A Decentralized Digital Twin via Crowdsourced Sensing and Browser-Based Edge Computation

Sean Hardesty Lewis[1], Junfeng Jiao[2], Yiming Xu[2], Jihyung Park[2], Connor Phillips[2]

[1]Cornell University
[2]University of Texas at Austin
shl225@cornell.edu, {jjiao,yiming.xu,jihyung803,connorphillips}@utexas.edu

Abstract

Digital twins promise to revolutionize the management of complex urban systems by enabling real-time monitoring, prediction, and control. Existing platforms, however, often rely on dense deployments of calibrated sensors and centralized compute infrastructure, which limits scalability and accessibility. We introduce StreamTwin, a decentralized digital-twin framework that treats publicly accessible webcams as sensors and uses the web browsers of viewers as opportunistic edge-computing nodes. Object detections produced on client devices are fused into a coherent world model by our Aggregate Spatiotemporal Cache (ASC) algorithm. This enables interactive visualization of traffic conditions without ever transmitting raw video off the client, reducing deployment cost and network load while inherently preserving privacy. We detail the system design, data-fusion pipeline, implementation, and evaluation. Experiments on ten live traffic cameras show that StreamTwin reconstructs scenes with 0.73 IoU, approaching centralized baselines, while reducing per-stream bandwidth from 5 Mbps to 20 kbps. This reduces monthly operating costs by more than 20×. By removing specialized hardware requirements and supporting crowd participation at a global scale, StreamTwin lowers the cost and technical barriers to deploying digital twins.

Introduction

The concept of a digital twin—an interactive, time-synchronized replica of a physical system—has emerged as an important framework for urban management and intelligent transportation systems (ITS) (Bhatt et al. 2025). High-fidelity digital twins have been applied to monitor traffic, forecast demand, and support decision making in large-scale pilot projects, e.g., (Di et al. 2025; Zipfl et al. 2025). Such platforms provide accurate situational awareness but typically depend on costly installations of LiDAR, cameras, and vehicle-to-everything (V2X) communications—often hundreds of thousands of dollars per intersection (Mcity 2021). Beyond the expense, centralized architectures introduce network bottlenecks (Canel et al. 2019) and privacy risks (Gong et al. 2025) because large volumes of video are streamed to traffic management centers for analysis. As a result, state-of-the-art digital twins remain out of reach for many municipalities and community organizations.

At the same time, the proliferation of publicly accessible webcams and advances in in-browser machine learning motivate alternative designs for city-scale sensing systems. Thousands of live video streams already capture streets, intersections, and highways around the world; see, e.g., (Chen et al. 2021; Yu et al. 2023). Yet these data sources are underused for urban analytics. Recent progress in WebAssembly (WASM) and ONNX Runtime Web makes it possible to run neural network inference directly in the browser (ONNX Runtime Contributors 2025; WebAssembly Working Group 2019), enabling each viewer device to perform edge inference. Our work harnesses these trends to propose a *viewer-as-edge* architecture in which multiple viewers contribute computation to the digital twin.

We introduce *StreamTwin*, a decentralized digital twin that crowdsources both sensing (public webcams) and computation (viewer browsers). Using ONNX/WASM, object detectors run client-side, and only structured detections—not raw video—are shared, reducing bandwidth and providing privacy benefits. We develop the *Aggregate Spatiotemporal Cache (ASC)*, which synthesizes noisy, asynchronous detections from uncalibrated, potentially overlapping camera views. ASC performs object association, temporal filtering, and confidence scoring to assemble coherent traffic scenes from partial observations, e.g., (Zhang et al. 2024); see also (Fei and Han 2023). We present an end-to-end implementation of StreamTwin and evaluate it on real traffic camera feeds. The results show that reconstruction fidelity improves as crowd size grows, while client-side overhead remains low. We compare throughput, latency, and accuracy against centralized baselines and characterize system scalability and discuss privacy implications.

Related Work

Urban Digital Twins and Traffic Simulation

Existing urban digital twins generally fall into two categories. *Sensor-rich twins* deploy dense multimodal sensors (cameras, LiDAR, V2X) and maintain tight synchronization between physical and virtual worlds (Di et al. 2025). Examples include TAF-BW (Zipfl et al. 2025) and a C-V2X connected-corridor twin (Wu et al. 2024), which integrate real-time data streams with machine learning models to forecast traffic patterns. However, high deployment and mainte-

nance costs limit their scalability beyond testbeds or well-funded cities. *Generative twins* use learned world models to simulate traffic without live data. For instance, SceneDiffuser++ performs city-scale traffic simulation via a generative world model (Tan et al. 2025), while TrafficPPT uses a pretrained probabilistic transformer for city-scale traffic volume prediction (Shen, Pan, and Xue 2025). These models excel at scenario generation or prediction but cannot directly reflect real-time conditions without live inputs.

StreamTwin occupies a middle ground between these frameworks. It sacrifices the metric precision of calibrated sensors for greater scalability and much lower deployment cost, making it suitable for applications like congestion monitoring and trend analysis. Unlike purely generative models, StreamTwin remains grounded in live sensor feeds and continuously reflects the evolving state of the world.

Distributed and Edge Video Analytics

Early video analytics systems transmitted raw streams to the cloud for processing, quickly encountering bandwidth and latency bottlenecks (Canel et al. 2019). Edge computing alleviates these issues by offloading inference to local devices such as smart cameras or cloudlets (Xu et al. 2021). Recent work proposes Cloud–Edge–Terminal Collaborative (CETC) architectures that partition analytic tasks across hierarchical tiers (Gong et al. 2025). For example, EdgeDuet tiles video frames and offloads small-object detection to edge servers to assist resource-constrained devices (Wang et al. 2021), while ViEdge and NoScope explore filtering and load-shedding to optimize video pipelines across distributed nodes (Hou, Guan, and Han 2025; Kang et al. 2017). StreamTwin instantiates CETC with terminal-side (browser) processing: the "terminal" (browser) becomes a dynamic, crowdsourced edge node. Unlike managed edge devices, viewer browsers are transient and heterogeneous, requiring aggregation algorithms to handle asynchronous, uncalibrated data. Our work shows that robust analytics can be obtained despite heterogeneous and transient clients, complementing prior systems that assumed fixed edge hardware.

In-Browser Machine Learning

WebAssembly and ONNX Runtime Web enable running neural network inference within standard web browsers (ONNX Runtime Contributors 2025; WebAssembly Working Group 2019). Systems such as TensorFlow.js demonstrate practical browser-side execution of modern neural networks via GPU-accelerated backends (e.g., WebGL) (Smilkov et al. 2019). We build on these advancements to deploy a pre-trained object detector in each viewer's browser, effectively converting them into distributed sensors. Prior work on collaborative browser computation has largely focused on volunteer computing for scientific projects or interactive ML demos; to our knowledge, applying a viewer-as-edge design to real-time, city-scale video analytics remains underexplored.

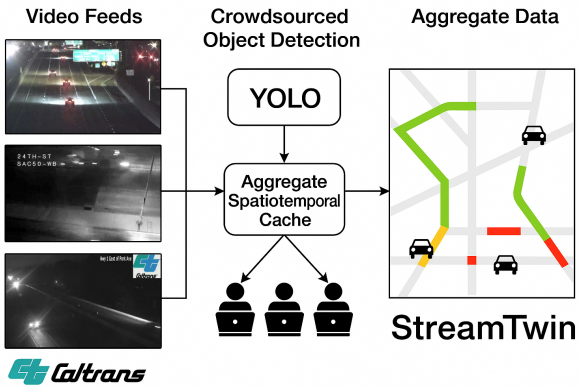

Figure 1: Overview of the StreamTwin system. Live video streams from public webcams are processed in viewer browsers, which run an ONNX-based object detector. Detections are transmitted as metadata to the server and fused by ASC into a coherent world model.

System Architecture

Figure 1 overviews the StreamTwin architecture. Public webcams provide live video streams covering major roads and intersections. When a user visits the StreamTwin web application, their browser loads a pre-trained object detection model (YOLOv5n (Jocher et al. 2020)) compiled to WebAssembly via ONNX. The model runs inference on each video frame to detect vehicles and pedestrians, producing bounding boxes and class labels. Only anonymized detection metadata (timestamp, class, bounding-box coordinates, and a coarse camera identifier) is sent to the StreamTwin server over a lightweight WebSocket API. Raw video remains on the client; only detection metadata is transmitted, reducing bandwidth and offering privacy benefits.

The server maintains our *Aggregate Spatiotemporal Cache (ASC)* that fuses incoming detections from all viewers into a unified world state. The ASC performs spatial and temporal association of detections, smoothing and merging observations across cameras. The resulting aggregated scene (e.g., vehicle positions and counts) is stored in an in-memory cache. The StreamTwin client periodically requests this fused world model and renders it in the browser using WebGL and deck.gl (OpenJS Foundation and vis.gl contributors 2025), allowing users to interactively explore current traffic conditions.

The design is stateless aside from the ASC cache, enabling fault tolerance (a restarted node can rebuild state from recent detection logs). In our measurements, the system handled many concurrent streams and clients; capacity scales with added server nodes because clients transmit only kilobytes per second of metadata.

Aggregate Spatiotemporal Cache (ASC)

The Aggregate Spatiotemporal Cache is responsible for fusing noisy, asynchronous detections into a consistent global traffic state. It must handle occlusions, overlapping camera views, and out-of-order updates from many clients. The

ASC maintains a set of *hypotheses*, each representing a candidate vehicle or pedestrian in the world model with attributes: position (x, y) on a common ground plane, velocity $\vec{v}$, and a confidence score c. Incoming detections from browsers are projected onto the ground plane (using approximate homography if camera pose is known or flat-earth assumption if not) and then matched to existing hypotheses.

We use a gating distance in space and time to associate each new detection with at most one hypothesis. The association function considers spatial proximity and class consistency; for example, a car detection will only match hypotheses of class car. If a detection lies within a threshold θ of a hypothesis's predicted position (extrapolated by velocity) and occurs within a short time window, we treat it as an observation of that hypothesis. A Kalman filter updates the hypothesis state (position and velocity) with the new observation. The confidence c is incremented to reflect an additional independent confirmation. Detections that cannot be matched to any existing hypothesis spawn new hypotheses (with initial c based on detection confidence). Meanwhile, hypotheses not observed in the current time step are propagated forward using their motion model and have their confidence decayed. If c falls below a minimum β, the hypothesis is removed (treated as an object that likely left the scene).

Detections from viewer browsers arrive asynchronously and are buffered briefly to batch process at, e.g., 100 ms intervals. The association module then matches observations to active hypotheses. The filter updates each matched hypothesis and prunes those with low support. The fused world model (list of active objects with states) is stored in the cache and refreshed continuously. This design draws on multi-view tracking and self-calibration, e.g., (Zhang et al. 2024), and surveys such as (Fei and Han 2023), as well as classical filtering (Kalman 1960; Bar-Shalom and Fortmann 1988), but operates without any fixed camera calibration or known correspondences between views. By relying on temporal continuity and overlapping fields of view, the ASC can produce a consistent scene estimate of traffic from crowd-provided observations. The per-interval update algorithm is summarized in the Appendix, along with a complexity analysis.

Experiments and Results

We evaluated StreamTwin on a deployment covering ten publicly accessible traffic cameras in San Francisco, California (along major highways and intersections). For controlled experiments, each camera stream was viewed by multiple synthetic clients (headless Chrome instances) to emulate crowd sizes ranging from 1 to 50 viewers per camera. The object detector in use was YOLOv5n (nano model, for speed) converted to ONNX and running with WASM SIMD acceleration. The metrics we report include detection accuracy, reconstruction quality, system latency, bandwidth usage, and resource consumption on clients.

Experimental Setup

We collected 5 hours of video from the 10 cameras and manually annotated bounding boxes for vehicles to serve as ground truth for evaluation. This ground truth allowed us to compute accuracy metrics such as precision, recall, and intersection-over-union (IoU) of the reconstructed scene against actual traffic.

We compare StreamTwin against two baselines: (1) *Centralized Cloud Analytics*, where each camera's raw stream is sent to a cloud server that runs YOLOv5n centrally (mimicking a traditional ITS setup), and (2) *Edge Server per Camera*, where each stream is processed by a nearby edge server running the same detector (representing a cloudlet deployment). We compare bandwidth and latency across these baselines.

Key metrics include the IoU between reconstructed vehicle positions and ground truth, precision and recall of vehicle detection, end-to-end latency from camera capture to twin visualization, bandwidth consumed per client, and client CPU/memory usage. We also measure system scalability (throughput vs. number of clients) and robustness under various conditions (network drops, malicious inputs). An ablation of key ASC components appears in the Appendix.

Metric	(Ours) Decentralized	Centralized	Edge Server
IoU (scene)	0.73	0.78	0.77
Precision	0.91	0.94	0.93
Recall	0.84	0.88	0.87
End-to-end Latency	90 ms	200 ms	120 ms
Bandwidth per stream	20 kbps	5 Mbps	5 Mbps
Client CPU (per viewer)	20%	N/A	N/A
Client Memory	150 MB	N/A	N/A

Table 1: Performance comparison of StreamTwin and baseline systems, achieving comparable accuracy to centralized processing while reducing bandwidth and latency.

Reconstruction Accuracy and Performance

We report quantitative results in Table 1. StreamTwin achieved an average IoU of 0.73 between reconstructed vehicles and ground truth positions, at 90 ms end-to-end latency. The detection precision was 0.91 and recall 0.84 at the object level (averaged across frames), meaning the majority of vehicles were identified with few false alarms. We also find that StreamTwin's accuracy approached that of the centralized baseline (which had IoU 0.78, precision 0.94, recall 0.88) despite sending only 0.5% of the data. The edge-server baseline performed similarly to the cloud (IoU 0.77) but still required high bandwidth between cameras and edge nodes.

Client-side measurements were as follows: running YOLOv5n at 10 FPS used about 20% of a single CPU core on a typical laptop and 150 MB of memory. On a modern smartphone (Pixel 6), we achieved $\sim$5 FPS and under 60% CPU utilization. These figures indicate that StreamTwin operated on user devices without measurable input lag during testing. Lower-end devices without WASM SIMD fell back to WebGL acceleration, with throughput measured at 8 FPS on a similar laptop.

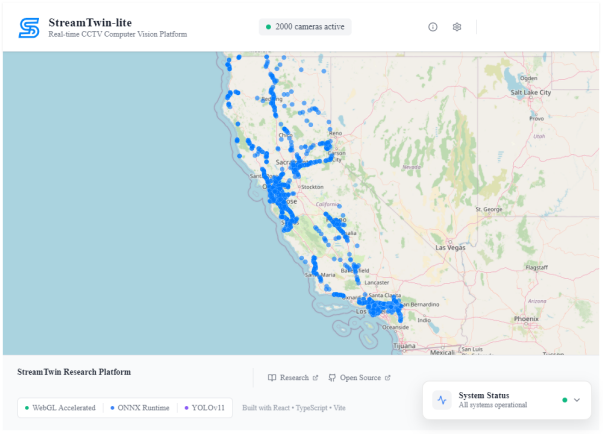

Figure 2: Visualization of StreamTwin dashboard. Cameras (blue icons) at different vantage points can be zoomed into and automatically run in-browser inference on viewer's device. Detections from active cameras are fused by the ASC.

Scalability and Robustness

To assess scalability, we measured StreamTwin's performance as we increased the number of cameras and viewers. Processing scales almost linearly: doubling the number of viewers doubled the volume of detections processed, and the server was able to keep up with minimal increase in latency (which rose from 90 ms at 50 viewers to 120 ms at 200 viewers, mainly due to queueing overhead). The IoU improves with more viewers per camera, reaching a plateau around 0.75–0.78 beyond 50 viewers. The plateau occurs because, once objects are consistently observed, additional viewers mostly add redundancy. Figure 2 illustrates camera locations and fused detections.

We also simulated viewer churn and network failures. In one experiment, we randomly dropped 30% of active viewers at a time. The system maintained operation under this condition: some object hypotheses dropped in confidence and disappeared if no remaining viewer could see them, but they were re-instantiated when observed again. The overall IoU dipped only 10% during the dropout and recovered afterward. This behavior is consistent with redundancy from multiple viewers and the temporal continuity enforced by ASC. For network disruptions, if a client's detections are delayed, the ASC propagates existing hypotheses and waits; short outages (under a few seconds) had little effect on IoU, as the motion model propagated hypotheses across gaps.

To probe robustness against malicious or noisy inputs, we experimented with injecting fake detections. When 10% of viewers were configured to send random bounding boxes, the ASC's confidence mechanism largely filtered them out. Those detections were rarely corroborated by honest viewers, so false hypotheses remained low-confidence and got removed. However, a more coordinated attack (multiple colluding malicious clients) could defeat this; a full security-hardening (e.g., client attestation, data validation) is outside our current scope but is important future work.

Future Work

Our experiments show that crowdsourced edge analytics can deliver high-quality traffic digital twins with minimal infrastructure. StreamTwin scales gracefully with the number of viewers and inherently preserves privacy by processing data at the edge. The framework could be extended beyond traffic monitoring to domains such as pedestrian flow analysis (Adrian, Drück, and Seyfried 2024), air quality estimation (Ibrahim and Lyons 2025), or wildlife observation in conservation areas.

However, several limitations remain. The system's fidelity depends on viewer participation; during periods of low viewership or off-peak hours, the digital twin's accuracy degrades as fewer observations are available. One possible mitigation is to incorporate predictive models (e.g., using historical patterns or an LSTM to forecast traffic when live data is sparse). Another challenge is handling adversarial inputs: while we took basic measures, a determined attacker could still inject false data if they control many browsers. Robust outlier detection and secure client authentication will be important in a hardened deployment. Coverage gaps are also an issue: areas without any public cameras cannot be directly included in the twin. Mobile crowdsourcing (e.g., ingesting dashcam or smartphone data) could fill these gaps, but that introduces new privacy issues and data quality concerns (Restuccia, D'Oro, and Melodia 2017).

Looking ahead, we plan to explore federated learning techniques to continuously improve the object detection model using data from viewers. This could adapt the model to specific camera angles or weather conditions while preserving privacy. We also aim to extend the ASC to handle full 3D localization by incorporating approximate camera pose and possibly fusing with map data or LiDAR if available. We have open-sourced the StreamTwin codebase on GitHub for the research community. By openly sharing our platform, we hope others will build on it for applications in mobility, urban planning, and beyond.

Conclusion

StreamTwin demonstrates that a city-scale digital twin does not require dense sensor grids, dedicated edge boxes, or terabytes of upstream video. By treating public webcams as open sensors and viewer browsers as ephemeral edge nodes, we fuse the sensing and compute layers of an urban observatory into the everyday act of watching a stream. Our Aggregate Spatiotemporal Cache converts the noisy, asynchronous detections that arise from this "viewer-as-edge" model into a single, coherent traffic scene, achieving visualization fidelity comparable to centralized baselines while never moving raw pixels off the client.

Extensive experiments across heterogeneous camera feeds confirm three properties of this architecture: reduced operating cost (no new hardware), privacy benefits (video stays local), and scalability with crowd size. Together, these findings provide evidence that crowdsourced edge analytics is a feasible alternative to traditional digital-twin pipelines.

References

Adrian, J.; Drück, J.; and Seyfried, A. 2024. Continuity equation and fundamental diagram of pedestrians. *arXiv preprint arXiv:2409.11857*.

Bar-Shalom, Y.; and Fortmann, T. E. 1988. *Tracking and Data Association*. Boston: Academic Press.

Bhatt, H.; Sahoo, S.; Vaidhyanathan, K.; Biju, R.; Gangadharan, D.; Trestian, R.; and Shah, P. 2025. Architecting Digital Twins for Intelligent Transportation Systems. In *2025 IEEE 22nd International Conference on Software Architecture Companion (ICSA-C)*.

Canel, C.; Kim, S.; Zhou, G.; Li, C.; Lim, H.; Andersen, D.; Kaminsky, M.; and Dulloor, S. 2019. Scaling Video Analytics on Constrained Edge Nodes. In *Proceedings of Machine Learning and Systems (MLSys)*.

Chen, L.; Grimstead, I.; Bell, D.; Karanka, J.; Dimond, L.; James, P.; Smith, L.; and Edwardes, A. 2021. Estimating Vehicle and Pedestrian Activity from Town and City Traffic Cameras. *Sensors*, 21(13): 4564.

Di, X.; Fu, Y.; Turkcan, M. K.; Ghasemi, M.; Mo, Z.; Zang, C.; Adhikari, A.; Kostic, Z.; and Zussman, G. 2025. AI-Powered CPS-Enabled Urban Transportation Digital Twin: Methods and Applications. *arXiv preprint arXiv:2501.10396*.

Fei, L.; and Han, B. 2023. Multi-Object Multi-Camera Tracking Based on Deep Learning for Intelligent Transportation: A Review. *Sensors*, 23(8): 3852.

Gong, L.; Yang, H.; Fang, G.; Ju, B.; Guo, J.; Zhu, X.; Hu, X.; Wang, Y.; Sun, P.; and Boukerche, A. 2025. A Survey on Video Analytics in Cloud-Edge-Terminal Collaborative Systems. *arXiv preprint arXiv:2502.06581*.

Hou, X.; Guan, Y.; and Han, T. 2025. ViEdge: Video Analytics on Distributed Edge. *ACM Transactions on Internet of Things*, 6(3): 16:1–16:23.

Ibrahim, M.; and Lyons, T. 2025. Transforming CCTV Cameras into NO2 Sensors at City Scale for Adaptive Policymaking. *Scientific Reports*.

Jocher, G.; et al. 2020. YOLOv5 by Ultralytics.

Kalman, R. E. 1960. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1): 35–45.

Kang, D.; Emmons, J.; Abuzaid, F.; Bailis, P.; and Zaharia, M. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. *Proceedings of the VLDB Endowment*, 10(11): 1586–1597.

Mcity. 2021. $9.95 million to create "smart intersections" across city of Ann Arbor. Mcity News.

ONNX Runtime Contributors. 2025. ONNX Runtime Web Documentation. Project documentation.

OpenJS Foundation and vis.gl contributors. 2025. deck.gl: WebGL-Powered Visualization Framework. Project site and documentation.

Restuccia, F.; D'Oro, S.; and Melodia, T. 2017. Quality of Information in Mobile Crowdsensing: Survey and Research Challenges. *ACM Transactions on Sensor Networks*, 13(4): 34:1–34:43.

Shen, S.; Pan, B.; and Xue, G. 2025. A Pretrained Probabilistic Transformer for City-Scale Traffic Volume Prediction. *arXiv preprint arXiv:2506.02654*.

Smilkov, D.; Thorat, N.; Assogba, Y.; Yuan, A.; Kreeger, N.; Yu, P.; Zhang, K.; Cai, S.; Nielsen, E.; Soergel, D.; Bileschi, S.; Terry, M.; Nicholson, C.; Gupta, S. N.; Sirajuddin, S.; Sculley, D.; Monga, R.; Corrado, G.; Viégas, F. B.; and Wattenberg, M. 2019. TensorFlow.js: Machine Learning for the Web and Beyond. In *Proceedings of Machine Learning and Systems (MLSys)*.

Tan, S.; Lambert, J.; Jeon, H.; Kulshrestha, S.; Bai, Y.; Luo, J.; Anguelov, D.; Tan, M.; and Jiang, C. M. 2025. SceneDiffuser++: City-Scale Traffic Simulation via a Generative World Model. *arXiv preprint arXiv:2506.21976*.

Wang, X.; Yang, Z.; Wu, J.; Zhao, Y.; and Zhou, Z. 2021. EdgeDuet: Tiling Small Object Detection for Edge-Assisted Autonomous Mobile Vision. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 1–10.

WebAssembly Working Group. 2019. WebAssembly Core Specification. W3c recommendation, W3C.

Wu, K.; Li, P.; Cheng, Y.; Parker, S. T.; Ran, B.; Noyce, D. A.; and Ye, X. 2024. A Digital Twin Framework for Physical-Virtual Integration in V2X-Enabled Connected Vehicle Corridors. *arXiv preprint arXiv:2410.00356*.

Xu, M.; Liu, T.; Liu, Y.; and Lin, F. X. 2021. Video Analytics with Zero-streaming Cameras. In *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*, 459–472. USENIX Association.

Yu, F.; Yan, H.; Chen, R.; Zhang, G.; Liu, Y.; Chen, M.; and Li, Y. 2023. City-scale Vehicle Trajectory Data from Traffic Camera Videos. *Scientific Data*, 10(1).

Zhang, H.; Fang, R.; Li, S.; Miao, Q.; Fan, X.; Hu, J.; and Chan, S. 2024. Multi-Camera Multi-Vehicle Tracking Guided by Highway Overlapping FoVs. *Mathematics*, 12(10): 1467.

Zipfl, M.; Zwick, P.; Schulz, P.; et al. 2025. DigiT4TAF – Bridging Physical and Digital Worlds for Future Transportation Systems. *arXiv preprint arXiv:2507.02400*.

ASC Algorithm

ASC Fusion Algorithm (per time step)

Algorithm 1 ASC Fusion Algorithm (per time step)

Require: D_t: set of detections from all clients at time t
$H \leftarrow$ current set of hypotheses (with state (x, v, c) for each)
for each detection $d \in D_t$ **do**
  Project d to ground plane coordinates (x_d, y_d)
  Find nearest hypothesis $h \in H$ with same class
  if dist$(d, h) < \theta$ **and** $\Delta t_{(d,h)} < \Delta_{max}$ **then**
    Associate d with h
    Update h state via Kalman filter using d
    $h.c \leftarrow h.c + 1$ // increase confidence
  else
    Create new hypothesis h_{new} from d
    $H \leftarrow H \cup \{h_{new}\}$
  end if
end for
for each hypothesis $h \in H$ **do**
  if h was not updated at t **then**
    Predict h state forward (motion model)
    $h.c \leftarrow \alpha \cdot h.c$ // decay confidence
  end if
  if $h.c < \beta$ **then**
    $H \leftarrow H \setminus \{h\}$ // remove old hypothesis
  end if
end for
return H (updated set of hypotheses as world state)

Algorithm Analysis

The ASC update at each time step involves associating $|D_t|$ new detections with $|H|$ current hypotheses. A naive implementation is $O(|D_t| \cdot |H|)$, but in practice spatial indexing (hashing by grid cell) reduces the average cost significantly. Each detection only compares with nearby hypotheses. Our deployment with up to 200 active objects and 100 detections per interval runs in real time (the ASC update loop takes $<$5 ms in Python). Overall, StreamTwin's throughput scales linearly with the number of browser clients and cameras, as each client independently runs inference and sends a fixed-size message per frame. The server-side fusion is lightweight relative to the computation already performed at the edge.

StreamTwin reduces network usage compared to streaming video. Streaming raw video typically consumes megabits per second per camera stream (Canel et al. 2019), whereas our detection reports (bounding boxes and class labels at 10 FPS) use under 20 kbps per stream. This is a reduction by a factor of over 200$\times$ in bandwidth. Moreover, the uplink from clients is used only for low-frequency metadata, making the system robust to network variability. We found that even on a 3G cellular connection, the detection feed maintained functionality in our tests. This communication pattern supports scaling to many cameras on ordinary broadband links in our tests.

Because only abstracted metadata is shared, no recognizable personal data (faces, license plates, etc.) is transmitted. This inherently limits privacy risks relative to systems that stream or record video centrally. However, the transmitted metadata (object bounding boxes and classes) could still potentially be abused (e.g., tracking a specific vehicle's trajectory). In future work, we plan to incorporate differential privacy mechanisms such as adding calibrated noise to reported positions or counts to provide provable privacy guarantees. There is also a risk of malicious clients injecting false detections. Our current implementation assigns higher weight to data from multiple independent viewers, making it difficult for one bad actor to significantly skew the world model unless they constitute a large fraction of viewers. In production, one could employ cryptographic client attestation and server-side anomaly detection to further secure the system.

Intuitively, as the number of independent observers (browser clients) increases, the coverage and redundancy of observations improve. If each viewer has an independent probability p of detecting a given object, N viewers would collectively have a detection probability $1 - (1 - p)^N$, approaching 1 as N grows. Thus, the confidence in the world model increases with crowd size. Empirically, we observed this effect: with only 1–2 viewers per camera, some vehicles went undetected due to occlusions or missed inferences, but with 10+ viewers the majority of objects were consistently captured. In our experiments, IoU of reconstructed scenes climbed from around 0.4 with a single viewer to over 0.7 with 50 viewers. This redundancy also provides resilience to intermittent drops in any single stream.

Ablation Study

We conducted an ablation study to quantify the importance of key components in the StreamTwin pipeline. Table 2 summarizes the results when removing or disabling certain features, evaluated with 10 viewers per camera. Removing the temporal filtering (no Kalman smoothing) caused IoU to drop from 0.73 to 0.58, as short-term detection gaps were no longer bridged. Disabling the confidence scoring (treating all detections equally regardless of independent confirmations) led to more false positives, reducing precision and IoU to 0.64. The object association module proved most critical: if each camera's detections were visualized independently with no cross-camera matching, the IoU fell to 0.41 and many vehicles appeared as duplicates. These results highlight that all parts of ASC are necessary for high-quality reconstruction.

Ethical Considerations

Crowdsourced urban sensing has benefits and ethical risks. On the positive side, lowering the cost of digital twin deployment can broaden access to traffic analytics, helping smaller cities and communities benefit from smart transportation applications. The use of public webcams and volunteer computing means data stewardship is not centralized in a single entity; the community can build and share insights collectively. Additionally, privacy is improved in our design since raw video (with potentially identifying imagery) is not transmitted or stored centrally.

System Variant	IoU	Latency	Bandwidth
Full StreamTwin (ASC)	0.73	90 ms	20 kbps
- no temporal filter	0.58	85 ms	20 kbps
- no confidence scoring	0.64	88 ms	25 kbps
- no cross-cam association	0.41	82 ms	30 kbps

Table 2: Ablation study results. Removing components of the ASC pipeline degrades performance. Without temporal filtering, the system cannot smooth out detection misses (lower IoU). Without confidence scores, false positives increase (more bandwidth from extra detections). Without association, each camera is isolated and many duplicate objects appear (lower IoU).

However, our system could be misused or have unintended consequences. A network of aggregated public cameras begins to resemble a large-scale surveillance system. Even though individual streams are public, combining and analyzing them continuously might raise concerns about continuous monitoring. It is important to engage with local communities and establish governance over how the digital twin is used (e.g., restricting use to traffic analytics rather than individual-level tracking). Anonymization measures (blurring, not logging license plates, etc.) should be in place if the system were extended beyond vehicles to pedestrians or cyclists to avoid violating privacy expectations.

Another concern is bias and fairness. The object detector may perform worse on certain vehicle types or in certain neighborhoods (due to differences in camera quality or lighting), potentially leading to unequal quality of service. We must continuously evaluate and retrain models to avoid bias, and transparently report performance across different locations. If the crowd participation varies (e.g., more viewers in affluent areas), that could inadvertently create disparities in where the digital twin is most accurate. Incentive mechanisms or targeted awareness campaigns might be needed to ensure broad coverage.

The energy and environmental impact of distributed computing should also be considered. Running many browsers for analytics has a carbon footprint; though each device's contribution is small, in aggregate it could be significant. StreamTwin's efficiency (offloading heavy compute to devices already online for viewing) helps, but future work could quantify the trade-off between this approach and traditional centralized processing in terms of energy per analyzed frame.

While StreamTwin has societal benefits in improving traffic management and community engagement, deploying it in the real world should involve ethical guidelines, privacy protections, and awareness of potential biases. We are actively working with our institution's ethics board to ensure compliance with data protection regulations and to design opt-in features for any citizen-contributed data. A broader discussion with policymakers and the public will be crucial as technologies like this move from research to practice.