANTA) at Metropolitan-Scale for Agile Transportation Planning," ????, pp. 1–22. https://doi.org/10.1080/23249935.2021.1936281, URL https://www.tandfonline.com/doi/full/10.1080/23249935.2021.1936281.

[10] Axhausen, K. W., and ETH Zürich, *The Multi-Agent Transport Simulation MATSim*, Ubiquity Press, ???? https://doi.org/10.5334/baw, URL http://www.ubiquitypress.com/site/books/10.5334/baw/.

[11] Chan, C., Wang, B., Bachan, J., and Macfarlane, J., "Mobiliti: Scalable Transportation Simulation Using High-Performance Parallel Computing," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, ????, pp. 634–641. https://doi.org/10.1109/ITSC.2018.8569397, URL https://ieeexplore.ieee.org/document/8569397/.

[12] Park, B. B., and Schneeberger, J. D., "Microscopic Simulation Model Calibration and Validation: Case Study of VISSIM Simulation Model for a Coordinated Actuated Signal System," Vol. 1856, No. 1, ????, pp. 185–192. https://doi.org/10.3141/1856-20, URL http://journals.sagepub.com/doi/10.3141/1856-20.

[13] Rasouli, S., and Timmermans, H., "Activity-Based Models of Travel Demand: Promises, Progress and Prospects," Vol. 18, No. 1, ????, pp. 31–60. https://doi.org/10.1080/12265934.2013.835118, URL https://doi.org/10.1080/12265934.2013.835118.

[14] Boeing, G., "OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks," Vol. 65, ????, pp. 126–139. https://doi.org/10.1016/j.compenvurbsys.2017.05.004, URL https://linkinghub.elsevier.com/retrieve/pii/S0198971516303970.

[15] Metropolitan Transportation Commission, and Association of Bay Area Governments, "Plan Bay Area 2040," , ???? URL http://2040.planbayarea.org/files/2020-02/Travel_Modeling_PBA2040_Supplemental%20Report_7-2017.pdf.

[16] Foti, F., and Waddell, P., "A Generalized Computational Framework for Accessibility: From the Pedestrian to the Metropolitan Scale," ????, p. 14.

[17] Toledo, T., Koutsopoulos, H., Ben-Akiva, M., and Jha, M., "Microscopic Traffic Simulation: Models and Application," *Simulation Approaches in Transportation Analysis*, Springer-Verlag, ????, pp. 99–130. https://doi.org/10.1007/0-387-24109-4_4, URL http://link.springer.com/10.1007/0-387-24109-4_4.

[18] Treiber, M., and Kesting, A., *Traffic Flow Dynamics: Data, Models and Simulation*, Springer Berlin Heidelberg, ????

[19] Waddell, P., Boeing, G., Gardner, M., and Porter, E., "An Integrated Pipeline Architecture for Modeling Urban Land Use, Travel Demand, and Traffic Assignment," , ???? URL http://arxiv.org/abs/1802.09335.

[20] Iqbal, M. S., Choudhury, C. F., Wang, P., and González, M. C., "Development of Origin–Destination Matrices Using Mobile Phone Call Data," Vol. 40, ????, pp. 63–74. https://doi.org/10.1016/j.trc.2014.01.002, URL http://www.sciencedirect.com/science/article/pii/S0968090X14000059.

[21] Yang, Q., and Koutsopoulos, H. N., "A Microscopic Traffic Simulator for Evaluation of Dynamic Traffic Management Systems," Vol. 4, No. 3, ????, pp. 113–129. https://doi.org/10.1016/S0968-090X(96)00006-X, URL https://www.sciencedirect.com/science/article/pii/S0968090X9600006X.

[22] Vascik, P. D., and Hansman, R. J., "Development of Vertiport Capacity Envelopes and Analysis of Their Sensitivity to Topological and Operational Factors," *AIAA Scitech 2019 Forum*, ????, p. 0526.

[23] Bose, S., *An Introduction to Queuing Systems*, ???? https://doi.org/10.1007/978-1-4615-0001-8.

[24] Levine, R. V., and Norenzayan, A., "The Pace of Life in 31 Countries," Vol. 30, No. 2, ????, pp. 178–205. https://doi.org/10.1177/0022022199030002003, URL https://doi.org/10.1177/0022022199030002003.

[25] Zelinski, S., "Operational Analysis of Vertiport Surface Topology," ????, pp. 1–10. https://doi.org/10.1109/DASC50938.2020.9256794.

[26] Xue, M., Rios, J., Silva, J., Zhu, Z., and Ishihara, A. K., "Fe ^{3} : An Evaluation Tool for Low-Altitude Air Traffic Operations," *2018 Aviation Technology, Integration, and Operations Conference*, American Institute of Aeronautics and Astronautics, ???? https://doi.org/10.2514/6.2018-3848, URL https://arc.aiaa.org/doi/10.2514/6.2018-3848.