Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

OUT OF THE PAST: AN AI-ENABLED PIPELINE FOR TRAFFIC SIMULATION FROM NOISY, MULTIMODAL DETECTOR DATA AND STAKEHOLDER FEEDBACK

Rex Chen[1], Karen Wu[1], John McCartney[2], Norman Sadeh[1], and Fei Fang[1]

[1]School of Computer Science, Carnegie Mellon University, Pittsburgh, UNITED STATES
[2]Path Master Inc., Twinsburg, UNITED STATES

ABSTRACT

How can a traffic simulation be designed to faithfully reflect real-world traffic conditions? One crucial step is modeling the volume of traffic demand. But past demand modeling approaches have relied on unrealistic or suboptimal heuristics, and they have failed to adequately account for the effects of noisy and multimodal data on simulation outcomes. In this work, we integrate advances in AI to construct a three-step, end-to-end pipeline for systematically modeling traffic demand from detector data: computer vision for vehicle counting from noisy camera footage, combinatorial optimization for vehicle route generation from multimodal data, and large language models for iterative simulation refinement from natural language feedback. Using a road network from Strongsville, Ohio as a testbed, we show that our pipeline accurately captures the city's traffic patterns in a granular simulation. Beyond Strongsville, incorporating noise and multimodality makes our framework generalizable to municipalities with different levels of data and infrastructure availability.

1 INTRODUCTION

Traffic simulation is an important tool in transportation research: both for performing traffic analysis on experimental substitutes of real-world transportation systems (Barceló 2010), and for training and evaluating intelligent transportation systems, e.g., those based on reinforcement learning (Mei et al. 2024). If the results of traffic simulations are to be deployed in real-world transportation systems, they must be sufficiently realistic to foster trust from stakeholders. Although there is a significant body of work on constructing efficient *simulators* for executing simulations (Zhang et al. 2019; Chen et al. 2023), an understudied problem is the construction of realistic *simulations* grounded in data from physical traffic systems.

Existing approaches to creating road network-scale traffic simulations have a number of limitations that hamper their realism and thus their practical applicability. We focus on limitations surrounding the central task of *demand modeling*, or the modeling of traffic volumes within the simulation. Demand modeling methods that construct origin-destination matrices from activity data are unrealistic and fail to make use of traffic detector data. Meanwhile, detector data-driven approaches to demand modeling have relied on suboptimal heuristics. All of these approaches also consider the source data to be the ground truth; they do not perform any calibration to account for sources of noise or multimodality in the data.

In this work, we contribute a detailed, systematic pipeline for modeling demand in a traffic simulation from noisy, multimodal detector data (Figure 1). Starting from raw detector data, our pipeline consists of three steps: (1) We apply a vehicle tracking-based computer vision method directly to camera footage, to obtain more accurate vehicle counts than the camera detectors themselves. (2) We solve a quadratic optimization program to populate our simulation with vehicle routes. In doing so, we account for multimodality by imposing multiple sets of optimization constraints based on different sources of vehicle counts. (3) We incorporate feedback from stakeholders to refine the simulation, using a large language model (LLM) agent that encodes natural language feedback into optimization constraints.

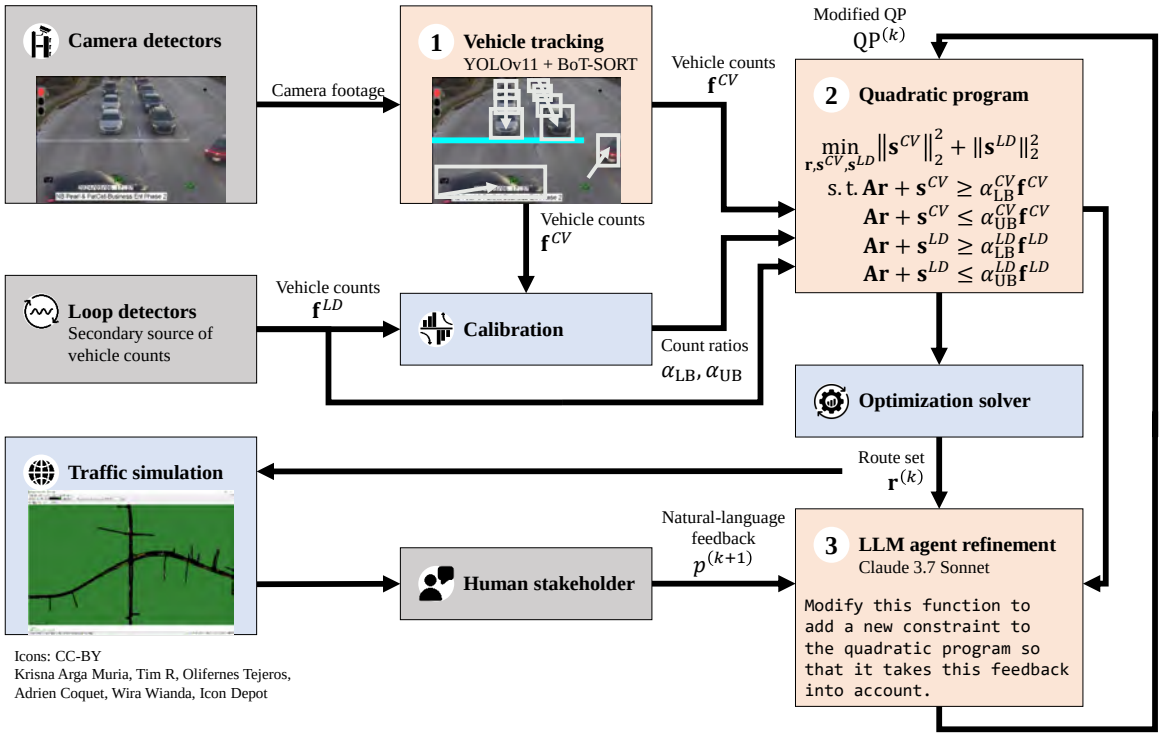

Figure 1: Our pipeline for generating a traffic simulation from multimodal detector data.

As a proof of concept, we apply our pipeline to simulate a high-traffic road network from the city of Strongsville, Ohio. Beginning with 24 hours of recorded camera footage and detector data from 36 intersections, we created a fully realized traffic simulation. We show that: (1) Our vehicle tracking-based computer vision method rectifies undercounting in camera detector data. (2) Our optimization method is able to generate a set of vehicle routes that is consistent with counts from both computer vision and loop detector counts, while still accounting for error in these counts. (3) Our LLM agent is able to synthesize code representing sensible, quantified constraints based on qualitative stakeholder feedback. Our pipeline code and LLM prompts are available at https://github.com/lythronaxargestes/strongsville-trafficsim-public.

2 RELATED WORK

2.1 Demand Modeling for Traffic Simulation

In this work, we focus not on the design of traffic simulators in general, but rather on the construction of *traffic simulations* within a given simulator so as to represent traffic conditions within a specific road network. *Demand modeling*, or the modeling of trips taken by individual vehicles from one point to another in a road network, is a central but difficult aspect of constructing data-driven traffic simulations (Barceló 2010). One popular approach is *activity modeling*, where trips are extrapolated from censuses of the daily activities taken by a sample of households in the study region (Uppoor and Fiore 2012; Codeca et al. 2015; Leon et al. 2023). However, these censuses represent coarsely discretized and not necessarily representative samples of the population. This means that activity modeling-based simulations are prone to significant error (Kwak et al. 2012); thus, we do not consider activity modeling-based approaches in this work.

An alternative approach to demand modeling directly uses data from traffic detectors. Detectors provide granular vehicle counts local to individual intersections, but converting them to fully-realized routes through a road network is nontrivial. Route generation procedures in prior work have relied on suboptimal heuristics.

Lobo et al. (2020) and Rapelli et al. (2022) used detector data to adjust activity models. Bieker et al. (2014) used counts to probabilistically assign traffic to turning movements, which only leads to correct simulation outcomes in expectation. Qiu et al. (2024)'s approach, which uses scripts included with the traffic simulator SUMO (Alvarez Lopez et al. 2018), is most similar to ours: they applied a two-step process of first sampling routes randomly, and then solving a linear program (LP) to approximate how many times each route should be used to match the detector counts as closely as possible. Unlike them, we solve the problem exactly as a quadratic integer program (QIP) without making intermediate approximations.

2.2 Computer Vision for Traffic Footage

Vehicle counting can be decomposed into two distinct but related problems: *vehicle detection*, the identification of vehicles in footage; and *vehicle tracking*, the identification of these vehicles' trajectories across frames (Tituana et al. 2022). These methods can be divided into two distinct waves of research. First, in the 1990s, advances in image processing led to vehicle detection methods based on extracting heuristically designed features (Coifman et al. 1998). Second, in the 2010s, the advent of *convolutional neural networks* (CNNs) led to various deep object detection algorithms capable of automatically extracting relevant features for vehicle detection (Wang et al. 2019). While these lines of work have more recently overlapped methodologically, they have not been comparatively evaluated to our knowledge. We provide an in-situ evaluation of AutoScope, a widely-deployed image processing method, against CNN-based counting.

2.3 Large Language Models for Transportation Research

Large language models (LLMs) are useful for aligning AI systems with human intuition. As such, they have been increasingly applied to the domain of transportation, e.g. to generate simulation scenarios (Chang et al. 2024; Li et al. 2024) and reward functions for vehicular agents (Han et al. 2024) based on natural language prompts. We leverage a strength of LLMs that has not been explored for transportation research to our knowledge: the synthesis of syntactically and semantically correct programs (Austin et al. 2021).

3 TRAFFIC DETECTION IN PRACTICE

Modern traffic systems make use of different types of detectors, which have varying strengths and limitations. To build a realistic traffic simulation from detector data, it is important to understand the primary use case of the data and the circumstances under which it may diverge from the ground truth. We consider two types of detectors that are commonly used in modern traffic systems.

Camera Detectors A camera detector is typically mounted in a fixed position above the roadway, and detection zones are placed on the camera's field of view. In the United States, the AutoScope vehicle detection algorithm (Michalopoulos 1991) is used for many cameras. It extracts features to label each detection zone as being in one of three discrete states: "background", "uncertain", and "vehicle". While this algorithm is able to generate vehicle counts, the counts do not reflect the actual vehicle volume — they are the number of times each zone was in the "vehicle" state. In high-volume traffic, consecutive vehicles may continuously actuate a detection zone (Figure 2a), leading to undercounting. This is more significant of an issue for traffic simulation than for the detectors' primary use in traffic signal control.

Inclement environmental conditions also contribute to inaccuracy in camera detector counts. Darkness (due to nighttime or fog) and precipitation (such as rain or snow, which cause glare) obfuscate the visual signal of vehicles, thus making it more difficult for vehicle detection algorithms to isolate them from the background (Medina et al. 2010). Shadows can also result in false detections (Rhodes et al. 2005).

Loop Detectors An induction loop detector consists of a loop of wire embedded in the pavement, which is actuated when a vehicle passes over it. Loop detectors are more robust to the environment than cameras. However, actuation depends on the detector's sensitivity, which is hard to configure accurately and may result in undercounting or overcounting. Overcounting can also occur due to excessive sensitivity to adjacent lanes (splashover) or detector interference (chattering) (Lee and Coifman 2012).

4 DEMAND MODELING PIPELINE

4.1 Computer Vision-Based Vehicle Counting from Camera Footage

To detect individual vehicles more accurately than the camera detectors themselves, we use the YOLOv11 object detection model (Jocher and Qiu 2024). In each frame of footage, the model predicts a set of bounding boxes, each of which encloses an object. For each box, the model outputs a class b_c, the x and y coordinates of the center of the bounding box (b_x and b_y), and the width and height of the box (b_w and b_h).

How can YOLO's detections of vehicles be converted to counts for each lane? We manually annotate each frame of the traffic footage with the stop bar's y-coordinate (S_y), and with x-coordinates for each lane (S_L^ℓ, S_R^ℓ). The most straightforward method to perform counting is to verify whether the bounding box for an object identified as a vehicle has crossed the position of the stop bar, i.e., we increment the count for lane ℓ when a detected vehicle has $S_L^\ell \leq b_x \leq S_R^\ell$, and $|b_y - S_y| \leq \varepsilon$ for some predefined threshold ε.

When we apply this method directly in practice, we encounter two issues. (1) Due to instability in the real-time streaming (RTSP) connection over which detector footage is retrieved, frames are frequently dropped. For many vehicles, this leads to the absence of the frames in which their bounding boxes' borders b_y are close to the stop bar S_y. These vehicles first appear far above the stop bar ($b_y \ll S_y$), and then far below the stop bar ($b_y \gg S_y$) once the footage resumes. (2) Pixelation artifacts, particularly around the detection zones marked on the footage, also obfuscate the bounding boxes.

To address dropped frames, we use the BoT-SORT algorithm (Aharon et al. 2022) to perform vehicle tracking. BoT-SORT reidentifies each bounding box across consecutive frames to provide a consistent ID b_t. In each frame, for each bounding box, we verify whether b_t has already been counted. To ensure that we do not capture traffic in other directions, we only consider b_t if $b_y < S_y$ initially. If b_t has not yet been counted and $b_y > S_y$, we increment the vehicle count and mark b_t as counted. Even if the footage is missing the frame where $|b_y - S_y| \leq \varepsilon$ for a vehicle, the vehicle will be counted in a later frame. We also apply filters to smooth the footage as a preprocessing step, including a non-local means filter (Buades et al. 2011), a spatiotemporal denoising filter (`hqdn3d`), a frame-blending motion interpolation filter (`minterpolate`), and a filter to remove detector actuation overlays. The results are shown in Figure 2b.

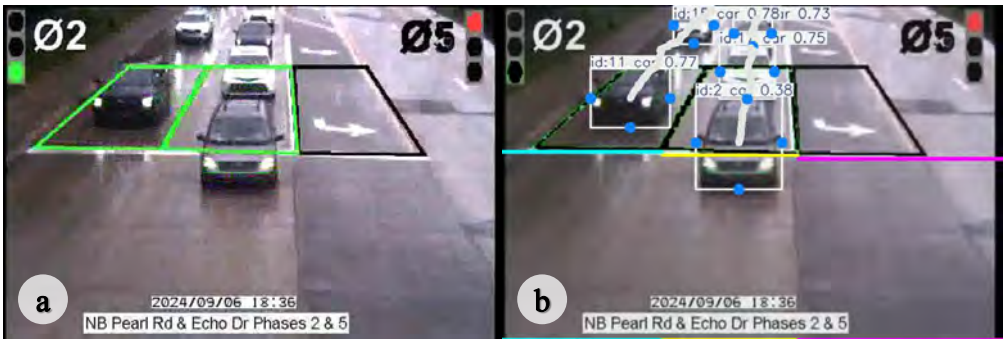

Figure 2: Demonstration of our vehicle tracking method on camera detector footage from intersection 8 (US 42 & Echo Rd) in Strongsville, Ohio. (a) Raw footage, showing two vehicles actuating a detection zone in the center lane. (b) Footage with preprocessing filters applied, and bounding boxes and tracks for counted vehicles annotated. The colored lines represent manually labeled stop bar positions.

4.2 Optimization-Based Vehicle Route Generation from Multimodal Data

Our vehicle tracking-based computer vision method from Section 4.1 outputs vehicle counts f_j^{CV} for a set of counting locations j. These represent the traffic volumes at the eastbound, northbound, southbound, and westbound approaches for each intersection (if they are available). We also have a set of counts from loop

detectors f_j^{LD}, which overlap with the computer vision counts at a subset of counting locations. But how can these multimodal vehicle counts be integrated to generate vehicle routes for a fully-realized simulation?

We make two key assumptions to identify the set of feasible routes. (1) Given an origin and destination in the road network, we assume that vehicles perform shortest-path (Dijkstra) routing. (2) We assume that most of the traffic in the road network originates at the fringes of the network, and few routes begin and end in the middle of a road edge (representing traffic from unmodeled driveways). This assumption holds as long as all major sources and sinks of traffic are modeled. Based on these assumptions, we enumerate the full set of routes between all counting locations, instead of randomly sampling them as in prior work.

Given this set of feasible routes, we aim to solve for the number of times each route should be used, so that the number of times they pass through the counting locations match the given vehicle counts as closely as possible. We do so by dividing 24 hours of count data into 15-minute time segments, which are indexed as $t \in \{0..95\}$. For each time segment, we match the total counts at each counting location, as well as counts for dedicated left-turn and right-turn lanes if they exist.

How closely should we match the counts? We assume that, for some counting locations j and time segments t, we have ground truth counts f_{jt}^{M}, and that error exists in both our computer vision counts f_{jt}^{CV} and loop detector counts f_{jt}^{LD}. Let M, CV, and LD denote the sets of counting locations for these three sources. Based on how much computer vision overcounts or undercounts for locations in common with the ground truth, we extrapolate conservative lower and upper bounds for the true counts:

$$\alpha_{\text{LB}}^{CV} = \min_{t} \min_{j \in M \cap CV} \frac{f_{jt}^{M}}{f_{jt}^{CV}}, \alpha_{\text{UB}}^{CV} = \max_{t} \max_{j \in M \cap CV} \frac{f_{jt}^{M}}{f_{jt}^{CV}}.$$

We also derive bounds for loop detectors, α_{LB}^{LD} and α_{UB}^{LD}, in a similar fashion. Then, we assume that, for time segments $t \in \{0..95\}$, $\alpha_{LB}^{CV} f_{jt}^{CV} \leq f_{jt}^{M} \leq \alpha_{UB}^{CV} f_{jt}^{CV}, \forall j \in CV$, and $\alpha_{LB}^{LD} f_{jt}^{LD} \leq f_{jt}^{M} \leq \alpha_{UB}^{LD} f_{jt}^{LD}, \forall j \in LD$.

Now, for each 15-minute time segment t, we used the solver Gurobi to solve the following quadratic integer program (QIP), where the decision variable is the number of usages r_{it} for each route $i \in \{1..n\}$:

$$\min_{\mathbf{r}_t, \mathbf{s}_t^{CV}, \mathbf{s}_t^{LD}} \|\mathbf{s}_t^{CV}\|_2^2 + \|\mathbf{s}_t^{LD}\|_2^2 + \lambda \sum_{i \in \text{nonfringe}} r_{it} \tag{1a}$$

$$\text{s.t. } \mathbf{A}\mathbf{r}_t + \mathbf{s}_t^{CV} \geq \alpha_{LB}^{CV} \mathbf{f}_t^{CV} \tag{1b}$$

$$\mathbf{A}\mathbf{r}_t + \mathbf{s}_t^{CV} \leq \alpha_{UB}^{CV} \mathbf{f}_t^{CV} \tag{1c}$$

$$\mathbf{A}\mathbf{r}_t + \mathbf{s}_t^{LD} \geq \alpha_{LB}^{LD} \mathbf{f}_t^{LD} \tag{1d}$$

$$\mathbf{A}\mathbf{r}_t + \mathbf{s}_t^{LD} \leq \alpha_{UB}^{LD} \mathbf{f}_t^{LD} \tag{1e}$$

$$\mathbf{r}_t \in (\mathbb{Z}^{\geq 0})^n.$$

Here, $\mathbf{A} \in \{0,1\}^{n \times m}$ is a binary matrix denoting which counting locations are used by routes: A_{ij} is 1 if route i passes counting location j, and is 0 otherwise, such that $\mathbf{A}\mathbf{r}_t$ gives the number of times the generated routes collectively pass each counting location $j \in \{1..m\}$; $\mathbf{s}_t^{CV}, \mathbf{s}_t^{LD} \in \mathbb{R}^m$ are slack variables that represent the error between the generated routes' counts $\mathbf{A}\mathbf{r}_t$ and the actual counts $\mathbf{f}_t^{CV}$ and $\mathbf{f}_t^{LD}$; nonfringe is the set of indices for routes where the start or the end of the route are interior edges in the road network; and λ is a hyperparameter for weighting the objective function penalty for these routes.

In the QIP, constraints (1b) and (1c) specify that $\mathbf{A}\mathbf{r}_t$ should lie within a probable range of counts extrapolated from computer vision counts. The lower bound $\alpha_{LB}^{CV} \mathbf{f}_t^{CV}$ assumes that computer vision is overcounting, and the upper bound $\alpha_{UB}^{CV} \mathbf{f}_t^{CV}$ assumes that it is undercounting. The sum-of-squares of the error $\|\mathbf{s}_t^{CV}\|_2^2$ is minimized in the objective function (1a). The two following constraints, (1d) and (1e), are analogous constraints for loop detector counts. Again, the error $\|\mathbf{s}_t^{LD}\|_2^2$ is minimized in the objective. Finally, the objective minimizes the number of nonfringe routes, which are rare under our assumptions.

Once we have obtained the solution $\mathbf{r}$, we assume that the set of routes corresponding to this solution are uniformly distributed within the 15-minute time segment indexed as t.

4.3 LLM Agent Simulation Refinement from Natural Language Feedback

QIP (1) is fundamentally underconstrained. For each counting location, the generated counts could lie anywhere between the lower bounds (constraints (1b) and (1d)) and the upper bounds (constraints (1c) and (1e)). Additionally, most municipalities do not install camera detectors at every intersection, meaning that our vehicle tracking method (Section 4.1) does not generate counts or impose bounds for the entire road network. Not all possible ways of assigning routes to match these counts are equally realistic. We can leverage the domain knowledge of stakeholders, such as traffic engineers, to ensure that the traffic simulation is aligned with downstream use cases such as traffic analysis. Yet, without experience in optimization, it is difficult for these stakeholders to directly modify QIP (1) to align with their intuition.

Our problem formulation is as follows. We are given K pieces of structured natural language feedback $p^{(k)} = (t^{(k)}, j^{(k)}, l^{(k)})$, where each piece consists of a time, intersection, and a natural language description of what corrections (if any) should be made to the simulated traffic state at this intersection. We are also given code which solves the original problem $\text{QP}^{(0)}$, and the route counts $\mathbf{Ar}^{(0)}$ obtained from solving $\text{QP}^{(0)}$. The objective is to produce an updated problem $\text{QP}^{(K)}$, which has been modified so that it will produce a new route set $\mathbf{r}^{(K)}$ that addresses $\{p^{(1)}, \ldots, p^{(K)}\}$. The core difficulty in this problem is converting the natural language feedback into concrete optimization constraints, which cannot be accomplished by traditional optimization methods. Instead, we solve this problem by using an LLM agent to answer prompts containing $(p^{(k)}, \text{QP}^{(k-1)}, \mathbf{Ar}^{(k-1)})$, and leveraging its code generation capabilities to generate $\text{QP}^{(k)}$.

Notably, we do not provide the LLM agent with any handcrafted information beyond the time segment $t^{(k)}$ and intersection $j^{(k)}$ that the feedback is targeted at; what is already available from $\text{QP}^{(k)}$; and a list of intersections and main roads. Based on the set of route counts $\mathbf{Ar}^{(k-1)}$ from the previous simulation, the LLM agent must automatically extract concrete, quantitative constraints that are aligned with the qualitative feedback. To solve this task, we prompt the LLM agent using a chain of thought (Wei et al. 2022) to:

(1) Extract the relevant counts by formulating a call to a `get_counts` tool, which retrieves the previous counts $\mathbf{Ar}_{jt}^{(k-1)}$ for a particular counting location and time segment (j,t);
(2) Write a constraint corresponding to the feedback $p^{(k)}$, using the counts from the previous step to make subjective judgments on how to set undetermined coefficients;
(3) Translate this constraint to Python code for the package `cvxpy` (Diamond and Boyd 2016);
(4) For the time segment $t^{(k)}$ specified in the feedback, add this constraint to the optimization function while minimally modifying the rest of the code;
(5) For adjacent time segments $(t^{(k)}-1, t^{(k)}+1)$, add relaxed constraints to ensure temporal continuity.

We use this LLM agent within an automated, iterative simulation refinement framework like that of Behari et al. (2024). For each of K pieces of feedback, we first use the LLM agent to generate a program. Then, we apply a rapid verification procedure to the generated program based on three criteria:

(1) *Syntactic correctness*. We attempt to execute the program in a Python interpreter to ensure it represents syntactically correct Python. If not, then it cannot generate an updated simulation.
(2) *Feasibility*. We attempt to solve the new QIP for the time segment $t^{(k)}$, as well as for adjacent time segments $(t^{(k)}-1, t^{(k)}+1)$. Assuming that the feedback is internally consistent, and given the underconstrained nature of the problem, we expect that the solver should be able to quickly find (see runtime results in Section 5.2) at least one feasible solution $\hat{\mathbf{r}}^{(k)}$ for the LLM-generated QIP.
(3) *Semantic correctness*. We attempt to verify that the LLM agent's modification to the simulation actually corresponds to the feedback given, based on the solution $\hat{\mathbf{r}}^{(k)}$ to the feasibility check. To do so, we use the LLM agent to perform *self-reflection* (Shinn et al. 2023). It uses the `get_counts` tool to first retrieve relevant counts $\mathbf{Ar}^{(k-1)}$ from the previous solution, and then the same counts $\mathbf{A}\hat{\mathbf{r}}^{(k)}$ from the candidate solution. Then, the LLM compares these counts while taking into account the feedback $p^{(k)}$ to return a binary signal of whether the modification is semantically correct.

If the program fails any of these three verification criteria, we discard the program and prompt the LLM agent to generate a new one. This process repeats until the LLM agent generates a correct program $\text{QP}^{(k)}$ for feedback $p^{(k)}$. In the next iteration, we prompt the LLM agent to directly modify $\text{QP}^{(k)}$ to produce $\text{QP}^{(k+1)}$. After at least K generations, we obtain a single program $\text{QP}^{(K)}$, which we execute for all time steps t to obtain the final solution $\mathbf{r}^{(K)}$ and a corresponding simulation.

5 SIMULATION RESULTS: STRONGSVILLE, OHIO

We applied our demand modeling pipeline from Section 4 to simulate a large road network in the city of Strongsville, Ohio. The Strongsville road network experiences heavy through traffic due to its connection to two interstates, I-71 and I-80; the ramps of these interstates respectively connect to two intersecting arterials, SR 82 (Royalton Road) and US 42 (Pearl Road). The daily traffic volumes of both of these roads have exceeded their designed capacities, leading to the implementation of various countermeasures to improve throughput (Euthenics and TranSystems 2023). As part of these countermeasures, Strongsville installed an adaptive traffic signal control system on SR 82 and US 42.

This system uses three types of detectors to adjust the signaling pattern for its controllers. (1) Camera detectors are used for the main roads at each intersection (i.e., along SR 82 and US 42) and on some side roads. (2) Loop detectors are used for most side roads and some turning movements. (3) Radar detectors are used for detection upstream and downstream of intersections. As our goal is to match the traffic state at the intersections themselves, we do not consider data from Strongsville's radar detectors.

Figure 3: Screenshot of traffic simulation of Strongsville, Ohio. Counts are available for 36 intersections, either derived from vehicle tracking on raw camera detector footage (approaches in blue) or from loop/AutoScope detectors (approaches in orange).

Our simulation, as shown in Figure 3, covers the intersections along SR 82 and US 42 for which the city has installed adaptive signal control. We first converted OpenStreetMap data to a SUMO (Alvarez Lopez et al. 2018) road network. Next, on Friday, September 6, 2024, we captured 24 hours of footage from 74 out of 86 counting locations where AutoScope detectors are installed. We used counts from AutoScope and loop detectors to fill in missing counts from vehicle tracking. After we applied our pipeline, we randomly assigned vehicles to different vehicle classes (Weinblatt et al. 2013), following a survey conducted by the Ohio DOT in September 2022. Finally, we implemented the traffic signal patterns that were in use.

In the rest of this section, we evaluate the accuracy of our pipeline steps for this simulation.

5.1 Accuracy of Vehicle Counting

To evaluate the accuracy of our vehicle tracking-based counting method (Section 4.1), we manually counted traffic from camera detector footage. As doing so would be infeasible for the entire simulation, we selected footage from four different intersections that are important to stakeholders (from the south, center, east, and north of the road network), and two one-hour time segments (12 pm, an off-peak hour, and 5 pm, a peak hour) for each intersection. Figure 2 shows a screenshot from one of these pieces of footage.

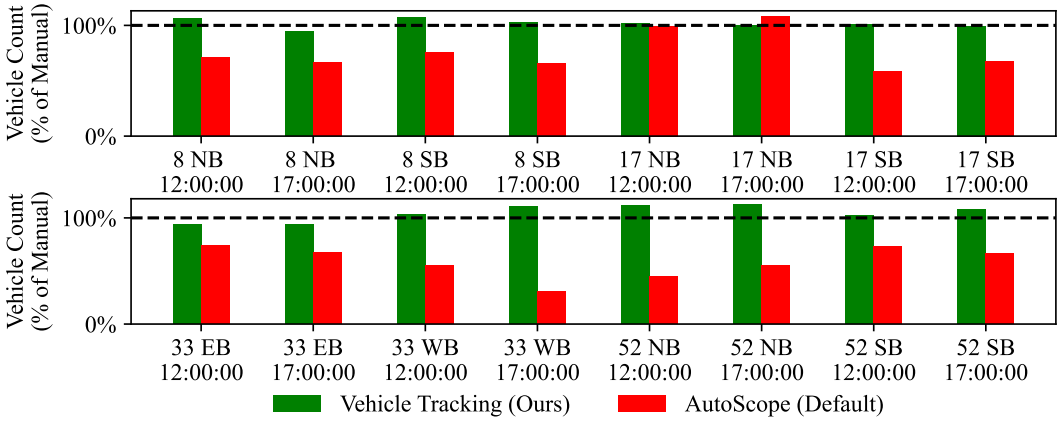

Figure 4: Plot of counts from our vehicle tracking method (green) and AutoScope (red), as ratios relative to manual counts. If a method's result perfectly matches the manual counts, it has a ratio of 100%.

In Figure 4, we compare the counts generated by our vehicle tracking method and by AutoScope to the ground truth from manual counting. Based on our evaluation, our method was able to faithfully capture the traffic state of Strongsville. For all 16 of the footage excerpts that we manually counted, our method had an error of less than 2 vehicles per minute (120 vehicles per hour). We used the same set of hyperparameters for preprocessing and detection/tracking across all counting locations; tuning these hyperparameters for individual counting locations could yield further gains.

Meanwhile, AutoScope exhibited a persistent pattern of undercounting across all of the approaches that we manually counted. In fact, it had an error of *more* than 3 vehicles per minute (180 vehicles per hour) for 14 out of 16 footage excerpts, with the exception being intersection 17's northbound approach (where we observed that lane switching resulted in duplicated actuations). The primary cause of this undercounting was the continuous actuation of detection zones by consecutive vehicles, as we discussed in Section 3. Consistent with this, AutoScope was generally more accurate under intermittent traffic during the 12 pm time segment, and its accuracy degraded under increased traffic levels during the 5 pm time segment.

We used these results to estimate lower and upper bound ratios between our vehicle tracking counts and the ground truth: $\alpha_{\text{LB}}^{CV} = 0.94, \alpha_{\text{UB}}^{CV} = 1.12$. As there is insufficient overlap between manual and loop detector counts, we extrapolated bounds for loop detector counts based on the ratio between vehicle tracking and loop detector counts. These bounds are loose due to the level of error in the loop detectors:

$$\alpha_{\text{LB}}^{LD} = \alpha_{\text{LB}}^{CV} \min_{t} \min_{j \in CV \cap LD} \frac{f_{jt}^{CV}}{f_{jt}^{LD}} = 0.02, \alpha_{\text{UB}}^{LD} = \alpha_{\text{UB}}^{CV} \max_{t} \max_{j \in CV \cap LD} \frac{f_{jt}^{CV}}{f_{jt}^{LD}} = 19.06.$$

5.2 Accuracy of Generated Simulation

Next, we solved QIP (1) to generate a set of routes consistent with these counts (Section 4.2). Owing to the underconstrained nature of the problem, obtaining a feasible solution for the 18 496-variable QIP in each time segment required less than 0.5 seconds; thus, our approach scales well to moderately-sized

road networks. However, to optimize solution quality in the final simulation, we ran the QIP for each time segment for 60 seconds. The objective function's value was mainly determined by bound violations for vehicle tracking counts $\|\mathbf{s}_t^{CV}\|_2^2$, which were two orders of magnitude larger than violations for loop detector counts $\|\mathbf{s}_t^{LD}\|_2^2$ and the fringe route penalty $\sum_{i \in \text{nonfringe}} r_{it}$. To balance the objective function, we set the hyperparameter $\lambda = 10$. We obtained a simulation with a total volume of 182 230 vehicles over 24 hours. Among these vehicles, 72.64% had routes that started and ended on the fringes of the road network.

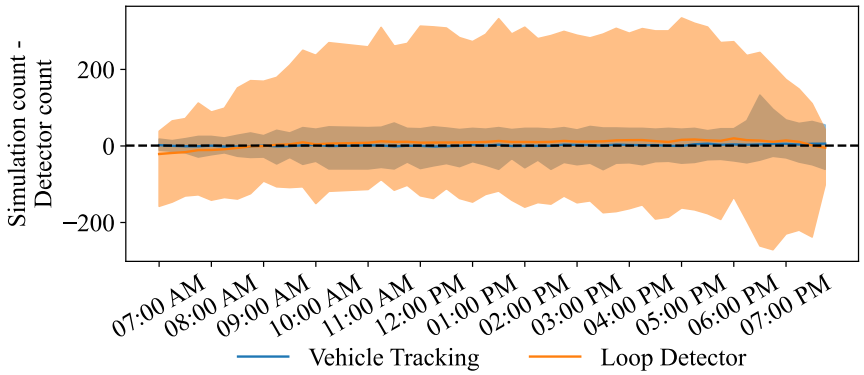

Figure 5: Plot of difference between counts in the QIP-generated simulation, counts from our vehicle tracking-based computer vision method, and counts from loop detectors. For each time segment, the solid line represents the mean across counting locations, while the shaded region represents the range.

In Figure 5, we focus on the accuracy of our simulation for the time interval between sunrise (6:59 am) and sunset (7:51 pm) on September 6, 2024. Across counting locations on average, the simulation was accurate to our computer vision counts f_j^{CV}, with no overflow or underflow. This can be attributed to the relatively narrow range of $[\alpha_{\text{LB}}^{CV}, \alpha_{\text{UB}}^{CV}]$. However, there are individual counting locations where the simulation has substantial overflow or underflow, especially so for the loop detector counts. We attribute these to counting locations with few vehicles where the detected traffic flow is inconsistent. Violating the expected bounds of these counts results in a small penalty compared to the rest of the objective function.

5.3 Accuracy of LLM Agent-Generated Constraints

As our LLM for iterative simulation refinement (Section 4.3), we used Claude 3.7 Sonnet. An earlier iteration of this model achieved state-of-the-art performance on code generation benchmarks (Zhuo et al. 2025). We performed two rounds of evaluation: one on synthetically-generated feedback (where a ground truth exists for constraint correctness), and one on real stakeholder feedback (where there is no ground truth). Our evaluation focused on the three criteria used by the LLM agent to verify generated code (Section 4.3): syntactic correctness, feasibility, and semantic correctness.

First, we randomly generated $K = 20$ pieces of structured feedback in the form of (intersection, direction, approach, increase/decrease) tuples. We used Claude to rephrase this feedback to match the style of stakeholder feedback. Here, we did not use reflection to assess semantic correctness, but instead directly verified the traffic volume in the updated simulation against the structured feedback. We generated ten programs for each piece of feedback with a temperature of 0.8.

The LLM agent always generated valid Python code, giving a syntactic correctness rate of *100%*. The feasibility rate was *87%*. We found that tool use was important to prevent hallucination of counts. Not all generated programs were feasible due to two issues in the added constraints: (1) they included a slack variable term s_t, which conflicted with the slack variable constraints from the original optimization problem, or (2) they were formulated in terms of vehicle tracking-based computer vision counts f_t^{CV}, which were not always available. Lastly, the semantic correctness rate was *87%* — whenever the generated program

was feasible, the result was also correct. This gave us confidence in continuing to use our approach, after adding the reflection procedure and modifying the prompt to prevent the two aforementioned issues.

Second, we collected $K = 20$ pieces of feedback by presenting our simulation for the 5:00 pm time segment to a stakeholder familiar with Strongsville's traffic conditions. Among the 20 pieces of feedback, 12 pointed to intersections that were true to real life (particularly those with counts based on vehicle tracking-based computer vision), while 8 pointed to intersections where the simulated traffic needed improvement. With reflection in place, we generated a single program for each piece of feedback, again with a temperature of 0.8. The LLM agent had a syntactic correctness rate of *100%*, a feasibility rate of *100%*, and a semantic correctness rate of *100%* from reflection, including for the 8 pieces of feedback that indicated changes.

Our final simulation of Strongsville was created by executing the simulation refinement procedure in sequence for all $K = 20$ pieces of feedback from the stakeholder. For each modification to the original simulation, the LLM agent's reflection procedure indicated that it accurately captured the feedback. When the stakeholder was presented with the final simulation, they concurred with the LLM agent regarding the improvements that had been made. The simulation had a total volume of 199 649 vehicles over 24 hours. This increase can be in part attributed to increased volume at intersections without vehicle tracking counts.

6 CONCLUSION

In this work, we presented an end-to-end pipeline for modeling demand in traffic simulations with three steps: computer vision-based vehicle counting, combinatorial optimization-based vehicle route generation, and LLM-based iterative simulation refinement from natural language feedback. We applied our pipeline to a high-traffic road network in Strongsville, Ohio. Based on our evaluation results, our demand modeling methodology adheres more faithfully to real-world traffic conditions than approaches used in past work, and it holds promise in generalizing to road networks from other municipalities with similarly multimodal detector data. For Strongsville, we could generate simulations quickly even when we exhaustively enumerated the route set. However, the number of routes increases exponentially with the number of intersections. To improve the scalability of our pipeline for even larger road networks, we suggest that the route set should not be sampled, but instead clustered into geographic subregions connected at boundaries by major roads.

We also envision that other sources of data could be incorporated into our demand modeling framework. Road state reports (e.g. Waze), weather data, and business information can all be indicative of factors that impact traffic. As LLMs' capabilities improve, they hold promise for integrating data from these heterogeneous sources (Chang et al. 2024). While our pipeline is offline, one line of future work is to convert it into a streaming pipeline capable of near-real-time use. Streaming capabilities would allow simulations to be updated based on live traffic. They would also enable the creation of an interactive interface where stakeholders can directly iterate on detection and optimization parameters through natural language feedback, while being able to review the results of their feedback instantaneously. This broad frontier of possibilities enabled by AI can help traffic simulations to better serve their users.

ACKNOWLEDGMENTS

We thank Scott Morse and Dave Palmer of Path Master, Kyle Love and Eric Raamot of Econolite, Sean Fitzgerel of PTV Group, Michael Schweikart of TMS Engineers, Ken Mikula and Lori Daley of the City of Strongsville, and Lisa Kay Schweyer, Stan Caldwell, and Karen Lightman of Traffic21/Safety21 — crucial stakeholders without whom this work would not have been possible. Additionally, we thank Jenny T. Liang, Kush Jain, Sean Qian, Naveen Raman, Yixuan Xu, and Jingwu Tang for their invaluable ideas and feedback on the technical portions of this work. This work was supported by a research grant from Mobility21, a US DOT National University Transportation Center, and the Tang Family Endowed Innovation Fund.

REFERENCES

Aharon, N., R. Orfaig, and B.-Z. Bobrovsky. 2022. "BoT-SORT: Robust Associations Multi-Pedestrian Tracking". *arXiv preprint* 2206.14651:1–13 https://doi.org/10.48550/arXiv.2206.14651.

Alvarez Lopez, P., M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, *et al*. 2018. "Microscopic Traffic Simulation using SUMO". In *Proceedings of the 21st International Conference on Intelligent Transportation Systems*, ITSC '18, 2575–2582. Piscataway: IEEE https://doi.org/10.1109/ITSC.2018.8569938.

Austin, J., A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, *et al*. 2021. "Fine-Tuning Language Models from Human Preferences". *arXiv preprint* 2108.07732:1–34 https://doi.org/10.48550/arXiv.1909.08593.

Barceló, J. 2010. "Models, Traffic Models, Simulation, and Traffic Simulation". In *Fundamentals of Traffic Simulation*, edited by J. Barceló, 1–62. Springer https://doi.org/10.1007/978-1-4419-6142-6_1.

Behari, N., E. Zhang, Y. Zhao, A. Taneja, D. Nagaraj, and M. Tambe. 2024. "A Decision-Language Model (DLM) for Dynamic Restless Multi-Armed Bandit Tasks in Public Health". In *Proceedings of the 38th Conference on Neural Information Processing Systems*, NeurIPS '24, 1–38. Vancouver, Canada: NeurIPS https://doi.org/10.48550/arXiv.2402.14807.

Bieker, L., D. Krajzewicz, A. Morra, C. Michelacci, and F. Cartolano. 2014. "Traffic Simulation for All: A Real World Traffic Scenario from the City of Bologna". In *Proceedings of the 2014 SUMO Conference*, SUMO '14, 47–60. Berlin, Germany: Springer https://doi.org/10.1007/978-3-319-15024-6_4.

Buades, A., B. Coll, and J.-M. Morel. 2011. "Non-Local Means Denoising". *Image Processing On Line* 1:208–212 https://doi.org/10.5201/ipol.2011.bcm_nlm.

Chang, C., S. Wang, J. Zhang, J. Ge, and L. Li. 2024. "LLMScenario: Large Language Model Driven Scenario Generation". *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54(11):6581–6594 https://doi.org/10.1109/TSMC.2024.3392930.

Chen, R., K. M. Carley, F. Fang, and N. Sadeh. 2023. "Purpose in the Machine: Do Traffic Simulators Produce Distributionally Equivalent Outcomes for Reinforcement Learning Applications?". In *Proceedings of the 2023 Winter Simulation Conference*, WSC '23, 1842–1853. San Antonio, USA: ACM https://doi.org/10.5555/3643142.3643294.

Codeca, L., R. Frank, and T. Engel. 2015. "Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of Mobility for Vehicular Networking Research". In *Proceedings of the 2015 Vehicular Networking Conference*, VNC '15, 1–8. Kyoto, Japan: IEEE https://doi.org/10.1109/VNC.2015.7385539.

Coifman, B., D. Beymer, P. McLauchlan, and J. Malik. 1998. "A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance". *Transportation Research Part C: Emerging Technologies* 6(4):271–288 https://doi.org/10.1016/S0968-090X(98)00019-9.

Diamond, S., and S. Boyd. 2016. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization". *Journal of Machine Learning Research* 17(83):1–5 https://doi.org/10.5555/2946645.3007036.

Euthenics, and TranSystems. 2023. "Preliminary Feasibility Study — Cuyahoga/Medina Traffic Study". Technical Report PID 116069, City of Strongsville.

Han, X., Q. Yang, X. Chen, Z. Cai, X. Chu, and M. Zhu. 2024. "AutoReward: Closed-Loop Reward Design with Large Language Models for Autonomous Driving". *IEEE Transactions on Intelligent Vehicles* Early Access:1–13 https://doi.org/10.1109/TIV.2024.3485964.

Jocher, G., and J. Qiu. 2024. "YOLO11". Technical report, Ultralytics. v11.0.0, https://github.com/ultralytics/ultralytics.

Kwak, M.-A., T. Arentze, E. de Romph, and S. Rasouli. 2012. "Activity-based Dynamic Traffic Modeling: Influence of Population Sampling Fraction Size on Simulation Error". In *Proceedings of the 13th International Conference on Travel Behaviour Research*, IATBR '12, 1–17. Toronto, Canada: IATBR.

Lee, H., and B. Coifman. 2012. "Identifying Chronic Splashover Errors at Freeway Loop Detectors". *Transportation Research Part C: Emerging Technologies* 24:141–156 https://doi.org/10.1016/j.trc.2012.02.005.

Leon, J. F., F. Giancola, A. Boccolucci, and M. Neroni. 2023. "A Demand Modelling Pipeline for an Agent-Based Traffic Simulation of the City of Barcelona". In *Proceedings of the 2023 Winter Simulation Conference*, WSC '23, 1777–1782. San Antonio, USA: ACM https://doi.org/10.1109/WSC60868.2023.10407551.

Li, S., T. Azfar, and R. Ke. 2024. "ChatSUMO: Large Language Model for Automating Traffic Scenario Generation in Simulation of Urban MObility". *IEEE Transactions on Intelligent Vehicles* Early Access:1–12 https://doi.org/10.1109/TIV.2024.3508471.

Lobo, S., S. Neumeier, E. M. G. Fernandez, and C. Facchi. 2020. "InTAS - The Ingolstadt Traffic Scenario for SUMO". In *Proceedings of the 2020 SUMO User Conference*, SUMO '20, 73–92. Berlin, Germany: SUMO https://doi.org/10.52825/scp.v1i.102.

Medina, J., M. Chitturi, and R. Benekohal. 2010. "Effects of Fog, Snow, and Rain on Video Detection Systems at Intersections". *Transportation Letters* 2(1):1–12 https://doi.org/10.3328/TL.2010.02.01.1-12.

Mei, H., X. Lei, L. Da, B. Shi, and H. Wei. 2024. "LibSignal: An Open Library for Traffic Signal Control". *Machine Learning* 113:5235–5271 https://doi.org/10.1007/s10994-023-06412-y.

Michalopoulos, P. G. 1991. "A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance". *IEEE Transactions on Vehicular Technology* 40(1):21–29 https://doi.org/10.1109/25.69968.

Qiu, A., P. A. Sathish, D. Wang, and H. D. Schotten. 2024. "Advanced Traffic Demand Generation in SUMO: ML-based Prediction of Flow Rate based on Real-world Measured Datasets". In *Proceedings of the 2024 IEEE 99th Vehicular Technology Conference*, VTC '24, 1–7. Singapore: IEEE https://doi.org/10.1109/VTC2024-Spring62846.2024.10683300.

Rapelli, M., C. Casetti, and G. Gagliardi. 2022. "Vehicular Traffic Simulation in the City of Turin from Raw Data". *IEEE Transactions on Mobile Computing* 21(12):4656–4666 https://doi.org/10.1109/TMC.2021.3075985.

Rhodes, A., D. M. Bullock, J. R. Sturdevant, and Z. T. Clark. 2005. "Evaluation of Stop Bar Video Detection Accuracy at Signalized Intersections". Technical Report FHWA/IN/JTRP-2005/28, Joint Transportation Research Program, Indiana Department of Transportation and Purdue University.

Shinn, N., F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. 2023. "Reflexion: Language Agents with Verbal Reinforcement Learning". In *Proceedings of the 37th Conference on Neural Information Processing Systems*, NeurIPS '23, 8634–8652. New Orleans, USA: NeurIPS https://doi.org/10.5555/3666122.3666499.

Tituana, D. E. V., S. G. Yoo, and R. O. Andrade. 2022. "Vehicle Counting using Computer Vision: A Survey". In *Proceedings of the 2022 IEEE 7th International Conference for Convergence in Technology*, I2CT '22, 1–7. Pune, India: IEEE https://doi.org/10.1109/I2CT54291.2022.9824432.

Uppoor, S., and M. Fiore. 2012. "A Large-Scale Vehicular Mobility Dataset of the Cologne Urban Area". In *Proceedings of the 14th French Conference on Algorithms and Telecommunications*, AlgoTel '12, 1–4. Hérault, France: HAL.

Wang, H., Y. Yu, Y. Cai, X. C. L. Chen, and Q. Liu. 2019. "A Comparative Study of State-of-the-Art Deep Learning Algorithms for Vehicle Detection". *IEEE Intelligent Transportation Systems Magazine* 11(2):82–95 https://doi.org/10.1109/MITS.2019.2903518.

Wei, J., X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, *et al*. 2022. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In *Proceedings of the 36th Conference on Neural Information Processing Systems*, NeurIPS '22, 24824–24837. New Orleans, USA: NeurIPS https://doi.org/10.5555/3600270.3602070.

Weinblatt, H., E. Minge, and S. Petersen. 2013. "Length-Based Vehicle Classification Schemes and Length Bin Boundaries". *Transportation Research Record* 2339(1):19–29 https://doi.org/10.3141/2339-03.

Zhang, H., S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, *et al*. 2019. "CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario". In *Proceedings of the 2019 World Wide Web Conference*, WWW '19, 3620–3624. San Francisco, USA: ACM https://doi.org/10.1145/3308558.3314139.

Zhuo, T. Y., M. C. Vu, J. Chim, H. Hu, W. Yu, R. Widyasari, *et al*. 2025. "BigCodeBench: Benchmarking Code Generation with Diverse Function Calls and Complex Instructions". In *Proceedings of the 2025 International Conference on Learning Representations*, ICLR '25, 1–55. Singapore: ICLR https://doi.org/10.48550/arXiv.2406.15877.

AUTHOR BIOGRAPHIES

REX CHEN (email rexc@cmu.edu) is a PhD graduate from the Societal Computing program of the Software and Societal Systems Department in the School of Computer Science at Carnegie Mellon University. His research focuses on applying reinforcement learning and other AI techniques to transportation and other socially impactful domains. His work aims to design AI systems capable of addressing the key deployment considerations of stakeholders in these domains.

KAREN WU (email karenw2@andrew.cmu.edu) is a second-year undergraduate student at the School of Computer Science at Carnegie Mellon University.

JOHN MCCARTNEY (email john.mccartney@pathmasterinc.com) is a 2002 University of Toledo graduate with a B.S. in Mechanical Engineering Technology. John is a Systems Support Engineer with 23 years in the transportation industry and 20 years at Path Master, which is a distributor of traffic control equipment for Ohio, Western Pennsylvania, Kentucky, and West Virginia. He is responsible for providing technical support and training for traffic controllers, detection, and management systems to industry professionals. His day-to-day activities include overseeing the implementation of Centracs ATMS and Mobility systems in the Path Master territory, troubleshooting software, electrical, and communication issues, plus guiding traffic signal projects to completion.

FEI FANG (email feifang@cmu.edu) is an Associate Professor in the Software and Societal Systems Department in the School of Computer Science at Carnegie Mellon University. Her research interests lie in the area of artificial intelligence and multi-agent systems, focusing on the integration of computational game theory and machine learning to address real-world challenges in critical domains such as security, sustainability, and mobility.

NORMAN SADEH (email sadeh@cs.cmu.edu) is a Professor in the School of Computer Science at CMU, where he is affiliated with the Software and Societal Systems Department, the Human-Computer Interaction Institute, and the CyLab Security and Privacy Institute. His research interests span mobile computing, the IoT, cybersecurity, privacy, machine learning, AI, and related public policy issues. His past work includes deployed planning and scheduling technologies for commercial systems.