Title

Complex Event Detection in Dynamic Distributed Sensing Systems

ISBN

9798190919967

Author

Wang, Brian

Publication Date

2026-09-08

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Complex Event Detection in Dynamic Distributed Sensing Systems

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Brian Wang

2026

ABSTRACT OF THE DISSERTATION

Complex Event Detection in Dynamic Distributed Sensing Systems

by

Brian Wang
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2026
Professor Mani B. Srivastava, Chair

Actions and decisions in physical environments often depend on detecting patterns of interest, known as complex events. In distributed sensing systems, complex event detection may require combining unstructured sensor data with structured measurements and inferences produced by sensing and recognition models. Complex events may unfold from seconds to weeks and across spatial scales ranging from a single room to an entire city.

Existing systems commonly assume that relevant operations, output representations, and execution plans have been selected in advance. However, observations may support competing explanations, sensor streams may expose more information than an application requires, and needed evidence and computation may change as an event progresses. This dissertation argues that distributed sensing systems should use the event and its application context to revise which events remain plausible, determine what information may be disclosed, and adapt sensing and computation as the event unfolds.

First, we address the challenge of **discovering and localizing complex events when distributed observations support competing event explanations**. Our system, *IncidentLens*, performs city-scale emergency monitoring by maintaining hypotheses about an incident and relationships among its observed effects. Evaluation on urban and synthetic data shows that incident priors and hypothesis management support discovery from sparse and partial observations.

Second, we address the challenge of **balancing utility with privacy and data-use constraints for complex event applications** in smart environments. Our system, *privMediator*, composes sensor-processing operations, verifies that outputs support the application task and applicable information flow rules, and selects a feasible representation that avoids unnecessary disclosure. Across smart space tasks, its outputs avoid configured policy violations and are judged more appropriate than raw media and developer-authored alternatives.

Third, we investigate **how complex event sensing and computation should adapt as events progress and system conditions change** in tactical environments. Through *FABLE*,a Python research prototype and controlled MQTT/replay testbed, we explore whether progress through a developer-authored complex-event workflow can determine which predefined sensing and inference work is useful next. Experiments on 89 recorded traces spanning nine separately authored workflows examine the recognition–computation tradeoff of progress-driven activation within these curated configurations, including a controlled network-disconnection case.

Together, these contributions examine complementary decisions in complex event detection for dynamic distributed environments, such as smart cities: distinguishing among competing event explanations, disclosing information appropriate to the application context, and using event progress to guide sensing and computation.

The dissertation of Brian Wang is approved.

Yuan Tian

Nader Sehatbakhsh

Omid Abari

Mani B. Srivastava, Committee Chair

University of California, Los Angeles

2026

The strength that carried me through was borrowed from the kindness of others.

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

I would like to thank my PhD advisor, Prof. Mani B. Srivastava, whose insight and guidance have given me the skills and confidence to pursue challenging, meaningful research. This has been a long journey, and his support throughout it has played an essential role in shaping me into the researcher I am today. I want to thank my doctoral committee members, Prof. Omid Abari, Prof. Yuan Tian, and Nader Sehatbakhsh for providing their valuable and insightful feedback that helped me shape my research directions. I also thank Prof. Songwu Lu for his valuable feedback and support during my doctoral research. I also wish to thank my undergraduate research advisor Prof. Mi Zhang (Ohio State University) who provided support early in my PhD journey.

I'm grateful to my collaborators and co-authors: Luis Antonio Garcia (UofU), Ziqi Wang (Qualcomm), Akash Deep Singh (Ownwell), Pengrui Quan (Samsung), Swapnil Sayan Saha (STMicroelectronics), Julian de Gortari Briseno (UCLA), Liying Han (UCLA), Pragya Sharma (UCLA), Gaofeng Dong (UCLA), Oliver Wang (UCLA), Lance Kaplan (ARL), Jeff Twigg (ARL), Maggie B. Wigness (ARL), Cale Crowder (Sandia National Labs), Henry Phillips (SoarTech), Ben Purman (SoarTech), and Jeff Craighead (SoarTech). I'm truly thankful of the time and effort they have invested into help with research and review. I am also thankful for the many engaging research discussions and supportive interactions I have shared with Jason Wu (UCLA), Xiaomin Ouyang (HKUST), Kang Yang (UCLA), Jeya Vikranth Jeyakumar (NVIDIA), Sandeep Singh Sandha (Abacus.AI), Renju Liu, and many others.

Portions of this dissertation are based on collaborative and co-authored research.

The *SIGMUS* material presented in Chapter 2 is based on Brian Wang and Mani Srivastava, "SIGMUS: Semantic Integration for Knowledge Graphs in Multimodal Urban Spaces," presented at the 14th International Workshop on Urban Computing (UrbComp 2025), KDD 2025, Toronto, August 3, 2025, arXiv:2509.00287. I led the research, including system design, implementation, evaluation, data analysis, and manuscript preparation.

Mani Srivastava provided research feedback and support.

The *PrivacyOracle* material presented in Chapter 2 is based on Brian Wang, Luis Antonio Garcia, and Mani Srivastava, "PrivacyOracle: Configuring Sensor Privacy Firewalls with Large Language Models in Smart Built Environments," in the 2024 IEEE Security and Privacy Workshops (SPW), pp. 239–245, IEEE, 2024. I led the research, including system design, implementation, evaluation, data analysis, and manuscript preparation. Luis Antonio Garcia and Mani Srivastava provided research feedback and support.

The *DANCER* material presented in Chapter 2 is based on Brian Wang, Julian de Gortari Briseno, Liying Han, Henry Phillips, Jeffrey Craighead, Ben Purman, Lance Kaplan, and Mani Srivastava, "Adapting Complex Event Detection to Perceptual Domain Shifts," in MILCOM 2024 – 2024 IEEE Military Communications Conference, pp. 1–6, IEEE, 2024. I led the research, including system design, integration, evaluation, data analysis, and manuscript preparation. Julian de Gortari Briseno and Liying Han contributed code and developed simulation tools. Jeffrey Craighead and Ben Purman contributed simulation tools and data. Henry Phillips, Lance Kaplan, and Mani Srivastava provided research feedback and support.

Chapter 3 is based in part on Brian Wang, Oliver Wang, Gaofeng Dong, and Mani Srivastava, "IncidentLens: Inferring City-Scale Incidents from Heterogeneous Urban Observations," in preparation for submission to the ACM Conference on Embedded Networked Sensor Systems (SenSys 2027). I led the research, including system design, implementation, evaluation, data analysis, and manuscript preparation. Oliver Wang and Gaofeng Dong provided labeling support and contributed to data analysis. Mani Srivastava provided research feedback and support.

Chapter 4 is based in part on Brian Wang and Mani Srivastava, "Contextual Privacy Pipelines for Frictional Shared Smart Environments," submitted to Proceedings on Privacy Enhancing Technologies (PoPETs 2027), Privacy Enhancing Technologies Symposium (PETS 2027). I led the research, including system design, implementation, evaluation, data analysis, and manuscript preparation. Mani Srivastava provided research

feedback and support.

Chapter 5 is based in part on Brian Wang, Mani Srivastava, Jeffrey Twigg, and Maggie B. Wigness, "FABLE: Frontier-Driven Orchestration for Distributed Complex Event Recognition in IoT Systems," in preparation for submission to the IEEE Internet of Things Journal. I led the research, including system design, implementation, evaluation, data analysis, and manuscript preparation. Jeffrey Twigg and Maggie B. Wigness provided experimental support, facilitated access to the experimental space, and helped construct the dataset used in the research. Mani Srivastava provided research feedback and support.

To my loving parents, Shige Wang and Chunzhi Bi, who have supported and encouraged me throughout the journey, and have given me the strength to do my best. I'd also like to thank my friends who have provided the necessary joy and personal growth that inspires both research and life. Ordered by last name, I'm thankful to Diederik Beckers, Nissryne Dib, Alessandro Dellarovere, Alex Kerns, Akshay Bharadwaj Krishna, Shruti Sharan, Amrutha Srinivasan, and many other friends I have made along the journey.

The research reported in this dissertation was sponsored in part by the National Science Foundation under Awards OAC-1640813, OAC-1636916, 1705135, and 2124252; the National Institutes of Health Center of Excellence for Mobile Sensor Data-to-Knowledge under Award 1-U54EB020404-01; the NIH mDOT Center under Award 1P41EB028242; the DEVCOM Army Research Laboratory under Cooperative Agreements W911NF-17-2-0196 and W911NF-22-2-0243 and STTR Contract W911NF-22-P-0064; the Air Force Office of Scientific Research under Award FA9550-22-1-0193; the DARPA ANSR Program under Contract FA8750-23-C-0519; and Sandia National Laboratories under Award 2169310. Any opinions, findings, conclusions, or recommendations expressed in this dissertation are those of the author and do not necessarily reflect the views of the sponsoring agencies.

VITA

2014–2018 B.S. in Computer Science, Michigan State University

PUBLICATIONS

Wang, Brian, et al. "FABLE: Frontier-Driven Orchestration for Distributed Complex Event Recognition in IoT Systems." **(In preparation)** To be submitted to IEEE Internet of Things Journal.

Wang, Brian, et al. "Contextual Privacy Pipelines for Frictional Shared Smart Environments." **(Under review)** Submitted to Proceedings on Privacy Enhancing Technologies (PoPETs 2027), Privacy Enhancing Technologies Symposium (PETS 2027).

Wang, Brian, Oliver Wang, Gaofeng Dong, and Mani Srivastava. "IncidentLens: Inferring City-Scale Incidents from Heterogeneous Urban Observations." **(In Preparation)** To be submitted to the ACM Conference on Embedded Networked Sensor Systems (SenSys 2027).

Sharma, Pragya, Brian Wang, and Mani Srivastava. "CADET: A Modular Platform for Evaluating Distributed Cooperative Autonomy in Connected Autonomous Vehicles." In 2026 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2026. arXiv:2606.04072. https://doi.org/10.48550/arXiv.2606.04072

Quan, Pengrui, Brian Wang, Kang Yang, Liying Han, and Mani Srivastava. "Benchmarking spatiotemporal reasoning in llms and reasoning models: Capabilities and challenges." Advances in Neural Information Processing Systems 38 (2026)

Wang, Brian, and Mani Srivastava. "SIGMUS: Semantic Integration for Knowledge Graphs in Multimodal Urban Spaces." **(Best paper award)** In the 14th International Workshop on Urban Computing (UrbComp 2025), KDD 2025, Toronto, August 3, 2025. arXiv:2509.00287.

Wang, Brian, Julian de Gortari Briseno, Liying Han, Henry Phillips, Jeffrey Craighead, Ben Purman, Lance Kaplan, and Mani Srivastava. "Adapting Complex Event Detection

to Perceptual Domain Shifts." In MILCOM 2024-2024 IEEE Military Communications Conference (MILCOM), pp. 1-6. IEEE, 2024.

Wang, Brian, Luis Antonio Garcia, and Mani Srivastava. "PrivacyOracle: Configuring sensor privacy firewalls with large language models in smart built environments." In 2024 IEEE Security and Privacy Workshops (SPW), pp. 239-245. IEEE, 2024.

Wang, Ziqi, Brian Wang, and Mani Srivastava. "Protecting User Data Privacy with Adversarial Perturbations." In Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021), pp. 386-387. 2021.

Antoniadou, Eleni, David Belo, Krittika D'Silva, Brian Wang, Brian Russell, Frank Soboczenski, Annie Martin, Graham Mackintosh, and Tianna Shaw. "A generative machine learning framework for synthesizing sympotomatic ecg astronaut health data." In NASA HRP Investigator's Workshop, Galveston, TX, USA. 2020.

Wang, Brian, and Mani B. Srivastava. "Enabling Privacy Policies for mHealth Studies." In 2019 IEEE International Conference on Big Data (Big Data), pp. 4045-4054. IEEE, 2019.

0.1 Introduction

The principal chapters of this dissertation address three problems in complex-event construction: discovering and localizing complex events from distributed urban effects, constructing privacy-aware outputs for complex-event applications, and adapting distributed sensing and computation as complex events unfold. These pillars emerged from three earlier investigations of event-centric sensing. The precursor systems addressed different applications and were not components of one architecture, but each exposed a decision that fixed sensor-processing pipelines leave implicit.

SIGMUS established an early foundation for the interpretation pillar by showing that incidents can organize heterogeneous urban evidence that would otherwise remain fragmented. Its dependence on incidents introduced through textual or authoritative reports exposed a harder problem: constructing a latent incident interpretation directly from distributed effects. That limitation motivates *IncidentLens*.

PrivacyOracle established an early foundation for the representation pillar by showing that high-level privacy context can govern whether sensor information is released, filtered, or transformed. Its reliance on unstructured language-model judgments and informal tool descriptions exposed the need for typed operators, explicit utility contracts, residual-flow construction, and deterministic policy evaluation. That limitation motivates *privMediator*.

DANCER established an early foundation for the execution pillar by showing that explicit complex-event dependencies can guide adaptation of lower-level perception. Its runtime nevertheless retained a mostly predetermined sensing pipeline. This exposed the need to use the changing progress of each partial event occurrence as a continuous signal for deciding which computations should run, where they should execute, and what state they require. That limitation motivates *FABLE*.

This chapter condenses the original publications to the abstractions, architectures, representative findings, and limitations needed to explain how those three problem formulations emerged. Publication-specific background, extensive related work, and secondary

Pillar	**Precursor**	**Challenge exposed**	**Principal system**
Interpretation	*SIGMUS*	Discover and localize latent incidents when incident identity, origin, extent, and observation membership are not known in advance.	*IncidentLens*
Representation	*PrivacyOracle*	Construct auditable, utility-sufficient outputs under contextual information-flow constraints.	*privMediator*
Execution	*DANCER*	Adapt distributed sensing and computation according to event progress and deployment conditions.	*FABLE*

Table 0.1: Research progression from precursor systems to the three construction pillars developed in the principal chapters.

experiments are omitted because the later chapters develop the mature interpretation, representation, and execution formulations.

CHAPTER 1

Foundations of Complex-Event Construction in Distributed Sensing

1.1 Introduction

The principal chapters of this dissertation address three problems in complex-event construction: discovering and localizing complex events from distributed urban effects, constructing privacy-aware outputs for complex-event applications, and adapting distributed sensing and computation as complex events unfold. These pillars emerged from three earlier investigations of event-centric sensing. The precursor systems addressed different applications and were not components of one architecture, but each exposed a decision that fixed sensor-processing pipelines leave implicit.

SIGMUS established an early foundation for the interpretation pillar by showing that incidents can organize heterogeneous urban evidence that would otherwise remain fragmented. Its dependence on incidents introduced through textual or authoritative reports exposed a harder problem: constructing a latent incident interpretation directly from distributed effects. That limitation motivates *IncidentLens*.

Pillar	Precursor	Challenge exposed	Principal system
Interpretation	*SIGMUS*	Discover and localize latent incidents when incident identity, origin, extent, and observation membership are not known in advance.	*IncidentLens*
Representation	*PrivacyOracle*	Construct auditable, utility-sufficient outputs under contextual information-flow constraints.	*privMediator*
Execution	*DANCER*	Adapt distributed sensing and computation according to event progress and deployment conditions.	*FABLE*

Table 1.1: Research progression from precursor systems to the three construction pillars developed in the principal chapters.

PrivacyOracle established an early foundation for the representation pillar by showing that high-level privacy context can govern whether sensor information is released, filtered, or transformed. Its reliance on unstructured language-model judgments and informal tool descriptions exposed the need for typed operators, explicit utility contracts, residual-flow construction, and deterministic policy evaluation. That limitation motivates *privMediator*.

DANCER established an early foundation for the execution pillar by showing that explicit complex-event dependencies can guide adaptation of lower-level perception. Its runtime nevertheless retained a mostly predetermined sensing pipeline. This exposed the need to use the changing progress of each partial event occurrence as a continuous signal for deciding which computations should run, where they should execute, and what state they require. That limitation motivates *FABLE*.

This chapter condenses the original publications to the abstractions, architectures, representative findings, and limitations needed to explain how those three problem formulations emerged. Publication-specific background, extensive related work, and secondary experiments are omitted because the later chapters develop the mature interpretation, representation, and execution formulations.

1.2 *SIGMUS*: Semantic Integration of Urban Evidence

Cities expose many public and semi-public data sources, including traffic cameras, environmental monitors, roadway measurements, weather services, news feeds, and public alerts. A major incident can affect several of these sources while remaining only partially visible in each. A wildfire may be named in a news report, appear as smoke in a camera image, increase particulate matter, disrupt traffic, and propagate according to current weather. Relating these observations requires a representation that separates the occurrence from the evidence used to describe it.

SIGMUS introduced a multimodal knowledge graph in which an *incident* acts as a semantic anchor. A *report* represents an observable unit such as an article, image, or time-series sample and records its source, time, location, modality, and generated annotations. Reports are produced by observers and collected through aggregators; text reports may additionally identify actors and lower-level events using a controlled vocabulary (noa, e). The ontology reuses temporal concepts from OWL-Time (Hobbs and Pan, 2004) and sensor concepts from the Semantic Sensor Network ontology (noa, h), while adding explicit records for model-generated captions, extracted entities, anomaly summaries, and cross-modal links.

The prototype ingested ten sources spanning text, images, and tabular measurements. Modality-specific components normalized each source, stored the underlying media or measurements, and wrote references and annotations into the graph. Language models extracted actors and event descriptions from text; vision-language models converted images into human-readable observations; and time-series analysis summarized anomalies and historical deviations. Candidate cross-modal links were generated using spatial and temporal proximity, then evaluated with contextual information and model world knowledge. The resulting graph provided a live memory in which users could inspect an incident together with the reports and inferred relationships that supported it.

The system demonstrated plausible semantic connections across five representative

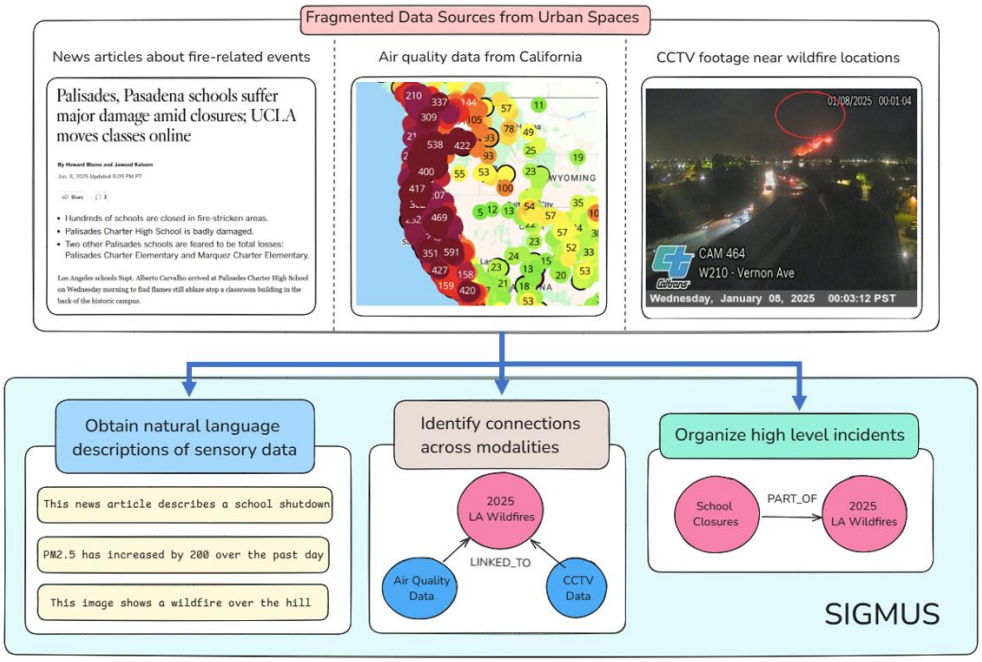


Figure 1.1: *SIGMUS* converts heterogeneous urban observations into reports and links them to incidents in a shared knowledge graph.

source types—news, CCTV imagery, air quality, weather, and traffic—and illustrated the approach using major Los Angeles incidents. It also exposed important limitations. Link quality depended on model judgment and incomplete source metadata, and the prototype did not provide a comprehensive event-detection benchmark. More importantly for this dissertation, the incident node was often introduced by a named news event. The system answered, "Which observations are related to this known incident?" but not the more difficult question, "Which latent incident best explains observations before an authoritative report names it?"

Chapter 2 addresses that gap. *IncidentLens* retains the separation between incident and evidence, together with time, space, source, and provenance, but replaces incident-centered linking with competing source hypotheses, explicit propagation templates, online evidence association, composition, and stability-based discovery.

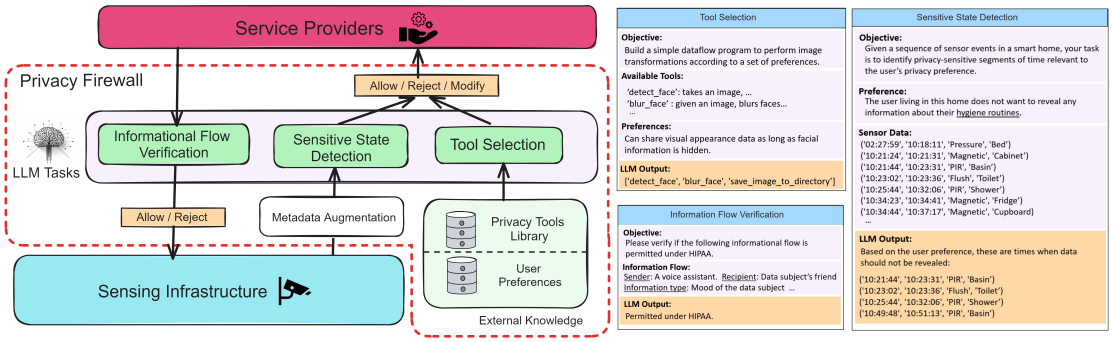


Figure 1.2: *PrivacyOracle* uses LLM reasoning to configure information-flow verification, sensitive-state filtering, and privacy-tool selection between sensing infrastructure and service providers.

1.3 *PrivacyOracle*: Automated Sensor Privacy Firewalls

Smart spaces can contain many sensors and services while occupants have limited awareness of the resulting information flows. Expecting each person to evaluate every sensor, recipient, purpose, inference, and privacy tool does not scale. *PrivacyOracle* explored whether large language models could supply the qualitative reasoning needed by an automated *privacy firewall*: a trusted layer between sensing infrastructure and service providers that can allow, reject, or modify a proposed flow.

The design used contextual integrity to represent an information flow through the subject, sender, recipient, information type, and transmission principle (Nissenbaum, 2020; Apthorpe et al., 2018). It studied three services. *Information-flow verification* asked whether a proposed flow was consistent with a legal or social context. *Sensitive-state detection* interpreted low-level, metadata-augmented sensor sequences to locate activities covered by a natural-language privacy preference. *Tool selection* chose or composed transformations capable of reducing disclosure while retaining an application's intended function.

The prototype provided early evidence that general-purpose models could assist with each task. It achieved low error on a small HIPAA information-flow study, approximately 75% binary agreement with prior user judgments of social acceptability, and up to 98%

accuracy when identifying specified privacy-sensitive activities in an instrumented-home dataset. It also generated plausible transformation pipelines using tools such as face redaction and image filtering. These results supported two ideas carried forward in the dissertation: privacy control should operate at the sensing-to-application boundary, and transformation into a narrower representation is often preferable to raw release or complete denial.

The experiments also revealed why an LLM should not directly act as the privacy decision maker. Legal and normative judgments were prompt-sensitive and incompletely calibrated; natural-language descriptions of tools did not precisely specify what information remained; and the system lacked an explicit contract connecting application utility to accepted output representations. A model could recommend a tool without establishing type compatibility, downstream utility, residual disclosure, or compliance with a maintained rule library.

Chapter 3 replaces this recommendation-centric design with structured mediation. *privMediator* uses LLMs only in bounded roles, while typed operator effects, application contracts, residual information flows, layered contextual-integrity rules, probe evidence, and least-revealing selection determine the final release. The research question therefore shifts from whether a model can make a plausible privacy recommendation to whether a mediator can construct and audit a utility-compatible representation whose disclosure is permitted in the current context.

1.4 *DANCER*: Adaptive Neurosymbolic Complex-Event Detection

Complex-event recognition over video requires two capabilities that are difficult to obtain from one model. Neural perception handles high-dimensional imagery but is vulnerable to changes in weather, lighting, viewpoint, and object appearance. Symbolic reasoning can express long temporal and spatial relationships but assumes reliable primitive events.

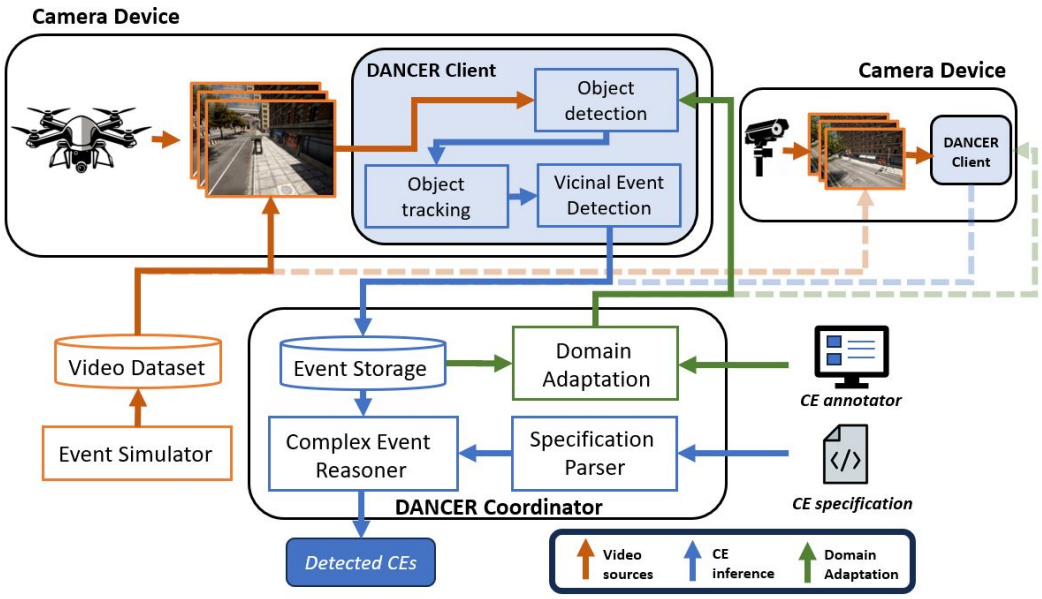


Figure 1.3: *DANCER* combines edge perception with centralized symbolic CE reasoning and event-guided domain adaptation.

DANCER combined these capabilities and asked how high-level event structure could reduce the human effort required to adapt perception after deployment.

The system represented an event hierarchy from object detections to tracked *vicinal events*, watchbox states, atomic events, and complex events. A Python-based specification language expressed logical, temporal, and sequencing relationships across regions and cameras. GPU-enabled edge clients performed detection and tracking using YOLOv5 and ByteTrack (Jocher, 2020; Zhang et al., 2022b), while a coordinator maintained watchbox state and evaluated complex-event programs. Simulation tools based on CARLA and Scenic supported rare urban and tactical scenarios under controlled appearance and environmental shifts (Team; Fremont et al., 2019).

The central adaptation procedure traced a failed or uncertain complex-event result backward through its atomic and vicinal dependencies. Users first verified compact event-level statements. Only when those checks failed did the interface request video review or image-level bounding-box annotation. Confirmed samples were then used to fine-tune the detector. This cascade used the CE program to focus attention on observations most

likely to explain a recognition failure instead of uniformly labeling frames.

Across the evaluated perceptual shifts, domain adaptation increased average CE accuracy from 50.17% to 74.3%, a 48% relative improvement. The event-guided interface reduced annotation time by up to 2.7× and reduced the number of manually annotated images. Beyond these results, *DANCER* established several concepts that recur in *FABLE*: explicit event programs, regions, tracked entity bindings, compact event messages, edge-local perception, and a coordinator that reasons across devices.

The remaining limitation was architectural. Event semantics guided an exceptional adaptation procedure around a mostly fixed pipeline; they did not continuously determine which models should run, where they should execute, or which state should move among nodes. Chapter 4 generalizes event-guided adaptation into runtime orchestration. Rather than asking which examples should be corrected after a pipeline fails, *FABLE* asks what distributed evidence computation should exist next for every live, partially matched event instance.

1.5 Lessons Learned

The precursor systems assign semantics a role earlier than the final application output. In *SIGMUS*, an incident organizes incomplete reports. In *PrivacyOracle*, contextual meaning determines whether and how sensor information should flow. In *DANCER*, event dependencies identify which perception outputs deserve additional work. Four lessons carry into the principal systems.

Semantic objects should remain distinct from their evidence and realization. An incident is not identical to any one report; a privacy goal is not identical to one transformation; and a complex-event predicate is not identical to one detector. Separating these objects enables the later systems to maintain alternative hypotheses, synthesize alternative pipelines, and select among alternative providers.

Intermediate state must be explicit. No single observation is sufficient in these settings. The later systems therefore expose state that a fixed pipeline would hide: source hypotheses and incident memory in *IncidentLens*, residual disclosure and candidate flows in *privMediator*, and partial bound instances with active frontiers in *FABLE*.

Learned world knowledge requires structured guardrails. Language and vision-language models were useful for extracting entities, explaining cross-modal relationships, identifying activities, and proposing tools. Their outputs were also variable and difficult to audit. The principal systems therefore treat learned outputs as candidate evidence or bounded assistance while relying on explicit schemas, contracts, provenance, deterministic checks, and rejection for final system decisions.

Event state can control lower-level work. *DANCER* provides the clearest precursor: dependencies in a CE program determine which frames are reviewed and adapted. *FABLE* elevates this idea into a general runtime abstraction. Likewise, *IncidentLens* uses the state of competing explanations to determine which observations remain relevant, and *privMediator* uses a candidate's accumulated effects to determine whether further transformations or rejection are necessary.

The relationship among the systems is evolutionary rather than a claim that the later projects are simple revisions. *IncidentLens* addresses uncertainty about *which event exists*; *privMediator* addresses constraints on *which representation may be exposed*; and *FABLE* addresses adaptation of *which computation should produce the next evidence*. These are distinct systems problems, but all benefit from making event semantics and their alternative realizations explicit.

1.6 Conclusion

The precursor systems presented in this chapter established the three research lineages that motivate the dissertation's construction pillars. *SIGMUS* showed how a known inci-

dent can organize heterogeneous urban observations, but retained the assumption that the incident is introduced through an external report or other anchor. *PrivacyOracle* showed how application context can govern sensor information flows, but relied on unstructured model judgments rather than typed representations, explicit utility requirements, and deterministic policy evaluation. *DANCER* showed how complex-event dependencies can guide adaptation of lower-level perception, but retained a largely predetermined sensing and execution pipeline.

Together, the systems exposed a common limitation. Important decisions about event interpretation, application-visible representation, and distributed execution were still assumed or selected before recognition. The principal chapters address those decisions directly by discovering and localizing incidents from distributed effects, constructing privacy-aware outputs under utility and information-flow constraints, and adapting sensing and computation as complex events unfold.

The next chapter begins with the interpretation problem. It removes the assumption that an incident has already been named and asks how a system can discover and localize a coherent city-scale event from heterogeneous and indirect observations.

CHAPTER 2

Discovering and Localizing Complex Events from Distributed Urban Effects

Chapter Overview

This chapter develops the first principal dimension of event construction: transforming distributed evidence into an event explanation. Chapter 1 showed how *SIGMUS* used a known incident as a semantic anchor for organizing multimodal urban reports. That architecture was useful for integration and retrospective analysis, but it depended heavily on textual or authoritative sources to introduce the incident itself. The harder setting considered here begins before such an anchor exists. The system observes effects—for example smoke, traffic disruption, air-quality changes, and public reports—and must determine whether they are manifestations of one latent incident, several incidents, or unrelated noise.

IncidentLens addresses this gap by maintaining explicit source hypotheses and evaluating how well they explain observations under incident-specific spatial, temporal, propagation, and composition models. In the dissertation's broader framing, the chapter asks *what happened?*: how can a sensing system construct a coherent event abstraction when the event is weakly specified and only indirectly observable? The original paper struc-

ture is retained where it supports that question, while publication-specific supplementary material has been removed from the main dissertation and preserved separately with the project artifacts.

2.1 Introduction

This chapter studies how complex events can be discovered and localized from distributed urban effects when the incident itself is not directly observed. Urban incidents provide a demanding instance of this problem. Emergencies and disasters can unfold over extended periods, affect broad geographic regions, and generate heterogeneous effects whose relationship to one another is not immediately known. The system must determine not only whether local anomalies occurred, but which observations belong together, what incident could have produced them, where it originated, and how it is evolving.

Figure 2.1 illustrates the diversity of incident behavior. The 2025 Palisades Fire began in the Santa Monica Mountains near Pacific Palisades and evolved over days into a large wildland–urban interface fire shaped by wind, terrain, fuel, and suppression. Its observable footprint extended beyond visible flames to smoke, air-quality changes, evacuations, road congestion, structure damage, and public reports across a broader region. In contrast, the 2008 Mumbai attacks unfolded through coordinated human actions at multiple urban sites. Rather than spreading continuously through physical fuel and wind, the attacks evolved through discrete actions, mobility, and emergency response. Both are *incidents*: emergencies or disasters whose effects extend across space and time. They also illustrate compositional structure, where local fires, attacks, closures, evacuations, or response activities must be interpreted as related parts of a larger event.

Timely incident interpretation can support warning, response and resource allocation, evacuation, situational awareness, and multi-agency coordination. Government agencies, companies, and citizens already provide heterogeneous sensing streams—traffic measurements, air-quality monitors, public cameras, emergency reports, social media, and news—

that may reveal an incident earlier or more completely than any single modality. Yet these streams rarely observe the incident directly. A camera may see haze rather than a fire front, a traffic sensor may see congestion rather than an evacuation, an air-quality sensor may see particulate matter rather than an ignition, and public reports may be delayed or spatially coarse.

Existing urban sensing methods typically solve narrower problems. Anomaly detectors identify deviations at sensors, road segments, regions, or social clusters, but often do not determine why observations distributed across larger regions and longer periods should be interpreted as one incident. Event classifiers recognize specific local phenomena such as crashes, smoke, floods, or crowds, but are generally specialized to a modality or event type. Source-localization and inverse-modeling methods reason backward from effects to causes, but often assume that the incident type, propagation model, and relevant observations are known in advance. Wide-area incident interpretation violates these assumptions: incident types are diverse, severe incidents are rare, labels are weak or delayed, sensors are unevenly deployed, and relevant observations may appear far from the source that caused them.

The resulting systems problem is to discover and localize a latent complex event by constructing a structured interpretation from physical, mobility, environmental, visual, and public-report observations. The incident's type, origin, spatial extent, temporal evolution, composition, and observation membership are treated as latent properties that must be inferred from indirect effects. Three challenges follow. First, observations are *indirect*: sensors often capture consequences rather than the incident source. Second, incidents follow *different dynamics*: a wildfire evolves through wind, fuel, terrain, and suppression, while a coordinated public-safety incident evolves through human decisions, mobility, and response. Third, incidents may be *compositional*: multiple local activities may belong to one larger event, while unrelated anomalies must not be incorrectly merged.

We address this interpretation problem with *IncidentLens*,a streaming sensing system that converts local observations into structured incident hypotheses. Each hypothesis contains a possible incident type, origin region, propagation behavior, and set of supporting

or contradicting observations. Given heterogeneous observations, *IncidentLens* generates and maintains competing hypotheses, scores their agreement with incoming observations, and updates, stabilizes, or decays them as new observations arrive. The system therefore interprets distributed effects as the latent structure of a complex incident rather than treating each local anomaly as an independent event.

This chapter makes the following contributions:

- We formulate wide-area incident discovery and localization as latent source-hypothesis maintenance, where distributed observations are treated as indirect effects of hidden incident sources rather than as independent local anomalies.
- We design *IncidentLens*, a streaming system that maintains competing incident hypotheses over space and time using incident-specific source priors, propagation models, and clustering criteria within a shared online inference engine.
- We evaluate *IncidentLens* on real Los Angeles data and controlled synthetic incident scenarios, comparing against direct observation, space-time clustering, text-only, late-fusion, and hotspot-scan baselines.
- We plan to release the implementation, evaluation code, synthetic generation pipeline, and shareable real-data labels and metadata to support reproducibility and future research in wide-area incident interpretation.

2.2 Problem Setting

2.2.1 Incident Definitions

Incidents are defined under the U.S. National Incident Management System (NIMS) as "an occurrence or event, natural or human-made, that necessitates a response to protect life, property or the environment" (Federal Emergency Management Agency, 2017a). In this work, we focus on unplanned incidents, such as emergencies and disasters, rather than

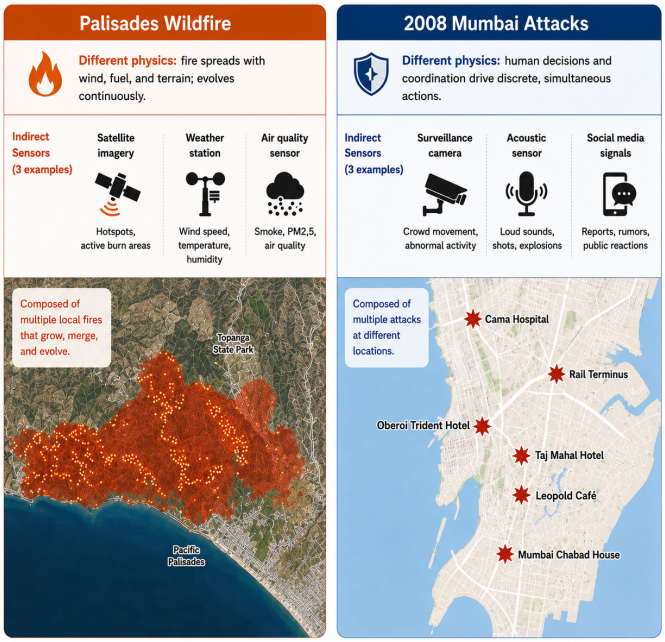


Figure 2.1: Two incident classes illustrating different propagation, composition, and observation characteristics.

planned events (such as concerts or parades). NIMS/ICS further characterizes incidents using a five-level complexity scale, ranging from Type 5 incidents, which are the least complex, to Type 1 incidents, which are the most complex (Federal Emergency Management Agency, 2021, 2018). Although these types are not defined by fixed spatial or temporal thresholds, higher-complexity Type 1–2 incidents typically extend across multiple operational periods, require substantial external resources, and may involve multijurisdictional or multidisciplinary response. In contrast, Type 3–5 incidents are generally more localized, shorter in duration, and require fewer response resources.

From the perspective of our system, we distinguish between three incident scales, summarized in Table 2.1. *Small-scale incidents* refer to lower-complexity incidents, roughly corresponding to NIMS/ICS Type 3–5 incidents, whose effects are expected to have a limited spatial and temporal footprint. Such incidents are therefore likely to be observable by only a few sensory vantage points. *Large-scale incidents* refer to higher-complexity Type 1–2 incidents, whose broader spatial and temporal footprints are more likely to generate distributed observations across many sensors, regions, and modalities. Finally, *incident complexes* refer to the NIMS-defined grouping of two or more individual incidents in the

same general area managed by a single Incident Commander or Unified Command (Federal Emergency Management Agency, 2017b).

These categories describe sensing footprint and organizational treatment rather than mutually exclusive causal classes. A small-scale incident may be the origin or early manifestation of a large-scale incident, such as a small ignition that develops into a wildfire or a localized road closure that later becomes part of a broader evacuation response. Conversely, an incident complex may compose multiple individual incidents, including small-scale incidents, large-scale incidents, or a mixture of both, when they occur in the same general area and are managed under a shared command structure. Thus, our terminology captures the difference between isolated small-scale incidents, large-scale incidents with broad sensing footprints, and composite incidents whose evidence arises from multiple related sub-incidents. In this work, we focus on detecting unplanned emergencies and disasters whose sensing footprints correspond to large-scale incidents, incident complexes, or large-scale incidents that emerge from initially small-scale incidents.

2.2.2 Problem Formulation

A city is instrumented with data sources $\mathcal{D} = d_1, \ldots, d_M$, including traffic sensors, air-quality monitors, cameras, operational text reports, social media posts, and official alerts. Each raw input record is

$$r_i = (d_i, x_i, t_i, y_i),$$

where d_i is the data source, x_i is its location or spatial footprint, t_i is its timestamp, and y_i is the raw payload. The city is represented by grid cells $\mathcal{G} = g_1, \ldots, g_N$ with static and historical context $\mathcal{X}$, such as roads, land cover, terrain, sensor coverage, and historical baselines.

The goal is to infer incident records from streams of partial observations. We represent

an incident as a latent spatiotemporal event

$$E_k = (c_k, S_k, A_k(t), I_k, \phi_k),$$

where c_k is the incident type, S_k is the source region, $A_k(t)$ is the affected region over time, $I_k = [t_k^s, t_k^e]$ is the active interval, and ϕ_k is its internal state. Incidents are latent because the source event is often not directly observed. Instead, the system observes effects that may be displaced in space or time, such as smoke downwind of a fire, congestion downstream of a crash, water near a flood, or public reports after an emergency response has begun.

This creates four spatial quantities that may not coincide: the *reported location* named by an external source, the *observed-effect location* where evidence is sensed, the *inferred source region* estimated by the system, and the *affected region* where effects are observed or predicted. For example, a wildfire may originate in a hillside cell while smoke, degraded air quality, evacuation reports, and congestion are observed kilometers away. This distinction is central to our problem setting: incident detection requires explaining distributed effects while estimating both the latent source and the region affected over time.

Given raw inputs up to time T, denoted $\mathcal{R}_{[0,T]}$, the system outputs a ranked set of incident predictions

$$\widehat{\mathcal{E}}_T = (\hat{c}_k, \hat{S}_k, \hat{A}_k(t), \hat{I}_k, \hat{\phi}_k, \hat{p} * k) * k = 1^K.$$

Here $\hat{p}_k$ is a relative support score rather than a calibrated posterior probability. The number of active incidents K is unknown in advance: new incidents may appear, multiple observations may refer to the same incident, and some observations may not correspond to any incident in the target scope. The task is therefore not only to classify individual observations, but to decide how many incident records should be created, where their latent sources may be, and which observed effects they explain.

Composition is treated as a secondary capability. Some records may later be linked

as part of a broader incident, such as multiple fires associated with a regional wildfire or several local public-safety records associated with a larger event. The primary problem in this chapter is to infer event-level incident records whose evidence can be explained by observable effects in the available data streams. *IncidentLens* is not intended to replace specialized wildfire simulators, traffic crash classifiers, gunshot localization systems, or other incident-specific detectors; it addresses the systems problem of maintaining incident-level predictions when evidence is sparse, indirect, multimodal, and weakly labeled.

2.2.3 Noisy Labels and Event-Level Evaluation

Real incident labels are delayed, incomplete, and biased toward reported events. A label may contain only a category, coarse location, approximate time interval, and provenance. We represent such an annotation as

$$\tilde{E}_k = (\tilde{c}_k, \tilde{A}_k, \tilde{I}_k, \tilde{m}_k),$$

where $\tilde{c}_k$ is the reported incident category, $\tilde{A}_k$ is the reported or affected area, $\tilde{I}_k$ is the approximate time interval, and $\tilde{m}_k$ stores metadata such as publication time and label provenance.

Because exact latent state recovery is usually unidentifiable in real data, we evaluate compatibility with available labels rather than closed-world correctness. A strict event-level match requires semantic, spatial, and temporal compatibility:

$$\text{match}_{\text{strict}}(\hat{E}, \tilde{E}) = \mathbf{1}[\text{sim}_c(\hat{E}, \tilde{E}) \geq \delta_c \wedge \text{sim}_x(\hat{E}, \tilde{E}) \geq \delta_x \wedge \text{sim}_t(\hat{E}, \tilde{E}) \geq \delta_t].$$

The similarity functions are instantiated according to label granularity: type hierarchy agreement or canonical label equality for sim_c, distance or spatial overlap for sim_x, and interval overlap for sim_t.

Synthetic labels provide source and affected-region ground truth, so synthetic evalu-

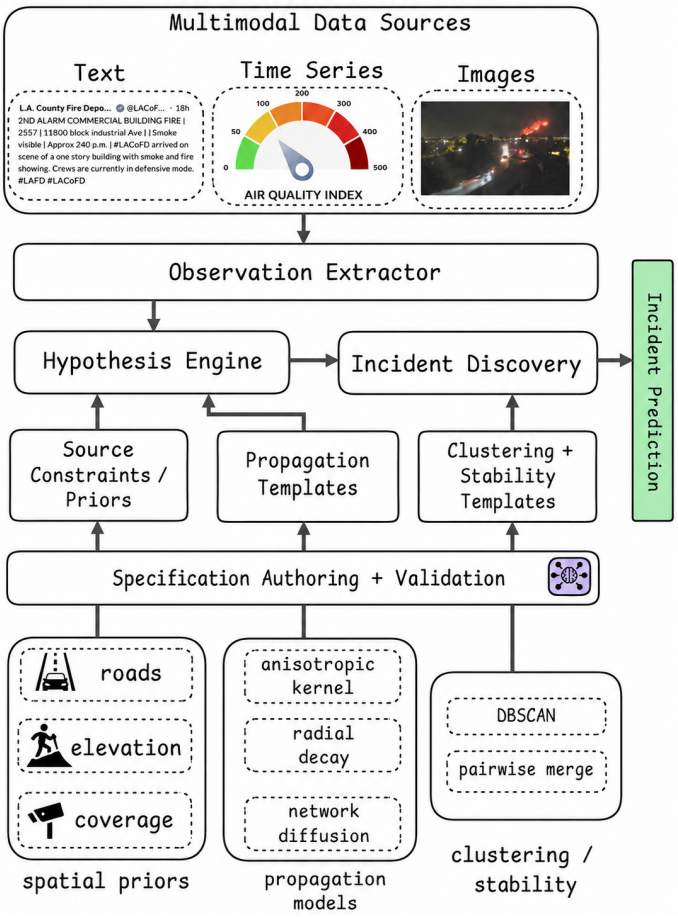


Figure 2.2: System architecture of *IncidentLens*. Semantic observations and incident-type specifications feed a shared online engine that maintains source hypotheses and emits incident records.

ation can apply the full semantic, spatial, and temporal match. Real labels are weaker: they may describe reported or affected locations rather than true sources. For this reason, the real-data evaluation treats type/time matching as the primary weak-label event metric and reports spatial quality separately. These real-data metrics measure compatibility with available reported labels, not complete latent source recovery or a closed-world census of all incidents.

2.3 Methodology

2.3.1 Overview and Terminology

IncidentLens separates two responsibilities that are often entangled in multimodal incident detection. The first is *semantic extraction*: converting modality-specific raw inputs into a common evidence representation. The second is *spatiotemporal hypothesis maintenance*: maintaining a persistent online state of possible incident sources that could explain those observations over space and time.

Figure 2.2 shows the system architecture. Offline, *IncidentLens* constructs a city grid, indexes static context layers, and validates incident-function specifications for each incident type. Online, each raw input record is mapped to a semantic observation. The hypothesis engine updates existing source hypotheses, proposes new hypotheses by reasoning backward from observed effects to possible causes, prunes low-support hypotheses, and passes stable hypotheses to incident discovery. Incident memory is used after discovery to maintain temporal consistency, suppress duplicate reports, and conservatively link records when there is explicit evidence that they are part of a larger incident. Thus, the core detection problem is source-hypothesis maintenance; composition is not required for an incident record to be detected. Figure 2.3 illustrates this observation-to-incident flow.

A raw input record is a timestamped sensor, camera, text, or alert item with a location or footprint. The observation extractor maps it to a semantic observation $o_i = (x_i, t_i, m_i, E_i, I_i, q_i)$, where E_i contains observed effects and I_i contains candidate incident types. A source hypothesis is a candidate latent cause with incident type, source cell, start time, propagation parameters, and support score. Incident-function specifications provide the incident-type-specific source priors, propagation models, clustering distances, and bounded parameters used by the shared online engine. Stable clusters of source hypotheses are emitted as incident records. Each emitted record therefore separates the inferred source region from the affected region and from the locations where evidence was observed.

When evaluating on historical data, we process records in timestamp order to simulate online operation. We refer to this as *stream replay*: past records are fed to the online system as if they were arriving live.

2.3.2 Semantic Observation Extraction

The Observation Extractor maps each raw input record to

$$o_i = (x_i, t_i, m_i, E_i, I_i, q_i),$$

where x_i is a location or footprint, t_i is time, m_i is modality, E_i is a sparse set of ontology effects, I_i is a set of candidate incident types, and q_i is a bounded quality score. Each effect is (ℓ, ρ) with strength $\rho \in [0, 1]$. This representation is evidence, not a final incident decision. Each effect label ℓ is drawn from a fixed semantic effect ontology spanning visible hazards, environmental changes, mobility disruptions, human activity, and operational reports. The extractor records the supporting modality and provenance for every emitted effect.

Extraction has a lightweight anomaly stage followed by schema-bounded semantic parsing. The anomaly stage filters or prioritizes reports using modality-specific signals such as text matches, visual prefilters, and structured time-series anomalies. The observation model then emits only fixed-vocabulary effects, candidate incident types, locations, timestamps, and quality scores. Context-only variables such as wind or temperature may condition later hypothesis updates but do not create incidents by themselves.

The extractor is hypothesis-aware only as a gating mechanism: reports inside the predicted support of an active hypothesis may bypass the anomaly prefilter so weak follow-up evidence is not discarded. Hypothesis context does not assign labels, update weights, or create incident records without an emitted semantic observation. Foundation models, when used, operate only behind the closed schema and are not allowed to emit free-form incident records or consume label-revealing metadata.

2.3.3 Incident-Function Specifications

The main interface between incident-specific reasoning and the shared online engine is the incident-function specification. Each incident type c is associated with

$$F_c = (Q_c, P_c, D_c, \Theta_c),$$

where Q_c is a bounded source-prior score, P_c is a set of effect-specific propagation models, D_c is a clustering distance, and Θ_c contains bounded parameters. This interface lets incident types express different physical and semantic assumptions while using the same online proposal, update, pruning, clustering, and stability logic.

The source prior $Q_c(g \mid \mathcal{X})$ is a proposal score over feasible source cells, not a probability that an incident is present at g. Hard constraints remove physically invalid cells, such as road incidents away from the road network, while soft context features are clipped and normalized within the current reverse-proposal envelope. Thus, Q_c affects which source hypotheses are proposed under a fixed budget, but it cannot by itself create an incident record. We also distinguish source plausibility from observability: sensor coverage and data availability determine whether a modality can support or contradict a hypothesis, but sensor density is not treated as evidence that an incident occurred.

For each effect ℓ, the propagation component provides a model

$$C_\ell(g, t \mid h_j, \mathcal{X}) \to [0, 1],$$

which estimates how strongly effect ℓ should appear at location g and time t under source hypothesis h_j. The clustering distance D_c defines when hypotheses of the same incident type are similar enough to form an incident candidate, using terms such as source distance, start-time difference, affected-region overlap, predicted-effect overlap, and shared supporting observations. The parameter set Θ_c bounds propagation scale, lookback horizon, effect weights, proposal budget, pruning thresholds, clustering thresholds, and stability thresh-

olds.

Specifications are schema-bounded templates rather than arbitrary per-incident programs. They select source constraints, propagation families, clustering terms, and parameter ranges from a fixed catalog. Foundation models may assist with offline or lazy template authoring, but their outputs must be structured and validated: they cannot introduce new effect labels, arbitrary propagation code, report-specific coordinates, or incident records. Runtime values such as source cell and start time are bound only when a hypothesis is instantiated. The evaluated library includes specifications for wildfire, urban fire, road accident, flood, protest, and coordinated public-safety incidents; each selects from the same validated catalog rather than introducing arbitrary runtime code.

2.3.4 Streaming Source-Hypothesis Update

IncidentLens maintains a bounded set of source hypotheses

$$h_j = (c_j, s_j, \tau_j, \theta_j, M_j, w_j),$$

where c_j is incident type, s_j is source cell, τ_j is start time, θ_j contains bounded runtime parameters, M_j is the bound effect-model set, and w_j is a relative support score. This representation is inspired by multi-hypothesis tracking and particle filtering, but the weights are not calibrated posterior probabilities: they are online support scores for competing incident explanations under sparse, displaced, and multimodal evidence.

The streaming update follows a birth–update–prune pattern similar in spirit to multi-hypothesis filtering, but specialized to incident explanations rather than physical targets. When a semantic observation o_i arrives, *IncidentLens*first scores existing hypotheses before proposing new ones. This ordering prevents an observation from immediately reinforcing hypotheses that it created. Existing hypotheses receive positive support when their predicted effects agree with the observation and negative support when a strongly observed effect is inconsistent with their predicted affectedness. New hypotheses are then proposed

only for candidate incident types and source regions whose forward propagation models could plausibly explain the observation.

For each active hypothesis h_j and each observed effect $(\ell, \rho_{i,\ell}) \in E_i$, the engine computes the predicted affectedness

$$a_{j,\ell} = C_\ell(x_i, t_i \mid h_j, \mathcal{X}).$$

Hypothesis support is updated with

$$w'_j = w_j \exp(\eta q_i u_j(o_i)),$$

where η is an update scale, $q_i \in [0, 1]$ is a bounded observation-quality weight, and $u_j(o_i) \in [-1, 1]$ is a clipped compatibility score. The weights are relative online support scores, not calibrated posterior probabilities. They are used to rank, prune, diversify, and cluster hypotheses; incident records are emitted only after persistent cluster-level support.

The compatibility score rewards predicted effects that are observed, gives a small bonus when predicted and observed strengths are close, penalizes strong observed effects in regions where the hypothesis predicts little affectedness, and adds weak support when the observation explicitly names the same incident type. Effects without a corresponding propagation model do not contribute positive evidence. Missing modalities are not treated as negative evidence: an unreported camera, traffic, or text effect does not contradict a hypothesis unless an explicit observed effect is incompatible with the hypothesis prediction.

The quality weight q_i controls the influence of noisy observations. It combines source or modality reliability, localization precision, extraction specificity, and anomaly strength when available, with missing components treated as neutral. Thus, q_i can downweight weak or ambiguous reports, but it cannot create a hypothesis or incident record by itself.

New hypotheses are proposed by reverse-envelope search. For each candidate incident

type $c \in I_i$, observed effect ℓ, and discrete lookback start time τ, the engine enumerates source cells g whose forward model could plausibly explain the observation:

$$C_\ell(x_i, t_i \mid c, g, \tau, \mathcal{X}) \geq \epsilon_{\text{rev}}.$$

Cells outside this envelope are ignored. The remaining cells are filtered by hard source constraints, reweighted by the bounded source prior Q_c, and sampled under a fixed proposal budget with spatial diversity. Algorithm 1 summarizes the per-observation streaming update. Each sampled cell and start time is then bound to the incident-type propagation template to create a source-hypothesis particle.

Some streams are evidence-only rather than proposal sources. For example, weather can strengthen or weaken existing wildfire hypotheses, but it should not by itself create a wildfire source hypothesis. Such support-only sources are used during hypothesis evaluation but skipped during hypothesis birth. This prevents contextual variables from generating incident detections without a concrete observed effect.

After scoring and proposal, weights are normalized. Duplicate hypotheses are merged or suppressed when they have the same incident type, nearby source cells, similar start times, and overlapping effect models. Low-support hypotheses are pruned, and a stratified top-k selection preserves diversity across incident type, source region, and start time. This maintained hypothesis population is the core online state of *IncidentLens*.

Soft data association. *IncidentLens* uses soft many-to-many association: each observation updates all active hypotheses through the compatibility score rather than being assigned to a single track. Event-level association is deferred to clustering, where hypotheses of the same incident type are grouped by source distance, start time, effect overlap, and supporting-report overlap.

Algorithm 1 Streaming source-hypothesis update

Require: source hypotheses H_t, semantic observation o_i, specifications $\mathcal{F}$, cap $N_{\max}$

for all $h_j \in H_t$ **do**
    Score predicted effects with $C_\ell(x_i, t_i \mid h_j, \mathcal{X})$
    Update $h_j.w \leftarrow h_j.w \exp(\eta q_i u_j(o_i))$
end for
for all $c \in I_i$ and $(\ell, \rho) \in E_i$ **do**
    Load $F_c = (Q_c, P_c, D_c, \Theta_c)$
    Compute bounded reverse-support envelope for (ℓ, ρ)
    Enumerate feasible source cells and start times
    Apply source constraints, reweight cells with Q_c, and sample diverse particles
end for
Normalize weights
Merge or suppress duplicate hypotheses
Prune low-support hypotheses and enforce diversity under $N_{\max}$
return updated hypothesis set H_t^+

2.3.5 Incident Discovery and Stability

Periodically, *IncidentLens* clusters source hypotheses of the same incident type using D_c. Each cluster defines a candidate incident

$$R_k = (\hat{c}_k, \hat{S}_k, \hat{\tau}_k, \hat{A}_k(t), M_k, U_k),$$

where $\hat{c}_k$ is type, $\hat{S}_k$ is source region, $\hat{\tau}_k$ is start time, $\hat{A}_k(t)$ is aggregate affectedness, M_k is normalized cluster mass, and U_k summarizes supporting reports, sensors, effects, and locations. Candidates are matched across update rounds by type, source region, and start time so that small particle changes do not produce duplicate records.

A candidate is promoted only after satisfying stability thresholds on cluster mass, age, recent support, decay, and missed updates. Stable records are emitted as

$$\hat{E}_k = (\hat{c}_k, \hat{S}_k, \hat{A}_k(t), \hat{I}_k, \hat{\phi}_k, \hat{p}_k, O_k),$$

where $\hat{I}_k$ is the inferred active interval, $\hat{\phi}_k$ is state, $\hat{p}_k$ is a support score, and O_k is the supporting evidence summary. Here $\hat{S}_k$ denotes the inferred source region, while $\hat{A}_k(t)$

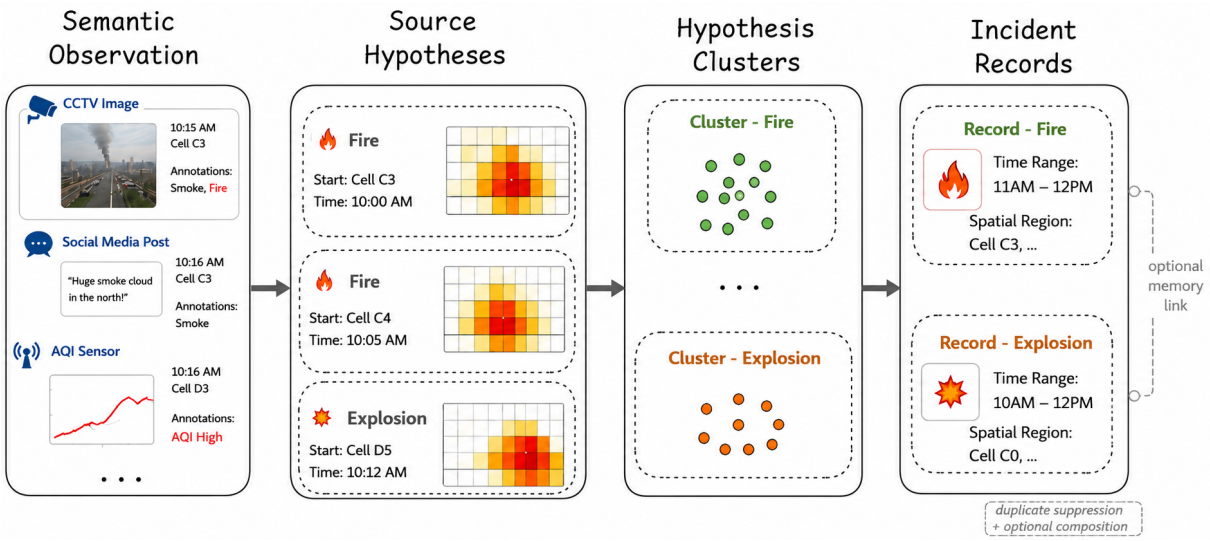


Figure 2.3: Observation-to-record pipeline. Semantic observations update source hypotheses, stable hypothesis clusters become incident records, and incident memory handles duplicate suppression and optional composition.

denotes the affected region. These are distinct from the observed-effect locations in the supporting evidence summary O_k and from any reported location in an external label.

LLM leakage mitigation. When an LLM is used for bounded cluster assessment, it receives only de-identified structured summaries: relative local coordinates/times, support counts, modality/effect categories, stability features, and neighborhood consistency. Prompts omit exact dates, coordinates, place names, incident names, URLs, report IDs, sensor IDs, and agency names, and labels are restricted to the allowed ontology. This cannot remove all pretrained world knowledge, but it reduces cues that would allow the model to retrieve memorized real-world events rather than reason from replayed evidence.

2.3.6 Incident Memory and Composition

Incident memory is a post-discovery module used to maintain temporal consistency and suppress duplicates. When a stable record is emitted, it is compared with recent records of compatible type, source or affected region, active interval, and supporting evidence. Compatible records update the existing memory entry rather than producing duplicate

reports.

Composition is optional and conservative. After event-level records already exist, memory may create a *part-of* link when heterogeneous incidents have explicit evidence of a broader event, such as fire plus road closure or shooting plus bomb threat. These links do not create the initial low-level detections and do not increase their detection scores in the main evaluation. Composition is therefore evaluated separately from low-level incident detection.

2.4 Evaluation

The evaluation tests one central claim: maintaining source-level hypotheses improves event-level detection when observations are sparse, indirect, and multimodal. We treat localization, timeliness, composition, and runtime as supporting properties rather than equally strong claims, because their observability differs between synthetic and real data.

We first describe the datasets, labels, metrics, and baselines. We then report synthetic low-level detection, real-world weak-label detection, focused ablations, synthetic composition, and stress/runtime results. The synthetic experiments provide the cleanest test of source and affected-region recovery because ground truth is known. The real-data experiments test whether the same design improves detection against weak external incident records, with spatial quality reported as a diagnostic rather than as the primary matching criterion.

2.4.1 Datasets and Labels

Label isolation and text provenance. For both the synthetic and real-data experiments, label-construction data are used only for evaluation and are never provided as input to *IncidentLens* or to any baseline. In the real dataset, news articles are used to construct incident labels, but the replay stream contains only non-news observations, including sensors and operational text sources such as social posts, citizen reports, and of-

ficial alerts. In the synthetic dataset, incident labels are derived from simulator plans and ground-truth metadata, but these plans and labels are not exposed to any detector during replay. Thus, in both settings, the systems are evaluated against held-out labels whose provenance is disjoint from the observation streams used for detection. The synthetic ablations further separate physical-sensor and operational-text performance.

Real-world data. We evaluate on Los Angeles observations from January 2025 to May 2026. Replay inputs include Caltrans CCTV (noa, b), ALERTCalifornia cameras (noa, a), PurpleAir air quality(noa, g), OpenWeatherMap weather (noa, d), Caltrans PeMS traffic (California), CitizenApp alerts (noa, c), and verified public-safety accounts on X (X Corp., 2026). GDELT/news metadata (noa, f) is used only for label construction and is not replayed to *IncidentLens* or to any baseline. The dataset has intermittent missing intervals and unequal temporal coverage across sources, reflecting realistic heterogeneous sensing conditions. Source coverage and missing intervals are retained in the released dataset metadata so that observability can be distinguished from detector error.

Weak-label construction. Labels are constructed offline from external reports using an LLM-assisted parser and deduplicated by semantic, spatial, and temporal overlap. We treat them as weak records of reported incidents rather than complete ground truth. Label-construction prompts, deduplication rules, and provenance records are retained with the evaluation artifacts but are not inputs to any detector.

Synthetic data. Synthetic incidents provide controlled multimodal scenarios with complete source-region, affected-region, timing, and provenance ground truth. The synthetic dataset contains 197 singular cases and 97 composite cases, spanning 294 simulator runs and 391 localized incidents.

Each synthetic case is grounded in a Los Angeles-area region and emits incident-relevant observations from edited CCTV / ALERTCalifornia imagery, structured time series such as air quality, weather, and traffic speed/occupancy, and textual public-report-

like sources such as news, citizen reports, and social media. We use the synthetic dataset for settings where real-world ground truth is difficult to obtain at the required granularity, including stress tests over observability, sensor dropout, noise, indirect-effect displacement, and missing modalities. Although the simulator and detector share a broad incident/effect ontology, the detector receives only emitted observations and sensor metadata, not simulator state, hidden incident regions, generation prompts, or ground-truth labels. The simulator configuration, generation prompts, and source-level provenance are preserved in the project artifacts for reproducibility.

The synthetic benchmark is intended to test controlled observability rather than to fully replace real deployment evidence. Because the simulator and detector necessarily share a broad incident/effect ontology, we interpret synthetic results as evidence that the hypothesis-maintenance design works under known latent incidents, not as proof that the exact templates will transfer unchanged to all cities or incident families. We reduce direct circularity by withholding simulator plans, hidden incident state, ground-truth regions, generation prompts, and labels from all detectors during replay.

2.4.2 Metrics and Matching Protocol

For event-level detection, we report precision, recall, F_1, F_2, and false positives. We emphasize F_2 because missed incidents are more costly than a small number of extra candidates, while false positives are still penalized. Synthetic evaluation uses one-to-one matching between predictions and simulator labels. A correct synthetic match requires compatible incident type, temporal overlap, and spatial compatibility, following the noisy-label matching protocol from Section 2.2.3. A wrong-type match is counted as a false positive for the predicted type and a false negative for the ground-truth type. We distinguish four spatial quantities throughout the evaluation: *reported location* is the location named by an external label or report; *observed-effect location* is where evidence is sensed; *inferred source region* is the latent origin estimated by *IncidentLens*; and *affected region* is the region where effects are observed or predicted. Synthetic labels contain source and

affected-region ground truth, while real labels often contain reported or affected locations rather than true sources.

Spatial metrics are reported in two forms. First, localization error is the centroid distance between the predicted inferred source region and the ground-truth or reported label region. For methods that do not estimate an explicit source, we use the predicted affected region or support points as a source-location proxy. Second, scenario-level spatial metrics compare the union of all same-type predicted affected regions in a run against the corresponding ground-truth affected region. We report spatial coverage recall,

$$\mathrm{SRec} = \frac{|\hat{A} \cap A|}{|A|},$$

and spatial intersection-over-union,

$$\mathrm{SIoU} = \frac{|\hat{A} \cap A|}{|\hat{A} \cup A|}.$$

Thus, S rec. measures whether the method covers the true affected region, while S IoU also penalizes over-predicted area. Temporal IoU is computed analogously by merging same-type predicted and ground-truth active intervals before computing intersection-over-union. For each ground-truth incident type in a run, we union all same-type predicted regions and compare that union against the corresponding ground-truth region.

For real data, labels are weak external incident records rather than complete simulator ground truth. The primary real-data results therefore use type/time non-spatial matching: predictions are matched to labels by canonical incident type and temporal compatibility, while spatial quality is reported separately. We pool predictions across date folders before matching because real incidents can span multiple days. Timeliness is measured relative to the earliest external article timestamp,

$$\mathrm{Delay}(\hat{E}, \tilde{E}) = t_{\mathrm{detect}}(\hat{E}) - t_{\mathrm{article}}(\tilde{E}),$$

where negative values indicate detection before the article used for label construction. We also report pre-article recall (Pre-rec.), the fraction of all labels matched by a detection before the earliest article timestamp, and pre-article fraction (Pre-frac.), the fraction of matched true positives detected before that timestamp.

Composition metrics are reported separately for the synthetic composition task. Because the reported composition table evaluates positive composition scenarios, we interpret its detection metrics as positive-case composition recall and its type metric as the accuracy of the predicted high-level incident type among detected true positives.

2.4.3 Baselines

Baselines. We compare against five streaming baselines that represent common alternatives to source-hypothesis reasoning. *Direct observation* emits an incident whenever a single semantic observation exceeds a confidence threshold, testing whether incident detection can be solved by local report classification alone. *Space-time clustering* links same-type semantic observations within a fixed spatial and temporal radius, providing a non-parametric clustering baseline for repeated local evidence. *Text-only clustering* applies the same idea only to operational text streams, isolating the strength of text without sensor fusion. *Hotspot scan* is our strongest general classical baseline: it searches for local concentrations of same-type semantic evidence over grid/time windows, capturing the main intuition behind scan-statistic and hotspot detectors while remaining applicable to sparse heterogeneous reports. Finally, *late-fusion voting* is included as a conservative multimodal-agreement diagnostic: it emits an incident only when multiple modalities support the same incident type in a local space-time window. Its role is to test whether simple cross-modal agreement is sufficient, not to represent the strongest possible detector.

We do not include full inverse-model baselines or a literal Kulldorff SaTScan implementation as primary comparators because they are not uniformly applicable to our setting. Inverse methods such as wind-based smoke/PM2.5 back-projection or traffic-network back-tracing require incident-specific physics, calibrated auxiliary variables, and

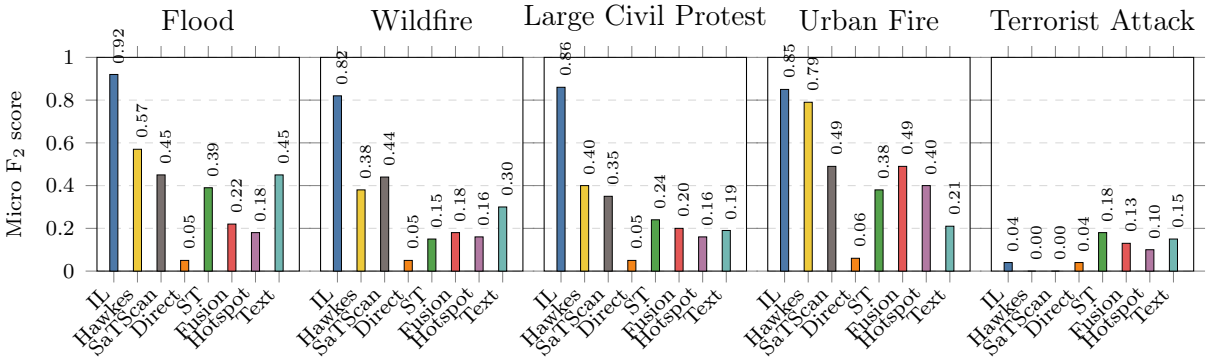


Figure 2.4: Scenario-conditioned micro-F_2 by selected ground-truth incident category on the synthetic low-level evaluation. Scores penalize extra predictions emitted in runs of the same ground-truth incident type.

modality-specific assumptions; they are valuable for particular incident families, but do not naturally cover the full ontology of fires, floods, road disruptions, protests, public-safety threats, and composite incidents. Similarly, Kulldorff-style SaTScan assumes a formal spatial or space-time scan statistic over count or case-control streams with an explicit background model, whereas our observations are sparse semantic outputs from heterogeneous sensors and operational text. We therefore use the hotspot scan as the closest general-purpose scan-statistic-style comparator and treat inverse-model baselines as important incident-specific extensions.

All baselines use the same set of input data and the same event-level matching protocol. When a baseline consumes semantic observations, it receives the same extractor outputs as *IncidentLens*. Thresholds are selected on the same validation period. Text-only baselines receive only text available by replay time t, and generic-propagation baselines use the same particle budget as *IncidentLens*.

2.4.4 Synthetic Low-Level Detection Results

Figure 2.4 breaks down micro-F_2 for selected synthetic incident categories. The figure shows that *IncidentLens* is strongest on flood, wildfire, protest, and urban-fire scenarios, while terrorist-attack scenarios remain difficult for all methods.

Table 2.2 reports aggregate synthetic low-level incident detection. *IncidentLens* achieves the best event-level precision, recall, F_1, and F_2, while producing far fewer false positives per run than the direct, clustering, fusion, hotspot, and text-only baselines.

2.4.5 Ablations

The ablations show that the most important contributors are reverse proposal, multimodal evidence, and incident-specific grouping. Removing reverse proposal causes the largest spatial degradation: S rec. drops from 0.798 to 0.172, indicating that proposing sources only near observed effects is insufficient when observations are indirect or displaced. The modality ablations also substantially degrade event-level performance, with F_2 dropping from 0.685 to 0.526 for sensor-only input and to 0.578 for operational text-only input, showing that neither modality family is sufficient across incident types. Generic clustering/stability also reduces F_2, suggesting that incident-specific grouping criteria help convert noisy hypothesis sets into stable incident records.

The propagation ablation is less conclusive in this dataset: generic propagation has similar F_2 and spatial recall to the full model, and in this run slightly improves both, although it lowers temporal IoU. We therefore do not interpret this ablation as independently validating incident-specific propagation templates. Instead, the strongest ablation evidence supports reverse source proposal, multimodal evidence, and incident-specific clustering/stability; propagation templates remain part of the system abstraction and appear most useful for temporal coherence.

2.4.6 Incident Composition

Incident composition. We also evaluate synthetic incident composition, where two lower-level sub-incidents may imply a higher-level incident such as wildfire, urban fire, civil protest, or terrorist incident. This task is not simply same-type linking: a composite event may combine heterogeneous children, such as fire plus road closure or shooting plus bomb

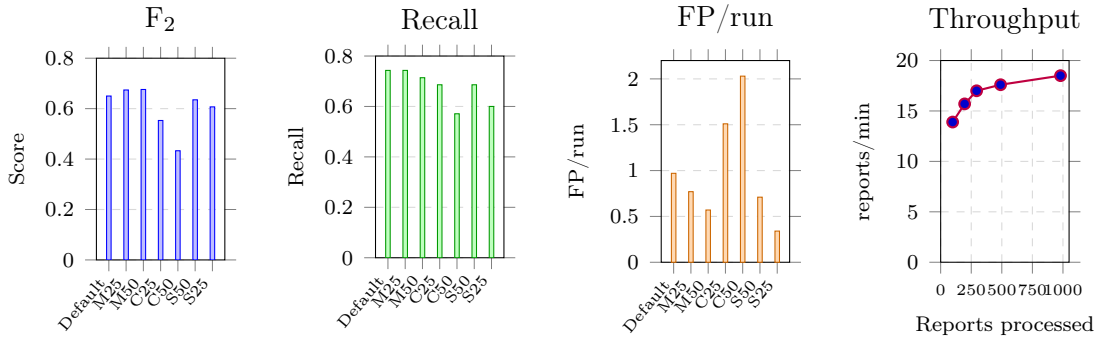


Figure 2.5: Stress robustness and throughput. The first three panels vary missing observations, corrupted observations, and sensor density; the final panel reports controlled end-to-end throughput.

threat. Table 2.4 compares *IncidentLens* with proximity-only and same-type temporal-overlap linkers on positive composition scenarios. The *proximity-only linker* predicts a composite incident whenever two lower-level incident records are within a fixed spatial distance, ignoring type and temporal overlap. The *same-type temporal-overlap linker* predicts a composite incident only when two lower-level records have the same incident type and overlapping active intervals. These results measure positive-case composition recall and high-level type accuracy, not the false-link rate on unrelated low-level incidents.

2.4.7 Stress Robustness and Runtime

Figure 2.5 reports stress robustness and controlled throughput. The first three panels show detection quality under missing observations, corrupted observations, and sensor-density stress settings on the 35-run scalability/stress subset. The throughput panel essentially duplicates reports for `wildfire1` with anomaly and observation-model processing enabled. M25/M50 remove 25%/50% of observations, C25/C50 corrupt 25%/50% of observations by replacing with a different effect/incident, and S50/S25 retain 50%/25% of sensors.

The throughput experiment is a controlled stress test rather than a full deployment benchmark: it duplicates one representative wildfire stream with anomaly and observation-model processing enabled. Table 2.5 shows that the symbolic hypothesis engine is not the dominant bottleneck; most time is spent in observation/anomaly processing and unin-

strumented overhead. Deployment-scale throughput would therefore depend on filtering, caching, batching, parallelism, and the number of heavy model calls.

2.4.8 Real-World Detection Results

Table 2.6 reports aggregate real-data detection under type/time non-spatial matching. Under this weak-label protocol, IncidentLens achieves the best precision, F_1, and F_2, while substantially reducing false positives relative to direct observation, space-time clustering, and hotspot scanning. Direct observation has higher recall but emits many more false positives. These results show improved compatibility with reported incident records, but not full spatial source recovery, because spatial matching is not part of the primary real-data event match. Late fusion obtains zero true positives because real streams are sparse and asynchronous, so we treat it as a conservative agreement diagnostic rather than a strong real-data baseline.

Because real labels often provide reported or affected locations rather than true source regions, localization error is source/proxy-to-label centroid distance, while S IoU measures overlap between predicted affected regions and reported or affected label regions. S IoU and T IoU are scenario-level diagnostics; Loc. err. and delay are shown only for matched true positives, as shown in Table 2.7.

Overall takeaways. Across synthetic low-level incidents, *IncidentLens* improves event-level precision, recall, and F_2 while reducing false positives by more than an order of magnitude relative to direct observation and simple clustering baselines. The spatial metrics are more nuanced: *IncidentLens* achieves high spatial recall but modest spatial IoU, indicating that it often covers the true affected region while also over-covering extra area.

The synthetic category breakdown reveals an important failure mode. Terrorist attack scenarios are difficult because the available observations often describe visible effects rather than intent or cause. Camera and sensor evidence may show fire, debris, smoke,

road blockage, or emergency vehicles, while operational text may only describe response effects. In such cases, the evidence is often more directly compatible with urban fire, road disruption, or generic public-safety activity than with the causal label "terrorist attack."

On real data, *IncidentLens* trades a small reduction in recall relative to direct observation for substantially better precision, F_1, F_2, localization error, and false-positive count. The negative median report delay indicates that many matched detections occur before the earliest external article timestamp, supporting the provenance claim that news articles are used for labels rather than replayed as detector inputs.

2.5 Discussion and Limitations

Discussion and limitations. *IncidentLens* is intended for observable incidents, not all urban events. Real-world labels are weak external records, and some may be outside sensing coverage, administrative rather than observable, or too underspecified to evaluate from available streams. Likewise, unmatched predictions may be false positives, duplicates, or plausible unreported incidents. We therefore interpret real-data precision and recall as measurements against reported labels, not as a complete census of city events.

We reduce label circularity by using news articles only for label construction; they are not replayed to *IncidentLens* or any baseline. Detector inputs are restricted to sensors and operational text such as social posts, citizen reports, and official alerts. This separation prevents direct reuse of label-construction articles, but does not remove all text-related bias: text-rich incidents may remain easier to detect than sensor-only incidents.

Foundation models introduce leakage and reproducibility risks. We mitigate these risks by using them only behind closed schemas for semantic extraction or bounded cluster assessment, and by excluding exact incident names, label articles, simulator ground truth, and other direct label cues from prompts. This reduces, but cannot eliminate, pretrained world-knowledge effects. Similarly, hypothesis weights are relative support scores for ranking, pruning, and clustering, not calibrated incident probabilities.

Synthetic data provides true source regions, affected regions, timing, and controlled observability, but may not capture the full ambiguity and reporting bias of real incidents. Some scenarios also expose limits of effect-based inference: terrorist-attack cases are difficult when observations show visible effects such as fire, smoke, debris, road blockage, or emergency response rather than intent or cause.

Finally, *IncidentLens* contains bounded design choices, including effect weights, propagation scales, source-prior weights, proposal budgets, pruning thresholds, clustering thresholds, and stability windows. Priors are locally normalized, proposals preserve spatial diversity, missing modalities are not treated as contradictions, and incident records require persistent cluster-level support. Deeper stress tests under deliberately misspecified priors, alternative propagation templates, and broader deployments remain future work.

Ethics and privacy. The system is intended to produce incident-level situational-awareness records, not person-level surveillance. The evaluation uses public or aggregated streams where possible, stores semantic effects rather than raw person identities, avoids tracking individuals across cameras or social media accounts, and reports only incident type, time, location, affected area, and supporting evidence. These choices reduce but do not eliminate surveillance risks; any deployment should include access controls, retention limits, audit logs, human oversight, and clear limits to incident-response or public-interest use.

2.6 Related Works

Urban sensing and anomaly detection. Urban sensing systems have long used fixed and mobile sensors to detect local anomalies in cities. Traffic systems detect freeway incidents from loop-detector occupancy and flow patterns using rule-based, neural, or probabilistic models (Payne and Tignor, 1978; Ritchie and Cheu, 1993); video systems

detect accidents and anomalies in surveillance or dashcam streams (Pawar and Attar, 2022; Sultani et al., 2018; Liu et al., 2018; Shah et al., 2018; Yao et al., 2020); and mobile sensing systems detect road hazards or crashes from vehicle- and phone-mounted sensors (Eriksson et al., 2008; Mohan et al., 2008; White et al., 2011). Other urban sensing work detects acoustic or social events, such as construction noise, sirens, gunshots, or local tweet bursts (Salamon et al., 2014; Cartwright et al., 2020; Zhang et al., 2016). These systems are effective at recognizing that a particular sensor, road segment, camera view, trajectory, or modality is abnormal. In contrast, *IncidentLens* treats incidents as latent explanatory objects: observations may be displaced, delayed, modality-specific, or partial, and the core task is to infer which observations share a cause and where that cause likely originated.

Source localization and inverse urban sensing. Our work is also related to source localization, inverse modeling, and social sensing, which explicitly infer hidden causes from indirect observations. Prior work localizes pollutant releases using Bayesian inversion and dispersion models (Keats et al., 2007; Chow et al., 2008), estimates wildfire smoke emissions and spread from satellite, weather, and atmospheric models (Saide et al., 2015; Huot et al., 2022), detects disease or flood clusters from spatial reports (Kulldorff et al., 2005; Arthur et al., 2018), localizes earthquakes or local events from social media (Sakaki et al., 2010; Zhang et al., 2017a), and identifies shooters or traffic root causes from acoustic or mobility observations (Simon et al., 2004; Maroti et al., 2004; Chawla et al., 2012). Recent learning-based systems further predict traffic accidents or risks over fixed spatial graphs using heterogeneous urban features (Yuan et al., 2018; Zhou et al., 2020; Yu et al., 2021). These approaches reason backward from effects to causes, but typically assume the incident class, propagation mechanism, sensing graph, and relevant observations are known in advance. *IncidentLens* instead compares multiple incident hypotheses under uncertain event type, modality availability, spatial support, and observability.

Probabilistic association and spatiotemporal evidence fusion. *IncidentLens* is related to probabilistic tracking and data-association methods, which maintain uncertainty over latent states and ambiguous observations. MHT maintains competing measurement to track association histories (Reid, 2003), JPDA marginalizes over uncertain assignments (Fortmann et al., 1983), and PHD/CPHD filters model an unknown number of targets using random-finite-set Bayes filtering (Mahler, 2003; Vo and Ma, 2006; Vo et al., 2007). Particle-filter and MCMC variants further support nonlinear dynamics, interacting targets, and variable target counts (Vo et al., 2005; Khan et al., 2005; Oh et al., 2004). These methods are relevant because they preserve multiple explanations under streaming uncertainty, but their hypotheses usually represent physical targets or target sets. In contrast, *IncidentLens* represents hypotheses as latent incident explanations: an incident type, source region, start time, propagation behavior, and supporting or contradicting multimodal evidence. *IncidentLens* also builds on multisensor fusion and spatiotemporal Bayesian filtering. General fusion frameworks study how to align, associate, and combine uncertain observations from multiple sources (Hall and Llinas, 2002), while belief-function methods provide alternative mechanisms for combining partial evidence (Dempster, 2008). Spatiotemporal filters have been applied to urban and environmental settings such as traffic-volume prediction (Okutani and Stephanedes, 1984), disaster mobility prediction (Sudo et al., 2016), and remote-sensing image fusion (Xue et al., 2017). Unlike these systems, *IncidentLens* does not simply fuse measurements of a known target or field. It uses semantic observations, incident-specific source priors and propagation models, backward proposal of candidate causes, and stability-based clustering to produce explanatory city-scale incident records. Thus, *IncidentLens* is closer to abductive spatiotemporal evidence fusion for urban incidents than to conventional multitarget tracking or single-field sensor fusion.

Complex and compositional event detection. Complex event processing (CEP) systems compose low-level events into higher-level patterns using user-specified rules or queries. Cayuga (Demers et al., 2007), for example, provides an event language and

automata-based execution model for continuous queries over high-throughput streams. Spatiotemporal logic languages (Ma et al., 2020a; Haghighi et al., 2015) add primitives for spatial and temporal selection, aggregation, and ordering. Recent work has combined neural models with CEP (Zhang et al., 2024) or used neurosymbolic architectures to connect neural perception with symbolic event programs (Xing et al., 2020; Vilamala et al., 2023; Liu et al., 2019). These systems are closest to our goal of connecting perception and symbolic event reasoning, but they generally require predefined event programs or known event patterns. *IncidentLens*differs by first inferring primitive incident records from sparse multimodal evidence and then reasoning over merge, composition, correlation, and co-occurrence relationships among those records.

Smart-city event platforms, event schemas, and foundation models. Smart-city middleware and analytics frameworks have integrated heterogeneous IoT and social streams for event detection and decision support. CityPulse, for example, provides a large-scale data analytics framework for smart-city streams and applications (Ali et al., 2016). Such systems motivate the need to combine urban data sources, but they generally emphasize stream integration, querying, or application-level analytics rather than maintaining latent source hypotheses that explain displaced multimodal evidence.

Event-schema induction in NLP provides another view of compositional events. Prior work extracts events and induces schemas from text (Huang et al., 2016), represents temporal complex event schemas as graphs (Li et al., 2021), and uses LLMs to induce open-domain hierarchical event schemas (Li et al., 2023). MultiHiEve (Ayyubi et al., 2024) extends this idea to video and text, extracting fine-grained event hierarchies across modalities. These works reason about event structure, but primarily from textual or curated multimodal event mentions rather than streaming city sensors with incomplete observability.

Recent work has also explored LLMs and foundation models for urban sensing and management. UrbanGPT (Li et al., 2024) and UrbanLLM (Jiang et al., 2024) adapt LLMs

to urban spatiotemporal prediction and planning, while ReFound (Xiao et al., 2024) learns multimodal urban region representations and AgentSense (Guo et al., 2026) uses LLM agents for participatory urban-sensing task assignment. Closer to incident detection, Liao et al. (Liao et al., 2024) use vision-language models for traffic accident anticipation and localization. These systems demonstrate the promise of foundation models for urban reasoning, but our use of LLMs is more constrained: they map inputs to a fixed ontology or assess bounded structured summaries, while the main incident inference state is maintained by the symbolic hypothesis engine.

2.7 Conclusion

This chapter presented *IncidentLens*, a streaming system for discovering and localizing city-scale complex events from sparse, indirect, and multimodal urban observations. Unlike approaches that treat local anomalies independently or assume that the incident type, propagation model, and relevant observations are known in advance, *IncidentLens* maintains competing hypotheses over incident type, origin, extent, progression, composition, and observation membership. As new observations arrive, it revises which hypotheses remain plausible and how observations are associated with them. Incident-specific source priors, propagation models, and clustering criteria allow different incident families to be represented within a shared online inference engine.

Across synthetic and real Los Angeles incidents, this structure improves event-level detection and reduces false positives relative to observation-level, clustering, hotspot, and text-centered baselines. The results also identify limits that remain important for future work. Weak real-world labels make source localization difficult to validate, broad affected-region predictions can trade spatial precision for recall, and rare or poorly observed incident families remain difficult to distinguish from correlated background activity.

Within the dissertation, *IncidentLens* addresses the interpretation pillar by showing how distributed urban effects can be assembled into a revisable explanation of what inci-

dent is occurring, where it originated, how far it extends, and which observations belong to it. Constructing an incident explanation, however, does not determine which observations or event attributes should be disclosed to an application. The next chapter therefore moves from event interpretation to application-visible representation and asks how a sensing system can release the information needed for a task while satisfying contextual constraints.

Table 2.1: Sensing-oriented incident terminology grounded in NIMS/ICS incident complexity.

Term	NIMS/ICS grounding	Sensing interpretation	Examples
Small-scale incident	Paper-specific term for lower-complexity incidents, roughly corresponding to Type 3–5 incidents on the NIMS/ICS complexity scale (Federal Emergency Management Agency, 2021, 2018).	Limited spatial and temporal footprint; likely to be observed by one sensor or a small cluster of nearby sensors.	Home crime; small fire; local road closure.
Large-scale incident	Paper-specific term for higher-complexity incidents, roughly corresponding to Type 1–2 incidents, which involve multiple operational periods, substantial resources, and potentially multijurisdictional or multidisciplinary response (Federal Emergency Management Agency, 2021, 2018).	Broader spatial and temporal footprint; likely to produce distributed observations across many sensors, regions, and modalities.	Wildfire; earthquake; major flood.
Incident complex	NIMS-defined grouping of two or more individual incidents located in the same general area and managed by a single Incident Commander or Unified Command (Federal Emergency Management Agency, 2017b).	Composite incident whose evidence may appear as multiple related spatial or temporal clusters that should be interpreted jointly.	Multiple nearby wildfires; related explosions and fires; storm-related closures and rescues.

Method	Prec. ↑	Rec. ↑	F_1 ↑	F_2 ↑	FP/run ↓	S rec. ↑	S IoU ↑	T IoU ↑
IncidentLens	**0.438**	**0.762**	**0.556**	**0.664**	**0.98**	**0.748**	0.077	**0.589**
SaTScan bg.	0.116	0.370	0.176	0.257	2.83	0.366	0.143	0.377
Hawkes detector	0.142	0.381	0.207	0.285	2.30	0.371	0.154	0.303
Direct obs.	0.006	0.630	0.013	0.031	96.76	0.255	0.177	0.405
ST clustering	0.043	0.619	0.081	0.169	13.73	0.451	0.216	0.404
Late fusion	0.038	0.564	0.071	0.150	14.25	0.525	**0.217**	0.390
Hotspot scan	0.030	0.569	0.057	0.123	18.56	0.593	0.111	0.391
Text only	0.041	0.519	0.076	0.155	12.20	0.199	0.096	0.337
Generic propagation	0.000	0.000	0.000	0.000	1.31	0.000	0.000	0.000

Table 2.2: Aggregate synthetic low-level incident detection results. Event metrics are micro-averaged; S rec., S IoU, and T IoU use scenario-level union accounting.

Variant	F_2	ΔF_2	S rec.	ΔS rec.	T IoU	ΔT IoU
Full	0.685	–	0.798	–	**0.718**	–
Generic all	0.668	-0.017	0.806	+0.008	0.642	-0.076
Gen. cluster	0.644	-0.042	0.784	-0.014	0.714	-0.004
Gen. prop.	**0.690**	+0.004	**0.815**	+0.017	0.673	-0.045
No reverse	0.622	-0.064	0.172	-0.626	0.574	-0.144
Sensor only	0.526	-0.159	0.655	-0.143	0.592	-0.126
Text only	0.578	-0.107	0.562	-0.236	0.603	-0.115

Table 2.3: Synthetic low-level ablations. Delta columns report change relative to the full model.

Method	Detected	Rec.	F_2	Strict type
IncidentLens	**64/66**	**0.970**	**0.976**	0.938
Proximity linker	54/66	0.818	0.849	**1.000**
Type+time linker	47/66	0.712	0.756	0.000

Table 2.4: Synthetic incident-composition results over positive composition scenarios. Type accuracy is computed over detected true positives.

Component	Time/report	Share
Hyp. proposal	0.029 s	0.8%
Hyp. evaluation	0.015 s	0.4%
Clustering	0.018 s	0.5%
Prediction	0.227 s	6.3%
Composition	0.055 s	1.5%
State/output write	0.075 s	2.1%
Obs./anomaly + overhead	**3.178 s**	**88.3%**
Total wall clock	3.598 s	100.0%

Table 2.5: Average per-report runtime breakdown for the controlled timing experiment with anomaly and observation-model processing enabled.

Method	Prec. ↑	Rec. ↑	F_1 ↑	F_2 ↑	FP ↓	TP	FN	Pre-rec. ↑
IncidentLens	**0.177**	0.586	**0.272**	**0.401**	**79**	17	12	**0.483**
Direct obs.	0.037	**0.621**	0.069	0.148	473	**18**	**11**	0.414
ST clustering	0.018	0.207	0.033	0.067	326	6	**11**	0.069
Late fusion	0.000	0.000	0.000	0.000	6	0	29	0.000
Hotspot scan	0.013	0.172	0.025	0.051	374	5	24	0.034

Table 2.6: Aggregate real-data low-level detection under type/time non-spatial matching. TP, FP, and FN are absolute counts over the pooled real-data evaluation.

Method	Loc. err. km ↓	S IoU ↑	T IoU ↑	Pre-frac. ↑	Med. delay h ↓	P25/P75 delay h
IncidentLens	**13.3**	**0.099**	0.149	**0.824**	**-29.2**	-48.7 / -4.2
Direct obs.	24.5	0.004	**0.205**	0.667	-17.3	-36.0 / 6.0
ST clustering	33.3	0.014	0.048	0.333	90.8	-1.9 / 367.4
Late fusion	–	0.037	0.000	–	–	– / –
Hotspot scan	30.4	0.005	0.045	0.200	38.3	8.8 / 165.8

Table 2.7: Real-data localization, overlap, and report-delay diagnostics under type/time non-spatial matching. Localization and delay are computed over matched true positives.

CHAPTER 3

Constructing Privacy-Aware Outputs for Complex-Event Applications

Chapter Overview

The previous chapter addressed uncertainty about which event best explains distributed evidence. This chapter shifts to a different boundary: once sensor data can support an application-relevant fact, what representation of that fact should the application receive? The distinction is important because event utility rarely requires every detail present in the underlying media. An occupancy application may need a count rather than identifiable video, and a safety application may need a sound-event label rather than speech content.

Chapter 1 introduced *PrivacyOracle*, which explored LLM-assisted privacy firewalls for verifying information flows, identifying sensitive states, and recommending transformations. The work established that privacy mediation should be able to modify a flow rather than merely allow or deny raw access, but it left final decisions too dependent on unstructured model judgments and informal tool descriptions. *privMediator* replaces that design with explicit application contracts, typed operator effects, residual information flows, layered contextual-integrity rules, and least-revealing feasible selection. In the dissertation's event-construction framing, this chapter asks *what may an application learn*

about what happened? It treats semantic outputs as governed interfaces between sensing infrastructure and applications, without claiming that every transformed output is itself a complex event.

3.1 Introduction

This chapter studies how to construct privacy-aware sensor outputs that preserve the event-relevant information needed by downstream applications, including complex-event applications. A downstream task may need a count, pose keypoints, a sound-event label, a redacted stream, or a final event record without requiring access to the identifiable video or intelligible speech from which that information was derived. The challenge is not only whether an application may access sensor data, but how to construct a processing pipeline and select an application-visible representation that preserves the required utility while limiting inappropriate disclosure.

This problem is especially difficult in *privacy-sensitive shared smart environments*: sensor-instrumented spaces in which privacy decisions would ideally be negotiated among multiple stakeholders, but such negotiation is difficult to perform in practice. The individuals captured by sensors may have limited ability to negotiate with the party that configured the space, such as a building operator, employer, or landlord. They may also have limited ability to negotiate with one another, even though one sensing decision can simultaneously affect regular occupants, employees, visitors, and bystanders. We focus on infrastructure and environment-integrated sensors controlled by a space operator rather than arbitrary personal devices carried by visitors.

Conventional privacy mechanisms only partially address this setting. Access-control policies (Android Developers; Apple Inc.), consent flows (European Parliament and Council of the European Union, 2009), privacy dashboards (Das et al., 2018), and privacy labels (Emami-Naeini et al., 2020) work best when data collection is tied to a user-facing interface such as an account, browser, phone, app session, or personal device. Smart spaces

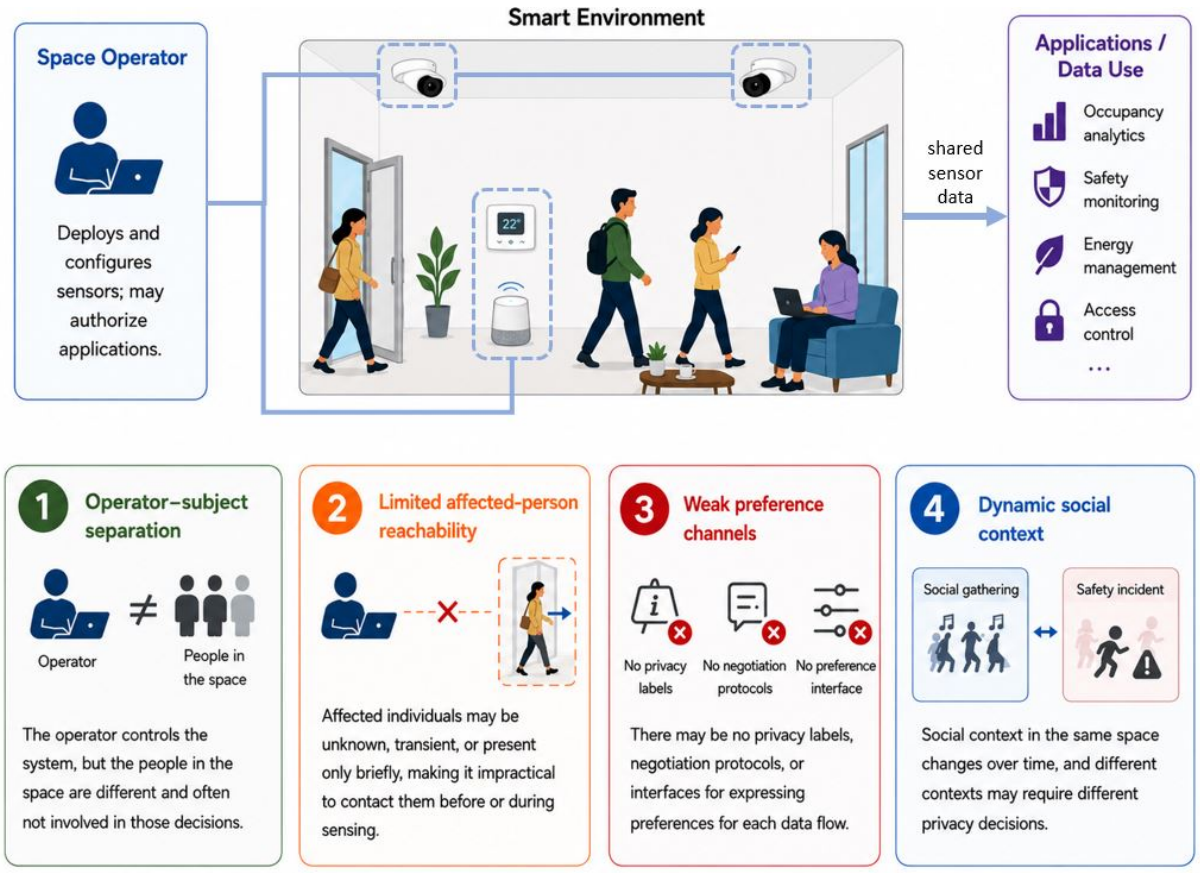


Figure 3.1: Privacy-sensitive shared smart environments combine operator–subject separation, limited affected-person reachability, weak preference channels, and dynamic social context.

collect data because people are present in a physical environment, not because each person intentionally interacted with a particular application or control surface. Notice-and-choice assumes that people can observe, understand, and act on notices; transparency mechanisms can describe sensing practices but do not determine which technical representation should be released; and smart-home negotiation mechanisms often rely on repeated relationships or direct interaction among household members (Zhou et al., 2024). These assumptions weaken in spaces with transient occupants, bystanders, changing purposes, and institutional operators.

Within the dissertation framing, this chapter does not claim that every privacy-preserving output is itself a complex event. Instead, it addresses how complex-event

applications and other sensing tasks obtain event-relevant information through an application-visible representation. A representation decision determines whether an application receives raw observations, transformed observations, semantic features, or an event-level result. Those alternatives differ in utility, residual disclosure, bandwidth, and downstream compatibility, and their appropriateness depends on the sender, subject, recipient, purpose, transmission principle, and context of the resulting information flow.

We address this representation problem with *privMediator*, a trusted privacy mediator that constructs sensor outputs for complex-event and other downstream applications in privacy-sensitive shared smart environments. Conceptually, *privMediator* is a virtualized sensing-hub service isolated on a secure platform for a particular space. It sits between infrastructure sensors and downstream applications, receives application requests and contextual information, and decides at runtime which transformed representation of a sensor stream should be released. The mediator prioritizes application utility subject to configured privacy constraints: it first searches for preprocessing pipelines that satisfy the application's utility contract, then selects the least-revealing output representation accepted by the active contextual-integrity rules. When no representation satisfies both requirements, *privMediator* rejects the application request, which a deployment may route to consent, human review, emergency escalation, or degraded-task modes.

This decision is context-dependent. An occupancy application may need a count in one context, coarse presence in another, and no access in a third. A security application may require detecting visitors at an entryway while having no need for continuous identifiable video. An audio safety application may need a glass-breaking label while speech content remains inaccessible. Existing mechanisms often assume fixed policies or developer-specified pipelines, such as allowing a stream, denying it, or applying a predetermined degradation. *privMediator* instead treats representation as a runtime synthesis problem: given an application utility contract and a contextual-integrity context, it constructs executable transformations that preserve task utility, evaluates the residual information flows they produce, and selects a permissible output.

Contextual integrity provides the normative foundation for this decision because it evaluates whether an information flow is appropriate for a context based on actors, attributes, purposes, and transmission principles (Barth et al., 2006). Contextual integrity alone, however, does not determine which technical representation should be produced. Granting raw access may reveal more information than an application needs, while denial may prevent useful applications from functioning. We therefore frame the challenge as *dynamic preprocessing-pipeline synthesis and application-visible representation selection constrained by contextual integrity*. Rather than requiring developers or space operators to author one pipeline for every application and context, *privMediator* constructs candidate pipelines from typed preprocessing operators, evaluates the residual flows they induce, and selects the least-revealing feasible representation according to a transparent residual-risk score.

Why this is not just policy compliance. The contribution is not the particular rule library used in the evaluation. A static policy engine could reject flows that match authored prohibitions, but it would not determine which concrete representation—raw stream, redacted media, pose, count, event record, or no release—should be produced for a utility-bearing application. *privMediator* connects a deployment's authored rules to typed preprocessing operators, residual-disclosure effects, utility contracts, and least-revealing selection. Another deployment can replace the rule library, residual-risk weights, or operator catalog without changing the synthesis procedure.

To make runtime synthesis possible, *privMediator* requires machine-readable descriptions of preprocessing behavior. We introduce a typed operator-effect model in which operators declare the representations they consume and produce, the utility capabilities they preserve, the contextual-integrity annotations they contribute, and the residual disclosure effects they induce. These contracts allow the mediator to compose operators, reason about the information flow produced by each candidate pipeline, and compare feasible outputs. For ambiguous contextual judgments, the system can optionally use

LLM-based norm approximations as a fallback, but these soft judgments are bounded by a deterministic decision structure and cannot override hard constraints.

This chapter makes three contributions. First, it introduces a typed operator-effect model for compositional reasoning about privacy-motivated preprocessing, including output representations, utility capabilities, contextual-integrity annotations, and residual disclosure effects. Second, it presents a runtime synthesis architecture that generates candidate pipelines, checks utility compatibility, evaluates residual information flows, and selects the least-revealing feasible output or rejects the request when none exists. Third, it evaluates *privMediator* across 33 contextual scenarios, four smart-space tasks, fixed-input and flexible-input application contracts, a Prolific (pro, 2026) appropriateness survey with 82 participants and 1,476 ratings, and ablations of contextual-integrity filtering and least-revealing selection. The results show that contextual integrity can be operationalized as executable representation selection while making utility, disclosure, and refusal tradeoffs auditable.

3.2 Background

Frictional shared smart environments. Figure 3.1 summarizes the setting we study: sensor-instrumented physical spaces where infrastructure sensors are controlled by a space operator, but the people captured by those sensors may be occupants, visitors, workers, customers, patients, guests, bystanders, or other transient subjects. We call these settings *frictional shared smart environments* because privacy negotiation is conceptually desirable but operationally intractable as a default mechanism. Four properties create this friction: *operator–subject separation*, where the sensing operator differs from the people being sensed; limited *bystander reachability*, where affected individuals cannot reliably be contacted before or during sensing; weak *preference channels*, where notices, negotiation tools, interfaces, or enforceable opt-outs may be absent or impractical; and *dynamic context*, where appropriateness changes with purpose, time, occupants, activity, or events.

These properties motivate mediation mechanisms that decide not only whether a sensor may be used, but what representation of the sensed environment should be released.

Pipelines, utility, and residual disclosure. We frame privacy management in these environments as a problem of selecting a preprocessing pipeline. Rather than granting or denying access to a raw sensor stream, the mediator decides what transformed representation the downstream application should receive. A camera stream might be transformed into a person count, cropped event frame, blurred video, object detections, pose keypoints, or activity label; an audio stream might become a decibel level, sound-event label, or command intent rather than intelligible speech.

We define *utility* as purpose-bound task utility: the extent to which a transformed stream preserves the downstream application's ability to perform its stated task. This follows the task-performance view common in privacy-preserving data mining, where transformed data is evaluated by whether it still supports useful analysis or accurate models (Agrawal and Srikant, 2000; Brickell and Shmatikov, 2008; Li and Li, 2009). In our setting, utility is specified by the application request: HVAC may require only room-level occupancy, fall detection may require pose or motion features, and entryway security may require an event notification or minimized verification image.

We use *privacy risk* as an umbrella term for exposure that remains after preprocessing. When precision is needed, we distinguish three related notions. A *residual disclosure vector* is the set of sensitive attributes that remain inferable from a transformed output, such as identity, face, speech content, activity, trajectory, gait, or co-presence. A *residual risk score* is a context-specific scalar metric computed over that vector to rank and compare feasible pipelines by how revealing they are in the current setting. A *residual CI flow* is the contextual-integrity flow induced by a candidate pipeline: the CI actors and transmission principle, together with the residual information types exposed by the transformed output. Thus, the relevant privacy question is not only which sensor was used, but what information remains after preprocessing, who receives it, for what purpose, and under

Table 3.1: Terminology used to connect application requirements, operator contracts, residual disclosure, and contextual integrity.

Term	Meaning	Example
Application output type	Technical representation requested or accepted by an application	`person_count`, `pose_keypoints`, `sound_event_label`
Operator utility capability	Task capability preserved or produced by an operator	occupancy estimation, fall detection support, sound-event recognition
Residual disclosure attribute	Sensitive attribute still exposed after preprocessing	face, identity, speech content, activity, trajectory, gait
CI information type	Contextual-integrity description of what information flows to a recipient	occupancy information, coarse presence, activity information, speech information

what conditions.

This distinction matters because common transformations reduce some disclosures while preserving others. Face-blurred video may remove facial identity while still revealing clothing, activity, and trajectory; pose keypoints remove pixels and background while potentially revealing gait, disability, exercise, or fall-related behavior; and occupancy counts remove direct identity while potentially revealing routines when collected continuously.

Fixed versus flexible application settings. Many deployed sensing applications have rigid, fixed-input dependencies: a model designed around raw video or audio may not immediately accept a minimized text label, pose sequence, or event record. We call this the *fixed-input setting*. At the same time, some emerging application designs can expose flexible utility contracts, or can be wrapped with adapters that consume semantic representations such as detections, counts, keypoints, sound-event labels, or task events. We call this the *flexible-input setting*. This distinction is an evaluation assumption rather than a claim that all future applications will natively consume arbitrary representations.

We evaluate *privMediator* under both settings. The fixed-input setting reflects deployed applications with brittle raw-like interfaces and tests whether mediation can reduce disclosure without changing the downstream application boundary. The flexible-input

setting is forward-looking: it tests the additional benefit available when an application exposes a purpose-bound utility contract or adapter boundary that can accept multiple representations. We do not assume that all current smart-space applications have this property; rather, flexible-input evaluation shows the design target that becomes available when applications are built to negotiate minimized outputs.

Contextual integrity. Utility and residual-risk metrics alone do not determine whether a data flow is acceptable. The same residual disclosure may be appropriate in one context and inappropriate in another: room-level occupancy may be acceptable for office energy management but inappropriate in a bathroom, prayer room, or medical setting. We therefore use contextual integrity (CI) as the normative framework for judging contextual appropriateness. CI models a personal information flow using parameters such as the *data subject*, *sender*, *recipient*, *information type*, and *transmission principle* (Nissenbaum, 2020):

$$\begin{aligned}&(\text{data subject}, \text{sender}, \text{recipient},\\ &\quad \text{information type}, \text{transmission principle}).\end{aligned}$$

To operationalize this framework, our implementation explicitly parametrizes the overarching social context and application purpose alongside these tuples to determine which specific contextual norms apply.

Our CI rules are not treated as a universal normative oracle. Instead, they are a configurable rule library for a particular deployment or experimental setting. Some rules encode hard legal, institutional, or policy constraints; others encode hand-authored contextual norms used to instantiate the scenarios in our evaluation. The configured-rule audit in Section 3.4 should therefore be read as a policy-compliance and selectivity metric, not as independent ground truth about all possible stakeholder preferences. For ambiguous cases, the system can use bounded norm judgments, and our user study evaluates whether resulting flows align with participant judgments. The key design point is that CI constrains the

information flow induced by a technical preprocessing choice: utility determines whether a pipeline can satisfy the application request, the residual disclosure vector describes what remains exposed, the residual risk score ranks feasible outputs by exposure, and the residual CI flow determines whether the resulting disclosure is appropriate for the current context.

3.3 Method

We design a privacy mediator for smart environments that sits between sensing infrastructure and downstream applications. Rather than granting or denying raw sensor access, the mediator treats each *application request* as a preprocessing-pipeline selection problem. An application request contains a utility contract, which specifies what the application needs to consume, and a CI context, which specifies the contextual-integrity fields used to evaluate the resulting information flow. Given this request, a typed operator-effect catalog, and a CI rule library, the mediator searches for a preprocessing pipeline whose output is useful for the requested task while inducing a residual information flow accepted by the configured rules. If no generated pipeline satisfies these constraints, the mediator returns a *rejected application request*: no transformed representation is released to the application.

The central abstraction is a *typed operator-effect model*. Each preprocessing operator declares the representations it can consume and produce, the task capabilities it supports, the CI terms it contributes, and the residual information that may remain inferable after the transformation. These declarations let the mediator enumerate type-compatible operator chains, check whether outputs match the application contract, propagate residual disclosure estimates, evaluate the induced CI flow, and select a feasible low-disclosure output under the active rule library.

This design avoids two extremes. Coarse access control forces applications into raw allow/deny decisions even when transformed representations such as counts, event labels,

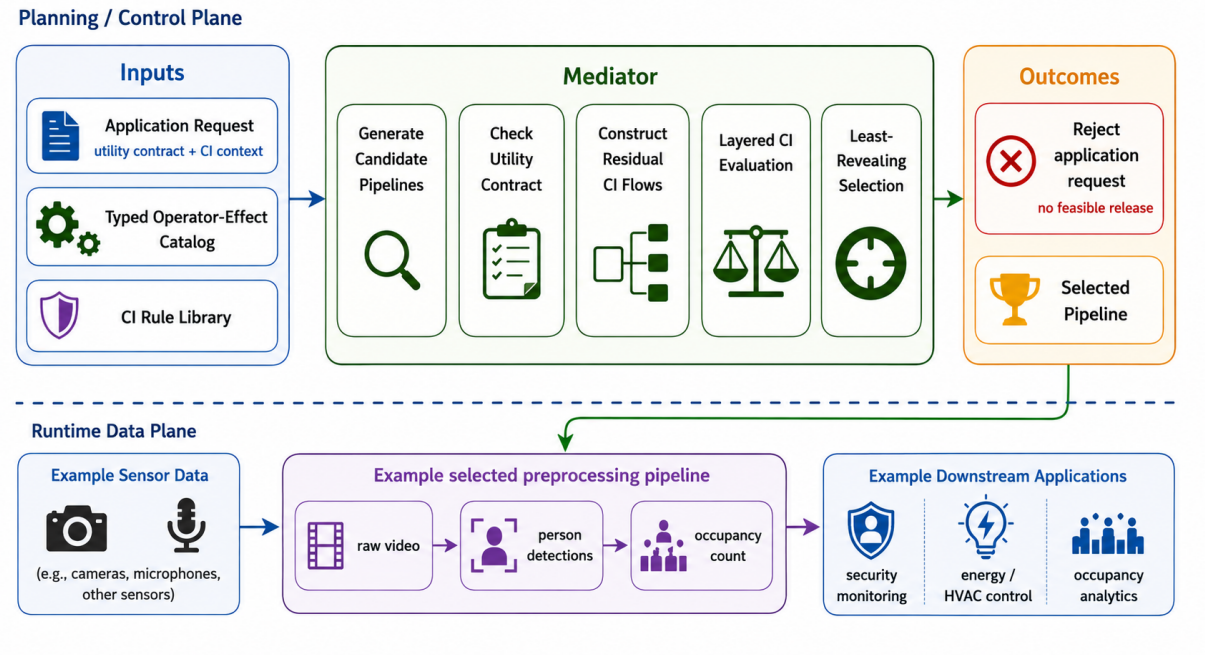


Figure 3.2: *privMediator* uses structured requests, operator-effect contracts, and CI rules to select a low-disclosure preprocessing pipeline or reject the application request.

keypoints, redacted media, or task decisions would suffice. Unconstrained LLM policy-making lets a model directly decide whether a data flow should be allowed. *privMediator* instead makes the residual information flow explicit: operator contracts define the candidate space, CI rules evaluate the induced flow in context, optional LLM judgments are bounded by hard rules, and final selection is performed by a transparent residual-disclosure minimization procedure.

3.3.1 System Overview

Figure 3.2 shows the mediator architecture. The implemented mediator operates in a planning/control plane and produces a decision that is applied in the runtime data plane. The planning inputs are an application request, a typed operator-effect catalog, and a CI rule library. Natural-language request normalization and context extraction can be used as an upstream front end, but are outside the evaluated mediator; all experiments assume that requests have already been converted into structured utility and CI fields.

The mediator produces one of two evaluated outcomes: a selected preprocessing pipeline or a rejected application request. A request is rejected when no generated candidate satisfies the active utility, CI, and residual-disclosure constraints. When a selected pipeline exists, it is deployed in the runtime data plane to transform sensor data before release to the downstream application. Candidate generation, utility checking, residual propagation, and final selection are symbolic. Optional LLM use is restricted to bounded norm judgment and cannot enumerate pipelines, override hard rules, or perform final selection. The symbolic planner and rule evaluator retain authority over all release decisions.

Trust and authority boundary. *privMediator* assumes that the mediation layer, operator-effect catalog, CI rule library, and runtime enforcement boundary are trusted and isolated from downstream applications. The evaluated system decides what representation to release once a structured request has reached this trusted boundary; it does not solve malicious sensors, compromised hardware, applications that lie about their purpose, or side channels outside the mediator. This assumption lets the chapter focus on the decision procedure for representation release, while Section 3.6 discusses complementary deployment controls.

3.3.2 Application Requests and Operator-Effect Contracts

Applications are represented by structured requests rather than raw-stream entitlements. An application request A contains a utility contract U_A and a CI context C_A. The utility contract specifies the requested task capability, acceptable sensing sources, accepted output representations, request-level CI requirements, residual-disclosure bounds, and numerical quality requirements. The CI context provides the sender, subject, recipient, purpose, transmission principles, physical space, and social context used during rule evaluation. For example, a visitor-presence application may accept a detection record, occupancy count, redacted video-like input for a fixed downstream model, or final event label, while also requiring minimization or forbidding identity-bearing outputs. The evaluated request fam-

ilies cover visitor presence, fall detection, activities of daily living, and domestic sound monitoring under both fixed-input and flexible-output application interfaces.

We model the operator catalog as a typed effect system over representations. Let $\mathcal{T}$ be the set of technical representations, including media representations such as raw video, redacted video, filtered audio, and semantic representations such as detections, pose keypoints, sound-event labels, occupancy records, and task events. Each operator variant $o \in O$ declares an input type predicate $\tau_{\text{in}}(o)$, an output representation $\tau_{\text{out}}(o)$, utility capabilities $\kappa(o)$, CI annotations $\gamma(o)$, transformation effects $\eta(o)$, and residual-disclosure effects $\delta(o)$. A pipeline candidate $c = (o_1, \ldots, o_k)$ is type compatible when each operator consumes the representation produced by the previous operator, subject to the catalog's cap-matching and schema-adapter rules. Table 3.2 shows one representative contract; the released artifact contains the complete operator catalog used in the evaluation.

A candidate is symbolically utility-compatible when its final output representation matches one of the accepted output caps in U_A and its accumulated utility capabilities support the requested task capability. Numerical quality requirements, such as latency, recall, precision, or count error, are not proved by symbolic planning; they are evaluated separately in the downstream utility experiments.

The residual disclosure state is a vector over sensitive attributes. Let $\mathcal{D}$ be the evaluated set of tracked residual attributes (e.g., *identity*, *face*, *gait*, *speech content*, *trajectory*, and *aggregate presence*). Each candidate has a residual vector $\mathbf{d}_c \in L^{|\mathcal{D}|}$, where $L = \{\mathsf{none}, \mathsf{low}, \mathsf{medium}, \mathsf{high}, \mathsf{unknown}\}$ is ordered as $\mathsf{none} < \mathsf{low} < \mathsf{medium} < \mathsf{high} < \mathsf{unknown}$. The implementation maps these levels to ordinal values $\rho(\mathsf{none}) = 0$, $\rho(\mathsf{low}) = 1$, $\rho(\mathsf{medium}) = 2$, $\rho(\mathsf{high}) = 3$, and $\rho(\mathsf{unknown}) = 4$.

Operators compose by abstract interpretation over this residual vector. For an operator o, the next residual vector is computed from the current vector $\mathbf{d}$ and the operator's residual effect $\delta(o)$. Media-preserving operators use a *preserve* default: unmentioned attributes remain unchanged. Semantic summarization operators may use a *drop* default, resetting unmentioned raw-media attributes to none before explicit effects are applied.

Table 3.2: Example operator-effect contract. The mediator uses the contract to reason about both utility and residual disclosure before execution.

Contract field	Example value
Input representation	raw video frames or video windows
Output representation	pose-keypoint sequence
Utility capabilities	fall-detection and activity-recognition support
Transformations	removes raw pixels, face texture, and background imagery
Residual disclosure vector	face: none; identity: low/medium; body shape: medium; gait: medium; activity: high; location: medium; trajectory: medium
CI annotations	data minimization, semantic feature release, no raw-video release

Join and fusion operators combine streams conservatively using attribute-wise maximum, and may add correlation risks such as co-presence or trajectory.

The implementation applies explicit effects in a fixed order. `set` assigns a declared risk level; `remove` sets an attribute to `none`; `reduce` lowers a risk level according to the declared reduction rule; and `add` conservatively raises an attribute when fusion or correlation may reveal additional information. Thus, residual propagation is deterministic composition over typed operator effects, not an LLM judgment.

Table 3.2 illustrates one contract; the residual vector records what remains inferable after preprocessing rather than treating a transformation as categorically private. These contracts are authored, versioned artifacts rather than learned privacy guarantees. The implementation therefore treats unknown or unvalidated effects conservatively, and the evaluation interprets residual-disclosure estimates as auditable engineering assumptions rather than formal leakage bounds.

3.3.3 Candidate Generation and Residual CI Flows

Candidate generation is a bounded symbolic search, not an LLM planning step. The operator catalog induces a directed multigraph $G = (V, E)$ in which vertices V are technical representations and edges E are operator variants that transform one representation into another. The planner searches over type-and-effect states rather than representations alone, because two paths may produce the same output representation while inducing different residual disclosures, CI terms, transformations, or utility capabilities.

A planner state is a tuple

$$s = (\tau, \mathbf{d}, K, \Gamma, P),$$

where τ is the current representation, $\mathbf{d}$ is the current residual disclosure vector, K is the accumulated set of utility capabilities, Γ is the accumulated set of CI annotations and transmission principles, and P is the operator sequence. The planner initializes source states from source variants that satisfy A's source requirements, then performs a goal-directed breadth-first search over operator variants whose input caps match the current representation. The search is bounded by a maximum depth and maximum number of expanded states.

A state emits a candidate when its current representation matches an accepted output cap in U_A and its accumulated capabilities satisfy the requested task capability. The planner may apply request-level prefilters, such as accepted-cap transformation constraints or residual-disclosure bounds, but these prefilters are not the final authorization decision; the layered CI evaluator and final selector still evaluate emitted candidates.

For each emitted candidate c, the mediator constructs a residual CI flow

$$f_c = (C_A, \tau_c, \mathbf{d}_c, \Gamma_c, \sigma_c),$$

where C_A supplies the sender, subject, recipient, purpose, context, space, and transmission principle; τ_c is the final output representation; $\mathbf{d}_c$ is the candidate's residual disclosure vector; Γ_c contains accumulated CI annotations; and σ_c records the pipeline stage. Residual attributes are mapped into CI-relevant information types: speech-content residuals become speech-related information types, aggregate-presence residuals become occupancy information, and trajectory residuals become inferred movement information. The staged-flow vocabulary distinguishes raw media, transformed media, semantic features, aggregate records, and final task events so rules can constrain disclosure at each stage.

The implementation preserves staged flows by default. We use four stage labels—raw input, intermediate representation, retained artifact, and output to application—so

hard rules can distinguish transient in-hub processing from artifacts that are retained or released. The no-staged-flows ablation collapses these labels to the output boundary.

3.3.4 Layered CI Evaluation and Probe Validation

The CI evaluator applies rule-library constraints to each residual flow f_c in layers. Classification rules first add tags to the residual flow. Hard denial rules reject prohibited flows. Required-condition, required-transformation, and conditional-allowance rules check whether the flow contains necessary transmission principles, contextual fields, transformations, or forbidden conditions. Preference-only rules may inform diagnostics or ranking, but are not treated as hard accept/reject decisions. The evaluated rule library includes hard denials, required conditions, required transformations, conditional allowances, and preference-only rules with explicit provenance.

Rule authoring and maintenance. The rule library is a deployment artifact. In our prototype it is hand-authored from contextual integrity scenario requirements and public policy or guidance examples, and each rule records provenance metadata: a source name, source type, natural-language policy statement, and, for external sources, a URL. The evaluated libraries draw from short-term-rental platform policies (Airbnb and Vrbo), campus and workplace surveillance policies, and legal, regulatory, or professional guidance for schools, research, and clinical settings; a small number of created templates capture mediator-specific boundaries such as final-task-decision outputs. We use these sources as representative policy constraints for evaluating the mediation architecture, not as a complete legal or platform-compliance implementation.

The rules fall into four operational families: *hard denials*, such as hidden recording devices or cameras in prohibited private spaces; *required conditions*, such as disclosure, explicit consent, authorized-personnel access, or limited retention; *required transformations*, such as face redaction, speech-content removal, or aggregation before release; and *preference-only rules*, which may affect diagnostics or ranking but do not authorize or

deny by themselves. In a deployment, these rules would be authored and revised by the organization or governance body responsible for the space, potentially with input from legal, compliance, resident, worker, patient, or research-ethics stakeholders. Rules need not change for every application: they change when policy, social context, sensor placement, or acceptable transmission principles change, while new technical capabilities are added through operator contracts.

If a candidate fails a hard rule, it is rejected before any LLM judgment is considered. For candidates that pass hard rules, the mediator may invoke an LLM as a bounded norm interpreter over the structured residual flow. In normal operation, the LLM cannot override hard rules, enumerate pipelines, or perform final selection; an LLM-only mode exists only as an ablation. The CI rule library is configurable rather than a universal normative oracle, so deployments can replace or extend the rules with institutional policy, legal constraints, or survey-derived norms without changing the synthesis algorithm.

The mediator also maintains a metadata residual estimate $\mathbf{d}_c^{\text{meta}}$ from operator contracts. When transformed artifacts and a probe configuration are available, empirical privacy probes estimate inferability from the concrete transformed output. These probes are implemented as auxiliary detectors or recognizers over transformed artifacts, such as face detection, person/body detection, OCR, scene-object probing, audio-presence checks, ASR-based speech-content checks, diarization, and aggregate or trajectory inference probes. They do not directly authorize a flow; they only produce a probe residual vector $\mathbf{d}_c^{\text{probe}}$.

Probe evidence is combined conservatively with metadata:

$$d_{c,i} = \max(d_{c,i}^{\text{meta}}, d_{c,i}^{\text{probe}}) \quad \forall i \in \mathcal{D},$$

where the maximum is taken under the residual-risk ordering above. Thus probe evidence can make a candidate appear more revealing, but cannot make it appear safer than its symbolic contract. The probes are implementation checks for concrete artifacts, not formal

privacy proofs against arbitrary observers. When no probe artifacts are supplied, final selection uses $\mathbf{d}_c^{\text{meta}}$ alone; Probe configurations and per-artifact outputs are retained in the released evaluation artifacts.

3.3.5 Least-Revealing Feasible Selection and Rejection

The final selector consumes generated candidates, CI evaluations, optional probe results, and the application request. Let $\mathcal{C}$ be the set of generated candidates. The selector first forms the feasible set:

$$\mathcal{C}_{\text{feas}} = \{c \in \mathcal{C} : U(c, A) = 1 \wedge \text{CI}(f_c, N) = \text{accept} \wedge \mathbf{d}_c \preceq \mathbf{b}_A\},$$

where $\mathbf{b}_A$ is the request's residual-disclosure bound and $\preceq$ is attribute-wise comparison under the residual-risk ordering.

Among feasible candidates, *privMediator* selects the lowest-disclosure feasible candidate according to a context-specific residual-risk score. Let $\mathbf{w}_{C_A}$ be the policy weight vector active for the request context, where $w_{C_A,i}$ encodes how strongly the deployment penalizes residual attribute i in this purpose and context. The residual-risk score is

$$R_{C_A}(c) = \sum_{i \in \mathcal{D}} w_{C_A,i} \cdot \rho(d_{c,i}).$$

The default selector chooses

$$c^* = \arg\min_{c \in \mathcal{C}_{\text{feas}}} R_{C_A}(c).$$

Deterministic tie breakers (including lexicographic attribute ordering $\text{Lex}(\mathbf{d}_c)$, pipeline latency, and application output priority) are applied sequentially if multiple candidates share the minimal risk score. The residual-risk weights are transparent policy parameters, not learned constants or universal privacy claims; in our experiments, they are hand-authored and tested through the ablations in Section 3.4.5.

If $\mathcal{C}_{\text{feas}}$ is empty, *privMediator* returns a *rejected application request*: no transformed

Algorithm 2 CI-constrained preprocessing pipeline mediation

Require: application request A, operator catalog O, CI rules N, optional probe artifacts T

Ensure: selected preprocessing pipeline or rejected application request

Load structured utility contract U_A and CI context C_A from A
Construct source states and operator variants from typed operator-effect contracts in O
Run bounded BFS over type-and-effect states $s = (\tau, \mathbf{d}, K, \Gamma, P)$
Emit candidates whose output caps and accumulated capabilities satisfy U_A
for each candidate c **do**
    Construct residual CI flow $f_c = (C_A, \tau_c, \mathbf{d}_c^{\text{meta}}, \Gamma_c, \sigma_c)$
    Evaluate f_c using CI rules N and optional bounded norm judgment
    if probe artifacts T are available for c **then**
        Run empirical probes to estimate $\mathbf{d}_c^{\text{probe}}$
        Set $\mathbf{d}_c \leftarrow \max(\mathbf{d}_c^{\text{meta}}, \mathbf{d}_c^{\text{probe}})$ attribute-wise
    else
        Set $\mathbf{d}_c \leftarrow \mathbf{d}_c^{\text{meta}}$
    end if
end for
Build feasible set $\mathcal{C}_{\text{feas}} = \{c : U(c, A) = 1 \wedge \text{CI}(f_c, N) = \text{accept} \wedge \mathbf{d}_c \preceq \mathbf{b}_A\}$
if $\mathcal{C}_{\text{feas}} = \emptyset$ **then**
    return rejected application request with diagnostics
end if
Select $c^* = \arg\min_{c \in \mathcal{C}_{\text{feas}}} R_{C_A}(c)$ with deterministic tie breakers
return selected preprocessing pipeline c^*

representation is released. Diagnostics distinguish no utility-compatible representation, hard denials, residual-disclosure failures, and unresolved constraints. Deployments could route some rejected or uncertain requests to consent, human review, or degraded-task negotiation, but our evaluation treats them as rejected application requests.

3.3.6 Algorithm

Algorithm 2 summarizes the implemented procedure.

3.4 Evaluation

We evaluate whether *privMediator* can mediate smart-space sensing requests by preserving downstream utility while enforcing configured CI constraints and reducing residual disclosure under operator contracts. The evaluation has five purposes: decision coverage, downstream utility, human judgments of perceived contextual appropriateness, ablations, and deployment-oriented context-switch latency. Throughout this section, CI-conflict and hard-rule-violation counts are *configured-rule audit metrics*: they measure whether a method releases flows that violate the authored rule library used in the experiment. They should be interpreted as policy-compliance and selectivity metrics, not as independent ground truth about all possible stakeholder preferences.

RQ1: Synthesis and feasibility. Does the mediator select a utility-compatible output when one is contextually acceptable, and reject the application request when no feasible release exists?

RQ2: Downstream utility. Conditional on selecting an output, do generated preprocessing pipelines preserve task performance relative to raw input and manually tuned preprocessing?

RQ3: Perceived contextual appropriateness. Are less revealing mediated information flows judged more appropriate than raw or utility-maximizing outputs, and how do method-level ratings change once rejected contexts are considered?

RQ4: Design choices. Which mediator components affect selection rate, utility retention, and contextual appropriateness?

RQ5: Dynamic context switching. Can the mediator recompute decisions quickly enough for context changes, and does it avoid hard-rule-violating releases during those changes?

Table 3.3: Downstream utility tasks, datasets, interfaces, and primary metrics.

Task	Dataset	Downstream model/input	Primary utility metric
Visitor presence detection	ChokePoint (Wong et al., 2011)	Person detector over image/video frames; predictions aligned to frame-level person annotations	Frame-level person-presence F1
Fall detection	Le2i Fall (Charfi et al., 2013)	Pose-based fall classifier over keypoint sequences/windows	Video-level macro-F1
ADL recognition	YouHome ADL (Pan et al., 2022)	Audio–visual ADL classifier over synchronized image frames and audio waveform segments	Macro-F1 over ADL classes
Domestic sound monitoring	CHiME-Home (Foster et al., 2015)	Multi-label audio tagger over fixed-duration audio chunks	Macro-F1 over sound labels

3.4.1 Experimental Setup

Tasks and datasets. We evaluate utility on four smart-space tasks that match the task categories in our contextual-integrity scenario pool: visitor presence detection, fall detection, activity-of-daily-living (ADL) recognition, and domestic sound monitoring. Table 3.3 gives the dataset, downstream interface, and primary metric for each task; dataset-specific preprocessing and evaluator configurations are retained in the released artifacts.

Context scenarios. We instantiate 33 contextual-integrity scenarios across the four tasks: eight visitor-presence scenarios, eight fall-detection scenarios, eight ADL-recognition scenarios, and nine domestic-sound scenarios. Each scenario fixes the context, space, sender, subject, recipient, purpose, and transmission principle, but leaves the released representation unspecified. Scenario identifiers are not contiguous because pilot and ambiguous child/guardian scenarios were removed after survey-design feedback; for example, the final pool contains 33 scenarios even though one retained identifier is S034.

Application contracts. We evaluate two application-interface assumptions. In the *fixed-input* setting, the downstream application expects the representation used by the existing task evaluator, such as image/video frames, pose-compatible inputs, synchronized audio-video, or audio waveforms. In the *flexible-input* setting, the application declares several acceptable output representations, including semantic outputs such as detections, occupancy counts, pose keypoints, sound-event labels, or event records. The fixed-input

and flexible-input requests share the same task goals but differ in the set of representations accepted at the application boundary; the exact schemas and adapters are retained in the released artifacts.

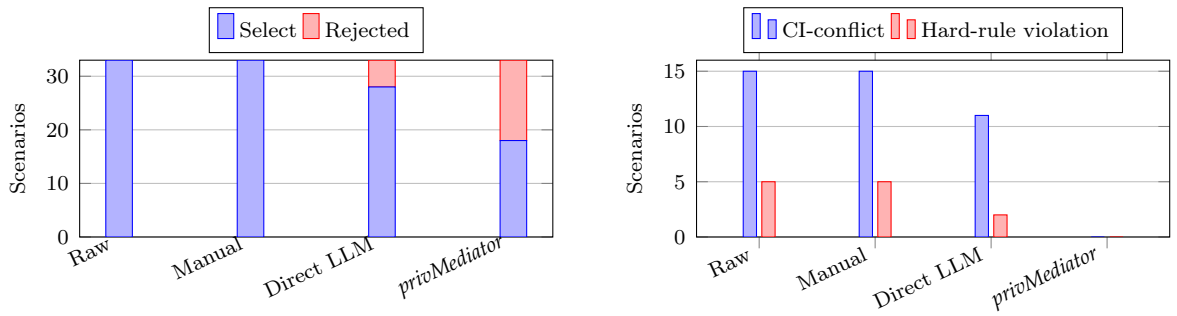
Compared methods. We compare RAW, a developer-written MANUAL preprocessing policy, a DIRECT-LLM baseline, and *privMediator*. RAW forwards the original sensor stream. MANUAL applies a fixed task-specific preprocessing policy without context-specific CI reasoning. DIRECT-LLM selects or refuses a pipeline using a prompt over the application request and context. *privMediator* performs symbolic candidate generation, utility checking, residual-flow construction, CI filtering, and least-revealing feasible selection. We also evaluate ablations that remove CI filtering, optimize utility only, or alter the final selection objective. The fixed and flexible operator catalogs contain 25 and 28 operators, respectively; selected pipelines use 23 distinct operator IDs, all present in the corresponding catalogs.

Hardware. Experiments were run on a workstation with an NVIDIA GeForce RTX 2070 GPU and an AMD Ryzen 7 3800X 8-core processor with 16 hardware threads. `GPT-4o-mini` (Hurst et al., 2024) was used as the LLM for both the direct LLM baseline as well as *privMediator*.

3.4.2 Pipeline Synthesis and Feasibility

This experiment asks whether each method selects an output and whether it releases data in contexts where the configured rule library says no feasible release should exist. We use *Rejected* for cases where the method returns no output because no generated pipeline satisfies the active utility, CI, and residual-disclosure constraints. We use *Review* only for the direct-LLM baseline when it explicitly asks for human review rather than selecting or rejecting.

Figure 3.3a summarizes decisions across the 33 context scenarios. RAW and MANUAL



(a) Decision coverage

(b) Configured-rule violations

Figure 3.3: Coverage and configured-rule audit outcomes across 33 scenarios. Direct-LLM review is counted with rejected decisions because no data is released. CI-conflict and hard-rule-violation releases are fixed-input outputs that violate the configured CI audit rules.

always select an output. DIRECT-LLM selects an output in 28 scenarios and returns no output in five cases: four rejected requests and one review decision. *privMediator* selects an output in 18 scenarios and rejects 15 requests. These rejections are mediation decisions, not execution failures: they indicate that, under the configured rule library and residual bounds, the mediator found no utility-compatible representation that also satisfied the active CI and residual-disclosure constraints. RAW releases raw video, audio, and AV streams; MANUAL, DIRECT-LLM, and *privMediator* primarily release redacted images, pose, filtered audio, or redacted/filter AV outputs when they select a pipeline.

Residual-disclosure accounting. We also audit residual disclosure and rejection causes. Contract-derived accounting shows that transformed outputs reduce raw-like exposure, face exposure, and speech-content exposure relative to RAW. Table 3.4 gives the compact main-body summary: in the fixed-input run, *privMediator* releases no raw-like outputs, no selected outputs with medium-or-higher face exposure, and no selected outputs with medium-or-higher speech-content exposure. These values are metadata audits over typed operator effects, not formal leakage guarantees; empirical probes are used as conservative implementation checks that can raise, but not lower, residual-risk estimates. The released evaluation records contain the full residual-disclosure vectors and probe outputs.

Table 3.4: Contract-derived residual disclosure for selected outputs. Values are descriptive metadata audits, not formal leakage guarantees.

Run	Method	Sel.	Raw-like	Mean risk	Face ≥ med. / Speech ≥ med.
Fixed	RAW	33	25	27.5	72.7% / 51.5%
Fixed	MANUAL	33	0	18.0	0.0% / 15.2%
Fixed	*privMediator*	18	0	17.4	0.0% / 0.0%
Flexible	RAW	33	25	26.3	72.7% / 51.5%
Flexible	MANUAL	33	0	16.3	0.0% / 15.2%
Flexible	*privMediator*	21	7	14.4	19.0% / 14.3%

The refusals also reflect substantive contextual constraints rather than missing implementations. None of the 15 *privMediator* rejections was dominated by lack of utility support: each had at least one utility-compatible representation, but all feasible representations failed a hard denial, required condition, or residual-disclosure constraint. Four refusals involved hidden or undisclosed monitoring, four involved private or sensitive spaces, and the remaining seven reflected routine surveillance, research/bystander limits, outdoor-location conditions, or indoor-audio constraints. This matters because a rejection is not equivalent to a failed pipeline run; it is the mechanism by which the mediator surfaces a conflict between application utility and configured contextual constraints. The released evaluation records retain the per-scenario rejection diagnostics and generated candidate traces.

3.4.3 Downstream Utility

We report utility conditional on a method selecting an output, and report selection/rejection counts separately. Utility retention is the primary task metric divided by the corresponding RAW metric; it is not coverage-adjusted, so it measures whether released outputs remain task-useful while leaving the availability cost visible. Fixed-input outputs are evaluated with the existing task evaluators, while flexible semantic outputs use the adapter boundary declared in the application contract.

Across four tasks, *privMediator* retains 96.5% fixed-input selected-output utility and 91.2% flexible-input selected-output utility. Table 3.5 shows that the utility tradeoff is not uniform across tasks. In the fixed-input setting, selected outputs retain manual-

Table 3.5: *privMediator* selected-output utility by task. Utility is conditional on release and normalized to RAW.

Task	Fixed sel.	Fixed ret.	Flex sel.	Flex ret.
Visitor	4/8	0.973	7/8	0.858
Fall	6/8	1.000	6/8	0.804
ADL	4/8	1.021	4/8	1.021
Sound	4/9	0.866	4/9	0.966

Table 3.6: Coverage-aware summary for principal methods. Utility is selected-output retention averaged over task-level primary metrics; configured-rule conflicts are fixed-input audit counts.

Method	Fixed sel.	Flex sel.	Fixed util.	Flex util.	CI-conflict rel.	Hard-rule viol.	Survey mean
RAW	33/33	33/33	1.000	1.000	15	5	2.83
MANUAL	33/33	33/33	0.965	0.927	15	5	3.43
DIRECT-LLM	28/33	33/33	0.965	0.822	11	2	3.56
privMediator	18/33	21/33	0.965	0.912	0	0	3.72

policy utility whenever a CI-feasible representation exists. In the flexible-input setting, Visitor and Fall lose selected-output utility because the mediator chooses more minimized outputs such as counts, detections, or pose keypoints; ADL remains stable, and Sound retains 96.6% of raw utility when selected. Table 3.5 presents the per-task selected-output results used in this summary.

Table 3.6 gives the combined coverage-aware view. It shows the tradeoff that selected-output utility alone obscures: always-releasing methods retain high availability but release in configured CI-conflict contexts, while *privMediator* reduces availability by refusing no-feasible-output scenarios and retains high utility on selected outputs.

Figure 3.3b summarizes the coverage-aware interpretation for the fixed-input run across the full context pool. Selected-output utility alone can be misleading: methods that always release data can obtain high utility by releasing in contexts where release should not occur. RAW and MANUAL each release in 15 CI-conflict contexts, including 5 hard-rule-violating releases. DIRECT-LLM reduces but does not eliminate these releases, selecting outputs in 11 CI-conflict contexts, including 2 hard-rule-violating releases. *privMediator* has lower selection rate, but eliminates CI-conflict and hard-rule-violating releases.

Selected-output utility shows that *privMediator*'sreleased pipelines remain task-useful;

selection rate and configured-conflict counts show the complementary cost and benefit of refusing outputs in contexts where the rule library finds no CI-feasible release.

3.4.4 Perceived Contextual Appropriateness Survey

We evaluate perceived contextual appropriateness with a Prolific (pro, 2026) survey in which participants rated concrete scenario-output information flows from the perspective of the person the data is about. Items combined a CI context with a shared output; method labels were hidden. The survey measures perceived appropriateness, not objective leakage, legal compliance, or consent validity. The completed-session analysis includes 82 participants and 1,476 analyzed ratings over deduplicated scenario-output pairs. The survey instrument, filtering criteria, randomization procedure, and anonymized response summaries are retained in the released artifacts.

privMediator receives the highest descriptive method-linked mean rating among the principal methods (3.72, with 66.0% of ratings at least 4), but differences among transformed outputs are smaller than the difference between raw and transformed outputs. A mixed-effects model finds that raw outputs are rated lower than the transformed-output reference condition (coefficient -0.64, $p < 0.001$), while differences among *privMediator*, DIRECT-LLM, and MANUAL selected outputs are comparatively small. We therefore interpret the survey as supporting the contextual plausibility of transformed releases, while the configured-rule audit captures *privMediator*'sdistinct policy-level selectivity: DIRECT-LLM still releases in 11 fixed-input CI-conflict contexts, including 2 hard-rule-violating releases, whereas *privMediator* eliminates those releases.

3.4.5 Ablations and Dynamic Context Switching

The ablations isolate two effects. First, CI filtering controls refusal: UTILITY-ONLY and NO CI FILTER select outputs in every context and retain high selected-output utility, but do so by releasing data in contexts where the full mediator rejects the request. Second,

Table 3.7: Key ablations. CI filtering controls refusal; least-revealing selection controls which feasible output is released.

Method	Fixed sel.	Fixed util.	Survey mean	Approp.
privMediator	18/33	0.965	3.72	66.0%
UTILITY-ONLY	33/33	1.000	2.97	44.2%
NO CI FILTER	33/33	0.965	3.45	59.5%
NO LEAST-REVEALING	18/33	0.989	3.31	52.5%
LATENCY-FIRST	18/33	0.989	3.31	52.5%

least-revealing selection controls what is released after feasibility filtering: NO LEAST-REVEALING and LATENCY-FIRST keep the same refusal decisions as *privMediator* but choose more revealing feasible outputs, increasing utility retention while lowering appropriateness ratings. Table 3.7 gives the comparison needed to isolate these two effects.

To check whether mediation is practical under changing contexts, we replayed 17 context phases across four dynamic traces. *privMediator* recomputed decisions when the CI tuple changed rather than per frame, selected outputs in 12 phases, rejected or reviewed 5, and made zero hard-rule-violating releases. RAW and MANUAL released in all phases but produced hard-rule-violating releases in 4/17 phases; DIRECT-LLM produced hard-rule-violating releases in 2/17. Median *privMediator* context-switch latency was 2.28s, dominated by candidate generation rather than CI rule checking. Per-phase decisions and latency traces are retained in the released evaluation records.

3.5 Related Work

Privacy tensions and contextual norms in smart environments. Prior work shows that smart-environment privacy is a social and contextual problem involving device owners, household members, guests, workers, and bystanders. Studies of smart-home users and incidental users identify concerns about sensing infrastructure owned or configured by others (Zeng et al., 2017; Cobb et al., 2021; Alshehri et al., 2022), while work on guests and bystanders shows that expectations depend on relationships, familiarity with the space, and the social cost of raising privacy concerns (Marky et al., 2022; Thakkar et al., 2022). Contextual integrity has been used to measure smart-home privacy norms

and interpersonal data-sharing expectations (Apthorpe et al., 2018; He et al., 2024), and to encode legal rules or privacy preferences for automated reasoning over information flows (Barth et al., 2006; Apthorpe et al., 2018; Abdi et al., 2021). This work motivates our use of CI, but it primarily tells us which information flows may be appropriate. Our contribution is complementary: we operationalize CI as a runtime synthesis problem, deciding which executable preprocessing pipeline can produce a utility-preserving residual flow that satisfies those contextual constraints.

Privacy awareness, notice, and preference learning. Another line of work seeks to make privacy practices more visible or easier to act on. Privacy labels communicate IoT or app data practices to users, but prior work shows that labels can be difficult to design, incomplete, or dependent on presentation format (Emami-Naeini et al., 2020; Li et al., 2022; Caven et al., 2024). Privacy assistants and notice-and-choice tools provide more interactive support by learning preferences, delivering notices, or helping users make privacy decisions in mobile and IoT settings (Langheinrich, 2002; Das et al., 2018; Liu et al., 2016; Colnago et al., 2020; Feng et al., 2021). These systems improve awareness and preference expression, but they generally assume that privacy decisions can be made by or for reachable individuals. Frictional shared smart environments weaken this assumption: affected people may be transient, unknown, socially unable to object, or sensed together with others. Our system therefore focuses on reducing unnecessary disclosure through mediation when direct notice or negotiation is unavailable or insufficient.

Managing privacy and control in shared environments. Several systems address multi-user control in smart environments. Prior work studies how households coordinate and contest control over smart-home devices and proposes mechanisms for access control or conflict resolution among multiple users and devices (Geeng and Roesner, 2019; Sikder et al., 2022). Other systems explore collective or negotiated control: Hivemind supports democratic control of public actuators (Kim et al., 2021), TEO provides ephemeral ownership so stakeholders can control data generated while they are present (Zhang et al.,

2022a), and ThingPoll supports negotiation of privacy settings for shared sensing devices (Zhou et al., 2024). These systems recognize that privacy decisions in shared environments cannot be made solely by device owners. Our work targets a different point in the design space: cases where affected stakeholders cannot reliably be reached or coordinated. Rather than depending on negotiation as the primary mechanism, the mediator uses contextual constraints and operator contracts to select a least-revealing feasible representation, or reject the application request when no acceptable release exists.

Privacy-preserving sensing and data-processing middleware. A complementary line of systems protects privacy by mediating how applications access, transform, and transmit sensor data. DARKLY interposes on computer-vision inputs and applies privacy-preserving transformations before perceptual applications receive visual data (Jana et al., 2013), while ipShield enforces context-aware privacy policies by mediating access to sensed information (Chakraborty et al., 2014). FlowFence, SmartAuth, and ContexIoT constrain IoT application behavior through declared data flows, semantic authorization, or context-aware prompts (Fernandes et al., 2016; Tian et al., 2017; Jia et al., 2017). Other systems use hubs, gateways, or middleware to filter, transform, or minimize data before it leaves the local environment, including Privacy Mediators, TIPPERS, PFirewall, Peekaboo, and PrivGuard (Davies et al., 2016; Mehrotra et al., 2016; Chi et al., 2019; Jin et al., 2022; Wang et al., 2022). These systems show that privacy can be enforced in the data path rather than only through notice or policy text. However, they typically enforce declared flows, apply fixed policies or fixed transformations, mediate known data types, or minimize outgoing data under a predetermined policy. *privMediator* instead synthesizes preprocessing pipelines at runtime from typed operator-effect contracts, checks whether candidate outputs satisfy an application utility contract, evaluates the residual CI flow, and selects the least-revealing feasible representation.

LLM-assisted privacy reasoning. Recent work uses large language models to parse privacy policies, reason about legal requirements, and assist with privacy-preserving data

processing (Khandelwal et al., 2021; Rahat et al., 2022; Rodriguez et al., 2024; Li et al., 2025b; Chen et al., 2025). Other work connects LLMs to contextual integrity or privacy-preserving function selection, using models to summarize information, judge whether disclosures are appropriate, or select privacy-preserving transformations from natural-language requirements (Li et al., 2025a; Khan et al., 2026; Wang et al., 2024). These works suggest that LLMs can help bridge informal privacy expectations and executable mechanisms. Our system uses LLMs only in bounded roles, such as optional upstream request normalization or structured norm judgment. LLMs do not enumerate pipelines, override hard CI rules, or perform final authorization. The enforceable decision structure remains symbolic: typed contracts define the candidate space, CI rules constrain the residual information flow, and the final selector chooses among feasible candidates or rejects the request.

3.6 Discussion

Trusted mediation boundary. *privMediator* assumes a trusted mediation layer between infrastructure sensors and downstream applications. The mediator, operator-effect catalog, CI rule library, and runtime enforcement boundary must be isolated and trusted. While the system reduces unnecessary exposure by transforming data before release, it does not prevent all adversarial behavior; a malicious application might combine permitted outputs with out-of-band side channels, misstate its purpose before the request reaches the mediator, or infer sensitive facts from repeated permitted outputs. A compromised sensor could also bypass the planning plane entirely. Practical deployments therefore require complementary infrastructure security, including sensor attestation, application authentication, audit logging, access revocation, and sandboxed execution. Our contribution is not a complete trusted-computing architecture, but rather the auditable decision procedure for what data representation should be released when such a mediation boundary is enforced.

Beyond policy compliance. A configured-rule audit by itself would only show that a method follows a policy. *privMediator* is intended to address the harder systems question of how an authored policy becomes an executable, utility-aware release decision. The same rule library can admit multiple technically valid outputs with different utility and disclosure profiles; conversely, a rule violation may remain even after strong redaction if the context forbids the underlying sensing or recipient relationship. The contribution is therefore the coupling of rule evaluation with typed operator synthesis, residual-disclosure propagation, and least-revealing feasible selection, not the claim that our experimental rule library is the only correct set of norms.

Normative scope of configured CI rules. The CI rule library, residual-risk weights, and operator contracts are authored artifacts rather than universal definitions of privacy. In practice, these artifacts can be derived from institutional regulations, workplace policies, building rules, community standards, or survey-derived norms. This separation intentionally decouples the synthesis procedure from the source of normative authority. However, it also means that configured-rule violations in our evaluation are best understood as policy-compliance and selectivity metrics: they show whether a method releases flows that violate the authored rules used for the experiment, not whether all stakeholders would necessarily agree with those rules. We therefore pair the configured-rule audit with selected-output utility, residual-disclosure accounting, probe sanity checks, ablations, and participant appropriateness ratings. The authoring burden is primarily a deployment/governance task rather than an application-developer task: rules should be revised when policies, spaces, sensor placements, or transmission principles change, while ordinary applications interact with the mediator through utility contracts. In our evaluation, rejection diagnostics were explained by recurring rule families—hidden recording, private-space bans, required notice/consent, limited access/retention, and outdoor/security-purpose conditions—rather than by per-application handcrafted decisions.

Specification and contract verification. System efficacy depends heavily on the accuracy of operator contracts and residual-disclosure estimates. Misconfigured operator profiles could understate leakage, while overly conservative profiles could cause unnecessary application rejections. Operator contracts should therefore be versioned, testable, and auditable. The residual-disclosure accounting in Section 3.4 is useful as a sanity check because it summarizes the exposure implied by these contracts across selected outputs, but it should not be read as a formal privacy or leakage guarantee. Our empirical privacy probes mitigate this risk by detecting unmodeled leakage in concrete transformed artifacts, but they function as implementation checks rather than formal privacy proofs; probe evidence can conservatively raise a candidate's risk profile, but cannot guarantee bounds against an unmodeled, stronger adversarial observer. A stronger deployment would include independent red-team testing, residual-disclosure regression tests, and review procedures for adding new operators or changing risk weights.

Operational costs of refusal. By making infeasibility explicit, *privMediator* avoids silent releases of data that violate the configured policy. Strict refusal also introduces operational costs in safety-critical settings such as fall detection, clinical monitoring, or security response. Real-world deployments must decide how to handle these explicit rejections, for example by routing them to human-in-the-loop review, emergency escalation, conditional consent, or degraded fallback tasks. *privMediator*'srole is to surface these conflicts and emit auditable diagnostics; exception handling remains a deployment-specific governance decision.

Flexible application contracts. The flexible-input evaluation demonstrates the potential benefit of applications that can accept multiple minimized representations, but it should not be read as a claim that all smart-space applications are already flexible. Some applications will remain bound to raw-like interfaces or will require new adapters before they can consume semantic outputs such as counts, keypoints, or event records. The flexible setting is therefore a design target: it shows what privacy mediation can gain when

downstream applications expose purpose-bound utility contracts instead of hard-coded raw-stream dependencies.

Limitations and generalizability. Our user study evaluates perceived contextual appropriateness rather than legal compliance, consent validity, or quantitative information leakage. Our utility evaluation is bounded by four smart-space tasks, 33 scenarios, and a finite operator catalog. Extending *privMediator* to other domains requires expanding the operator schemas, adapters, and CI rule libraries, and validating the resulting residual-disclosure estimates. Nevertheless, the central architecture—synthesizing typed preprocessing pipelines, evaluating residual CI flows, and selecting the least-revealing feasible representation—is not specific to the evaluated datasets and can be instantiated in other sensor-saturated shared environments.

3.7 Conclusion

This chapter presented *privMediator*, a privacy mediator that constructs application-visible sensor outputs in privacy-sensitive shared smart environments. Rather than reducing privacy to a binary decision between raw-data access and denial, *privMediator* generates candidate processing pipelines, checks whether their outputs satisfy an application's utility contract, constructs the residual information flows they induce, and selects the least-revealing representation permitted by the configured contextual-integrity rules. When no candidate satisfies the active utility, rule, and residual-disclosure constraints, the system explicitly rejects the application request.

Across 33 scenarios and four smart-space tasks, *privMediator* retains high selected-output utility while making no releases that violate the configured hard rules. The user study further indicates that transformed outputs are generally perceived as more appropriate than raw streams. The configured-rule audit captures a complementary benefit that perception ratings alone do not, because the mediator refuses release when the cur-

rent context admits no feasible representation. The contribution is therefore an auditable architecture for making utility, disclosure, and refusal tradeoffs explicit, not a claim of universal privacy or formal leakage elimination.

Within the dissertation, *privMediator* addresses the representation pillar by determining which technical form of event-relevant information may be provided to a downstream application. It establishes raw observations, transformed observations, semantic features, and event-level outputs as governed alternatives rather than predetermined interfaces. The chapter still assumes, however, that the sensing and inference operations needed to produce a selected representation are available for execution. The next chapter turns to that systems problem and asks how complex-event progress can determine which sensing and inference computations should run, where they should execute, and what state must accompany them.

CHAPTER 4

Adapting Distributed Sensing and Computation as Complex Events Unfold

Chapter Overview

The preceding chapters considered how a sensing system can construct event explanations and govern the representations exposed to applications. This chapter examines a third question: can the progress of a complex event help determine which sensing and inference computation should run next?

We explore this question through *FABLE*, a Python prototype and experimental testbed for progress-driven orchestration. At the time of the reported evaluation, each complex-event workflow was authored separately in Python. The author specified its predicates, participant roles, ordering, temporal conditions, and, in several cases, scenario-specific execution hints. The prototype then tracked partial occurrences of that workflow, identified which authored predicate was currently enabled, and activated a corresponding predefined provider pipeline. When more than one source or placement was feasible, a bounded ranking procedure selected among the small number of available

realizations.

This scope is deliberately narrower than a general-purpose complex-event runtime. The prototype did not provide an external language through which an arbitrary user could define a new event, nor did it synthesize novel perception pipelines. Instead, it allowed us to study whether event progress could serve as a useful orchestration signal in nine curated workflows executed through a controlled MQTT- and replay-based testbed. In the dissertation's framing, the chapter therefore asks: *within a developer-authored complex-event workflow, how can current progress determine which configured computation is useful next?*

4.1 Introduction

Complex events unfold through several related observations rather than one detector invocation. A convoy-like event may involve one vehicle crossing a view, another vehicle following within a bounded interval, and the scene later clearing. A robbery workflow may combine an audio trigger with evidence about a vehicle observed before or after that trigger. Recognizing such events requires maintaining partial progress, binding observations to participant roles, and evaluating temporal or spatial relationships across sensing modalities.

Complex event processing provides abstractions for temporal and relational patterns, but commonly assumes that the primitive events already exist as streams. In a camera- and microphone-based deployment, those primitives must first be produced by perception components. A predicate such as FOLLOWS may depend on detection, tracking, and geometric reasoning, while an audio predicate may require a classifier over a microphone stream. Running every component continuously can waste computation and bandwidth. A fixed pipeline, however, cannot take advantage of the fact that different evidence becomes useful at different stages of an event.

This chapter explores a simple hypothesis: *the current progress of a complex event can*

be used to decide which configured sensing and inference pipeline should be active. For example, a workflow need not evaluate a follower relationship until a possible leader has been observed. Similarly, a post-trigger vehicle search need not run before the corresponding alarm or gunshot stage becomes active. Event progress can therefore act as a control signal for starting, retaining, or stopping work.

We investigate this hypothesis through *FABLE*(**F**rontier-**A**ware **B**inding and **L**ifecycle **E**xecution), a Python research prototype coupled to an MQTT- and replay-based experimental testbed. The prototype represents the progress of a possible event occurrence as a partial instance. Its *active frontier* contains the predicates in the authored workflow that can currently advance that instance. When a predicate result is accepted, the prototype updates the instance, derives the next frontier, and activates the provider work associated with that frontier.

The scope of this contribution is important. At the time of the reported evaluation, *FABLE* did not expose a general user-facing complex-event language. Each evaluated workflow was implemented separately as Python code. The developer specified the event structure, roles, predicates, temporal conditions, and often scenario-specific annotations. The available perception implementations were also registered in advance as a small catalog of provider chains. In many workflow stages, only one chain was applicable, so the pipeline was effectively predetermined. Where several sources, provider variants, or placements remained feasible, the prototype checked and ranked those limited realizations using deployment state and profiled costs. It did not invent a new pipeline or search a large implementation space.

This distinction separates two forms of behavior. The developer determined *what the complex event meant* and which configured mechanisms could establish its predicates. The prototype automated *when the next authored stage became relevant* and, when a small choice existed, *which feasible realization to use*. We therefore present *FABLE* as an exploration of progress-driven orchestration within a curated design space, not as a general-purpose or production-ready complex-event system.

We study the prototype using 89 recorded traces spanning nine separately authored workflows. The traces are replayed through logical sensing and compute nodes, allowing the same data to be exercised under alternative activation policies and a controlled network disconnection. These experiments measure the behavior of the curated workflows and testbed. They do not establish that the prototype generalizes to unseen event definitions, large provider catalogs, or arbitrary physical deployments.

This chapter makes three contributions:

- We formulate progress-driven evidence activation as an orchestration question for complex events whose primitive observations must be produced by sensing and inference components.
- We develop an active-frontier prototype that automatically advances developer-authored CE workflows and activates the predefined provider work associated with their currently enabled predicates.
- We explore the resulting recognition–computation tradeoff in a controlled MQTT/replay testbed using 89 traces and nine separately authored workflow configurations, including a network-disconnection stress case.

The remainder of the chapter first defines the restricted problem considered by the prototype. It then describes Python-based workflow authoring, partial instances and frontiers, the limited provider-selection procedure, and the experimental testbed. Finally, it presents the controlled evaluation and discusses what the results do and do not establish.

4.2 Problem and Prototype Model

We consider a deployment containing sensing sources and compute nodes. A source has a modality and an associated node; a compute node has finite CPU, memory, and, where available, GPU capacity. Network links expose the connectivity and transfer conditions

used by the testbed. Recorded camera and audio streams can be replayed through the same source interfaces used by the prototype.

An evaluated complex event is represented by a developer-authored workflow A_q. The workflow declares participant roles, primitive predicates, logical composition, and temporal ordering or windows. It is constructed in Python and registered with the prototype before execution. The workflow is therefore an experimental input prepared by the developer, not the output of a user-facing language or automatic event-synthesis procedure.

For one possible occurrence of A_q, the prototype maintains a partial instance i. The instance records the workflow nodes already satisfied, participant bindings established by observations, relevant event-time intervals, and whether the occurrence remains active, has completed, or has expired. From this state, the prototype derives an active frontier

$$F(i) = \{(p, B, W)\},$$

where p is an enabled authored predicate, B contains any participant bindings already known to that predicate, and W is its applicable event-time interval. This frontier is the main interface between event progress and orchestration.

Each predicate p is associated with a small, developer-maintained set $\Pi(p)$ of predefined provider chains. A chain fixes a sequence of perception or reasoning components and their typed inputs and outputs. For example, a visual transition may use a detector followed by a tracker and a transition evaluator. The prototype does not synthesize such a chain. In several evaluated stages, $|\Pi(p)| = 1$, making the provider pipeline effectively fixed.

When a frontier predicate becomes active, the prototype constructs the feasible realizations of its registered chain or chains. A realization specifies the input source or retained artifact, the compute node used by each configured step, and any required transfer. In the small experimental catalog, the main remaining choices are often which source to inspect or where to place a known pipeline. Alternatives that violate availability, resource,

transfer, or timing constraints are removed. The remaining alternatives are ordered using a deterministic ranking over spatial preference, predicted completion, deadline slack, startup and resource use, and transferred bytes. If only one feasible realization remains, the ranking is vacuous.

The objective of the prototype is consequently modest: advance the authored workflow while avoiding provider work that is not currently useful. We do not claim to solve a general multi-tenant scheduling problem, discover new perception algorithms, or optimize over a large space of arbitrary dataflows. The experimental question is whether this restricted coupling between event progress and configured computation is useful in the nine workflows studied.

4.3 Prototype Overview

FABLE couples a semantic progress tracker to a distributed execution testbed. The semantic portion operates over a complex-event workflow authored in Python. It maintains candidate occurrences of that workflow and identifies the primitive predicates that could currently advance each candidate. The execution portion maps those predicates to predefined provider pipelines and runs the selected realization through logical sensing and compute nodes.

The resulting control loop is:

`Python CE workflow` → `partial instances` → `active predicates` → `configured provider pipeline` → `predicate results` → `updated instances`.

This loop contains genuine runtime automation. A developer does not manually invoke every successive stage after a trace begins. When a predicate result is accepted, the prototype advances or forks the affected instance, determines which authored predicates are enabled next, retires work that is no longer useful, and activates the corresponding provider work. The automation is nevertheless bounded by what the developer supplied:

the workflow, predicate parameters, registered provider chains, deployment description, and several scenario-specific annotations.

4.4 Developer-Authored CE Workflows

At the time of the evaluation, adding a new complex event required writing Python code. A workflow author declared event roles, added predicate nodes, and combined them using structures such as sequence, conjunction, alternatives, and bounded temporal relationships. The resulting graph was registered under a known event-family identifier. A request could select a registered family and provide supported parameters, but it could not supply a new arbitrary workflow through an external language.

This distinction is important because a family identifier such as `convoy` did not cause the prototype to infer what a convoy meant. It selected an existing Python implementation of that workflow. Similarly, an exact textual alias was only a convenient lookup for a known family. The prototype contained an interface through which a natural-language interpreter could have selected a registered family, but the evaluated implementation did not include a general natural-language-to-workflow compiler.

The nine evaluation workflows were authored and configured separately. Some shared semantic themes—for example, the two robbery workflows or several vehicle-interaction workflows—but each had its own concrete graph or profile. This authoring process determined much of the behavior later observed during execution.

Example: convoy. A convoy workflow specifies a leader vehicle observation followed by evidence about a distinct follower within a bounded interval; one version also includes a later scene-clear condition. The developer determines the roles, the ordering, the time window, and which predicate represents each stage. Once the workflow runs, however, a leader match automatically creates or advances a candidate and exposes the follower predicate as the next active work.

Example: robbery. The cross-sensor robbery workflow uses an alarm, gunshot, or other configured trigger and then requests vehicle evidence from a bounded interval. Its Python definition specifies the retrospective relationship, lookback interval, and identity checks. These are authored choices rather than behaviors inferred by the prototype. After the trigger is accepted, the controller automatically advances to the configured historical or departure stage.

4.5 Predicate Vocabulary and Workflow Structure

A primitive predicate describes an observation expected from a configured perception or reasoning component. Examples used by the workflows include vehicle crossings, entries and exits, movement, proximity, following, audio events, conversation, and identity association. Predicate schemas define their arguments and result types so that a provider result can be matched to the corresponding workflow node.

Roles provide continuity across stages. If a vehicle observation binds the role `leader`, a later FOLLOWS predicate can require its `leader` argument to refer to the same bound identity. A predicate may also introduce a new role, such as a possible follower. When several observations produce different valid bindings, the semantic runtime may keep separate candidate instances.

The workflow author is responsible for selecting the predicates and their parameters. Table 4.1 separates this authored content from the behavior performed automatically by the prototype.

The table also marks a limit of the approach. The prototype can reason about the progress of an authored graph, but it does not determine whether that graph is an adequate definition of the real-world event. A workflow that omits an important condition will still execute exactly as authored.

Developer-authored or configured	Performed by the prototype
CE roles, predicates, and graph structure	Maintain partial instances of that graph
Temporal windows and predicate parameters	Check enabled stages and temporal applicability
Provider chains and their typed interfaces	Activate a registered chain for an enabled predicate
Source, placement, or retrospective hints	Remove infeasible realizations and rank the remainder
Workflow-specific identity requirements	Apply results to compatible bindings and fork candidates

Table 4.1: Division between authored behavior and runtime automation in the evaluated prototype.

4.6 Partial Instances and Active Frontiers

For each possible event occurrence, the semantic runtime maintains a partial instance. An instance contains the satisfied workflow nodes, participant bindings, event-time intervals, and lifecycle state. Multiple instances may coexist when different observations could represent different occurrences or different participant assignments.

The *active frontier* is the set of primitive workflow nodes that are currently enabled for one partial instance. Sequence predecessors must be satisfied before a later predicate appears on the frontier. Children of a conjunction may appear together, while alternatives remain active until one branch succeeds. Temporal wrappers restrict the intervals during which their children can contribute. These operations are evaluated by shared runtime code rather than reimplemented as an imperative stage sequence for every trace.

When a provider result arrives, the runtime validates that it refers to a currently active predicate and compatible instance version. A true result updates the corresponding node, establishes or validates role bindings, and propagates progress through the authored graph. If the result introduces a new binding choice, the runtime may fork a new instance so that another candidate can remain possible. It then derives the next frontier. If the workflow root is satisfied, the instance completes; if a required window closes or a condition fails, the corresponding instance can no longer progress.

This automatic advancement is the principal implemented mechanism connecting CE

semantics to execution. It should not be confused with automatic event authoring. The runtime determines the next enabled node in a graph that the developer already supplied; it does not decide which real-world conditions ought to define the event.

4.7 From a Frontier to Executable Work

Each active primitive predicate is converted into a small execution request containing its predicate identifier, current bindings, applicable interval, and any configured source or placement restrictions. The controller consults a registry that maps the predicate to one or more predefined provider chains.

A provider chain is an authored pipeline, not a pipeline synthesized at runtime. For example, a visual relation may be registered as a detector, tracker, and geometry evaluator; an audio event may be registered as an audio classifier; and an identity predicate may use a registered descriptor and comparison path. The number of such chains was small. For several predicates there was one applicable chain in the evaluated configuration, so the chain itself was effectively the selected pipeline.

When a limited choice did exist, the prototype considered concrete realizations of those registered chains. Choices could include which camera or microphone supplied the input, whether a compatible retained artifact was available, and which configured node hosted a provider step. The feasibility checks accounted for node availability, resource capacity, data-transfer constraints, and the remaining useful time. A bounded ranking procedure then ordered the surviving realizations using a deterministic tuple that included spatial preference, predicted completion, deadline slack, startup and resource cost, and transferred bytes.

Calling this procedure a search should not imply a large or highly diverse design space. In the evaluated workflows, the semantic stage and registered catalog already determined most of the pipeline. The procedure was primarily a way to reject unavailable choices and select among a few source, provider, or placement variants. When only one feasible

realization existed, the procedure simply confirmed and dispatched it.

Provider activation and results. The controller sends activation commands to node agents, which launch or reuse the configured provider process and return predicate results. When the frontier changes, work associated only with the previous frontier can be stopped, while work still useful to another active instance may remain running. This realizes the chapter's central idea: provider activity follows the progress of the authored event rather than remaining fixed for the entire trace.

Deployment changes. The same feasibility path can be rerun when the testbed reports a node or link change. If another registered realization remains feasible, the controller can activate it; if no such realization exists, the affected evidence is lost. The prototype therefore supports limited replanning within its configured catalog, not general recovery from arbitrary infrastructure failure.

4.8 Replay and Controlled Testbed Operation

The orchestration core is connected to a testbed that replays recorded camera and audio streams through logical devices. MQTT carries control messages, provider status, and predicate results between the controller and node agents. The replay infrastructure makes it possible to run the same trace under different activation policies or network conditions without changing its underlying observations.

Replay serves two related purposes. First, it supplies the recorded streams used in the evaluation. Second, a workflow may explicitly request processing over a retained interval, as in the configured cross-sensor robbery workflow. Such retrospective behavior is authored into that workflow and uses the available replay/retention mechanism. The prototype does not infer that an arbitrary past interval should be searched or guarantee that every configured pipeline can be replayed.

The MQTT and replay mechanisms form experimental infrastructure around the core. They enable the prototype to coordinate logically distinct devices and exercise progress-driven activation, but they are not themselves research contributions of this chapter. The core was subsequently refactored, including the addition of a declarative CE language, while the general MQTT and replay approach remained similar. The evaluation in this chapter refers to the Python-authored core and workflow configurations used for the reported traces.

4.9 Prototype and Experimental Testbed

Python prototype. The evaluated implementation is written in Python. Its core components are a set of Python-authored CE workflow definitions, a semantic runtime that tracks partial instances and frontiers, a controller that maps active predicates to registered provider work, and node agents that execute that work. These components are sufficient to close the experimental loop from a provider result, through CE progress, to activation of the next configured pipeline.

Workflow definitions are ordinary Python modules. They construct graphs from roles, primitive predicates, logical composition, and temporal conditions and then register those graphs under known family identifiers. This arrangement made it possible to implement the nine evaluation workflows, but adding a new workflow required a developer to write and validate code. The evaluated version did not accept a declarative YAML or JSON CE specification.

Provider catalog and limited planning. Provider metadata record typed inputs and outputs, supported predicates, resource profiles, and permitted placements. Provider chains are registered separately as concrete pipelines. The controller does not assemble arbitrary chains from individual operations. It looks up the chain or small set of chains registered for the active predicate, enumerates feasible source and placement realizations,

and ranks the resulting candidates. In many stages of the evaluated workflows there is only one applicable chain and only a small number of placement choices.

The ranking logic uses profiled startup and execution time, estimated transfer cost, resource availability, deadline slack, and configured spatial preferences. These values help choose among the few remaining realizations; they should not be interpreted as evidence that the prototype solves a broad pipeline-synthesis problem.

MQTT control and result exchange. The controller and node agents communicate through Eclipse Mosquitto. Low-rate activation commands and predicate results use MQTT QoS 1, while recurring telemetry can use QoS 0. Messages include identifiers for the request, hypothesis, frontier or demand, and provider execution so that a redelivered message can be recognized. Node agents also publish provider and node status used by the controlled deployment experiments.

MQTT is an engineering substrate rather than part of the orchestration contribution. It provides a convenient way to connect logical devices and provider processes, but the chapter does not evaluate MQTT against alternative messaging systems or claim a new transport protocol.

Node agents and provider execution. Each logical sensing or compute node runs an agent capable of starting and stopping configured provider processes. Providers are packaged as Python or containerized components with explicit inputs and outputs. Example visual pipelines include object detection, tracking, and a predicate-specific evaluator; audio workflows use configured audio classifiers or interaction components. The testbed also includes identity and geometric components needed by particular workflows.

The provider implementations and chains were prepared for the evaluated scenarios. Although the registry exposes a uniform interface, the evaluation does not demonstrate that arbitrary third-party providers can be added without additional integration, profiling, and workflow-specific validation.

Replay and retained intervals. Recorded camera and audio data are replayed through source adapters connected to the node agents. Replay preserves the experimental event ordering while allowing different activation policies to process different portions of the same underlying trace. When an authored workflow requests evidence from a bounded earlier interval, the testbed can replay the corresponding retained data through a compatible configured pipeline. This capability is used as a case-specific mechanism; the prototype does not provide a general historical query service over arbitrary providers and intervals.

Control state. The code includes persistence and versioned-message mechanisms for recording controller state and avoiding repeated application of stale results. These mechanisms support the prototype's controlled runs, but restart recovery, replicated control, and long-running fault tolerance are not evaluated as independent system properties. Accordingly, we treat them as supporting implementation rather than contributions established by the experiments.

Controlled deployment. The reported experiments run on a workstation with an Intel Core i7-5930K CPU at 3.50 GHz, two NVIDIA RTX A5000 GPUs, 64 GB of RAM, NVIDIA driver 550.54.15, CUDA 12.4, and Ubuntu 20.04. Logical sensing, edge, and server roles are represented by separate processes or containers. Mininet-WiFi 2.6 is used where the experiment requires a controlled change in link availability. Although the code and testbed were developed with heterogeneous edge devices in mind, the reported evaluation does not constitute a deployment study across a fleet of Raspberry Pi, Jetson, and server nodes.

Instrumentation. The evaluation records CE outcomes together with CPU time, GPU time, and the number of processed video frames. These measurements are sufficient for the recognition–computation comparison reported in Section 4.11. The prototype contains additional status and timing hooks, but this chapter does not use their existence to claim

a comprehensive performance, reliability, or scalability evaluation.

Artifact version and later refactoring. After the reported work, the FABLE core was refactored. The later version adds a declarative CE language and reorganizes the semantic, planning, and execution layers. The MQTT communication and replay approach remain broadly similar, but the later core should not be read backward into the evaluation reported here. This chapter describes the Python-authored workflow implementation that produced the 89-trace results.

4.10 Related Work

Complex event processing and spatiotemporal monitoring. Complex event processing (CEP) systems provide languages and runtimes for detecting temporal patterns over structured event streams. SASE and Cayuga support filtering, sequencing, correlation, and stream-derived event generation (Gyllstrom et al., 2006; Demers et al.). CORE improves the representation and evaluation of partial matches (Bucchi et al., 2021), while spatiotemporal monitoring languages add spatial relations and distributed monitoring for cyber-physical systems (Ma et al., 2020b; Bartocci et al., 2017). These systems motivate FABLE's use of partial event progress. The question explored here is narrower and complementary: when primitive observations must be produced by perception components, can the current stage of an authored CE workflow determine which configured component should be active?

Unlike these CEP systems, the evaluated FABLE prototype does not contribute a general event language. Its nine workflows are implemented separately in Python. The comparison is therefore about the use of event progress as an execution signal, not about language expressiveness or general-purpose stream processing.

Complex-event recognition over video and multimodal streams. VidCEP and VEKG represent video streams using detected objects, relations, and spatiotemporal

graphs (Yadav and Curry, 2019; Yadav et al., 2020), while Bobsled targets scalable CEP over video streams (Han et al., 2025a). Neurosymbolic systems combine learned perception with symbolic event structure, including DeepCEP, Neuroplex, DeepProbCEP, and DLACEP (Xing et al., 2019, 2020; Vilamala et al., 2023; Amir et al., 2022). Other work evaluates neural, neurosymbolic, and foundation-model approaches for online multimodal CE detection (Han and Srivastava, 2024; Han et al., 2025b), and NeuroFlinkCEP integrates neural models with FlinkCEP across IoT platforms (Ntouni et al., 2025).

FABLE adopts the familiar separation between symbolic event structure and perception. Its prototype contribution is to expose the currently enabled primitive predicates of a partial instance to the execution controller. The provider implementations themselves are preconfigured, and the experiments do not show automatic construction of new perception functions.

Adaptive video analytics and edge execution. VideoStorm, Chameleon, NoScope, BlazeIt, FilterForward, Reducto, and EdgeVision adapt model configurations, filtering, or distributed execution under compute, bandwidth, and latency constraints (Zhang et al., 2017b; Jiang et al., 2018b; Kang et al., 2017, 2018; Canel et al., 2019; Li et al., 2020; Gao et al., 2024). Caesar and Argus coordinate cross-camera inference for complex activities and collaborative analytics (Liu et al., 2019; Yi et al., 2024). Nexus and INFaaS select resources or model variants for inference requests (Shen et al., 2019; Romero et al., 2021), while Mainstream shares common DNN computation across video applications (Jiang et al., 2018a).

These systems generally begin with a query or inference request that is already active. FABLE explores whether a hand-authored event workflow can decide when such a request becomes useful. The distinction should not be overstated: the evaluated catalog contains few provider chains, and for many predicates the pipeline is effectively fixed. FABLE's limited planner mainly chooses among available sources, provider variants, or placements for the current stage.

Distributed dataflow and scheduling. Stateful dataflow systems address event time, logical progress, and distributed execution. Naiad uses frontiers to summarize outstanding logical timestamps, while MillWheel provides persistent per-key state, timers, and fault-tolerant event-time processing (Murray et al., 2013; Akidau et al., 2013). FABLE's term *frontier* has a different local meaning: it is the set of authored primitive predicates currently able to advance one candidate CE occurrence.

General schedulers such as HEFT and Decima decide where known DAG tasks should execute (Topcuoglu et al., 2002; Mao et al., 2019), and Eddies reorders known relational operators as runtime conditions change (Avnur and Hellerstein, 2000). FABLE does not compete with these systems as a general scheduler. Its experiment couples a small placement/ranking procedure to semantic progress so that configured work can be activated and retired as a workflow advances.

Positioning. The relevant ideas have substantial precedent: CEP engines track partial matches, video systems adapt inference, and schedulers choose among placements. FABLE combines these ideas in an exploratory prototype whose partial CE state controls the activation of predefined sensing and inference pipelines. The novelty claimed here is this coupling and its experimental illustration, not a new CE language, a large provider-search space, or a production distributed runtime.

4.11 Exploratory Evaluation

The evaluation examines the behavior of the prototype in its curated workflow and provider environment. It does not evaluate whether an arbitrary user can author a new CE or whether the planner scales to a large provider catalog. Within this scope, we ask two questions. **RQ1** asks what recognition–computation tradeoff is observed when provider activation follows the progress of the nine authored workflows. **RQ2** asks how the prototype behaves in a controlled replay when one event-relevant network link becomes

Table 4.2: Recorded traces used in the exploratory FABLE evaluation. Each row uses a separately authored workflow configuration.

Authored workflow	**Traces**
Cross-sensor robbery	4
Pass–follow–clear convoy	6
Robbery with alarm	16
Route convoy	25
Talking/rendezvous	8
Three-visit stalking	3
Two-vehicle chase	19
Vehicle convergence	5
Vehicle rendezvous	3
Total	**89**

unavailable.

4.11.1 Methodology

Trace corpus and workflows. We use approximately 2.5 hours of recorded camera and audio data collected across three experimental sessions at two military test sites. The corpus contains 89 CE traces spanning the nine workflows listed in Table 4.2.

Each workflow was implemented and configured separately in Python. The graph specified its primitive predicates, roles, ordering, temporal conditions, and any workflow-specific execution annotations. The corresponding provider pipelines and relevant sources were also prepared in advance. The 89 traces therefore test repeated executions of nine curated workflows; they are not 89 independently supplied CE programs or evidence of coverage over an open-ended event language.

Replay testbed. The traces are replayed on a single GPU-equipped desktop. Logical sensing and compute devices run as separate processes or containers and exchange control and results through MQTT. This arrangement exercises the controller/node-agent path and permits repeatable changes to provider activation. For the network-disconnection experiment, Mininet changes link availability between logical devices while the underlying recorded trace remains fixed.

Policies. All policies use the same authored workflows and atomic-event detectors. They differ in when and where configured evidence-producing work is active.

- **B0: Produce-All** runs the configured atomic-event detectors broadly, making potentially relevant evidence available throughout the trace at the cost of additional computation.
- **B1: Static** runs detectors only on a fixed set of devices selected for the workflow. This reduces work but does not change the active configuration as the workflow advances.
- ***FABLE*** activates the predefined provider work associated with the current frontier. Where more than one configured source or placement remains feasible, it ranks those limited realizations; otherwise the applicable pipeline is effectively predetermined.

These baselines are intentionally simple. They isolate the effect of progress-driven activation in the configured testbed, but they do not compare FABLE with a general adaptive scheduler that has an equally rich view of runtime conditions.

Metrics. For end-to-end recognition, we report CE recall together with CPU time, GPU time, and the number of processed video frames. These measurements expose the observed recognition–computation tradeoff. For the network experiment, we report recall under nominal connectivity and after a persistent link disconnection. The network experiment uses a 24-trace subset of the full corpus.

4.11.2 RQ1: Recognition and Computation

Question. What recognition–computation tradeoff is observed when provider activity follows progress through the nine authored workflows?

Table 4.3 reports the aggregate result. In this configuration, *FABLE* reaches 85.7% recall, compared with 72.4% for B0 and 71.4% for B1. It processes 1,938 frames, compared

Table 4.3: Observed end-to-end recognition and computation over the curated workflow evaluation.

Policy	Recall (%)	CPU (s)	GPU (s)	Frames
B0: Produce-All	72.4	359	35.2	4,287
B1: Static	71.4	146	13.5	905
FABLE	**85.7**	**250**	**22.0**	**1,938**

with 4,287 for B0 and 905 for B1.

Relative to Produce-All, the progress-driven configuration uses less measured computation: GPU time falls from 35.2 s to 22.0 s, CPU time falls from 359 s to 250 s, and the number of processed frames falls from 4,287 to 1,938. The result is consistent with the intended effect of frontier-driven activation: configured components remain inactive during workflow stages in which their evidence cannot yet contribute.

The Static policy uses the least computation, but it also has the lowest recall. In the evaluated configurations, activating additional work when the authored workflow reaches the corresponding stage recovers evidence missed by the fixed device selection. FABLE therefore occupies an intermediate compute point while obtaining the highest observed recall.

These results require a deliberately limited interpretation. The comparison reflects the particular nine workflow implementations, provider configurations, resource limits, and traces. Because the workflows and provider pipelines were curated, the table does not demonstrate that FABLE will select useful pipelines for an unseen CE or outperform every static or adaptive policy. Nor does the higher recall establish that progress-driven execution intrinsically improves detector accuracy. It shows that, in this testbed, the timing and location of configured evidence production affected whether enough evidence was available to complete the authored workflows.

4.11.3 RQ2: Controlled Network Disconnection

Question. How does the configured prototype behave when one event-relevant link becomes unavailable during replay?

Table 4.4: Observed FABLE recall under a persistent logical link disconnection. The experiment uses a 24-trace subset of the corpus.

Condition	**Recall (%)**
FABLE (nominal)	83.3
FABLE (disconnect)	66.7

We apply a persistent link disconnection to a 24-trace subset. The sensing and orchestration processes remain containerized on the desktop, while Mininet removes one logical link during the trace. Table 4.4 compares nominal and disconnected operation.

Recall decreases from 83.3% to 66.7%. The controller can reconsider the configured realization after the topology change, and some traces still have a feasible path to the required evidence. Other traces lose evidence for which the small provider/source catalog contains no substitute. The result therefore illustrates limited adaptation within the prepared deployment; it does not establish general fault tolerance or independence from a particular sensing layout.

4.11.4 What the Evaluation Establishes

Taken together, the experiments show that the prototype can execute nine separately authored CE workflows through the replay testbed, use semantic progress to change which predefined provider work is active, and continue some executions after a controlled topology change. They also show an empirical recognition–computation tradeoff for these configurations.

The experiments do not establish usability by non-developer authors, semantic generality beyond the nine workflows, scalability to many concurrent requests, or the value of searching a large provider space. In many workflow stages no such large search exists: the registered provider chain is effectively fixed, and the prototype chooses only among a few source or placement realizations.

4.12 Discussion and Limitations

Research prototype rather than general-purpose system. The strongest conclusion supported by this chapter is that CE progress can be used to control configured evidence-producing work in a prototype testbed. FABLE automatically advances partial instances and activates the next authored stage, but the developer has already specified the workflow and the small set of provider pipelines available to it. The evaluation does not demonstrate a deployable service into which an arbitrary user can submit a new CE.

Manual workflow authoring. Each of the nine evaluated workflows was implemented separately in Python. This authoring determined participant roles, predicates, temporal structure, and several execution-relevant annotations. As a result, the workflow author performed substantial semantic and systems design before the runtime began. The prototype does not test whether these decisions can be obtained reliably from a concise user specification or transferred unchanged to a new site.

The post-defense refactor introduces a declarative CE language, but that work is outside the evaluation reported here. It should be assessed separately for expressiveness, authoring burden, compilation correctness, and compatibility with the execution layer.

Small provider and alternative space. The provider catalog contains only a limited number of chains relevant to the evaluated predicates. For many stages there is one applicable chain, making the pipeline effectively predetermined. Even when several realizations exist, they often differ only in source, provider variant, or placement. The ranking procedure is useful for enforcing feasibility and choosing among those options, but the experiments do not establish the value of sophisticated search over a large or diverse provider space.

A broader claim would require many interchangeable implementations per predicate, independently measured quality and cost profiles, and experiments showing that the selected realization materially differs from a simple preconfigured choice. This remains

future work.

Controlled replay and logical distribution. The replay testbed provides repeatability and exercises communication between the controller, node agents, and provider processes. However, the reported evaluation runs logical devices on one GPU-equipped workstation. Mininet controls link availability, but it does not reproduce every source of heterogeneity, contention, clock behavior, or failure found in a physical deployment. The results therefore characterize the experimental configuration rather than a fielded edge system.

MQTT and replay are stable and useful pieces of the testbed, but they are not the subject of the research claim. The evaluation does not compare messaging systems, establish high-availability control, or measure long-duration operation.

Evaluation scope. The 89 traces provide repeated evidence across nine workflows, which is more substantial than a single demonstration. Nevertheless, all workflows and provider mappings were curated by the developers, and all traces come from a small number of collection sessions. The observed recall and resource results may depend on those configurations. Testing unseen workflows, independently authored definitions, additional deployments, and stronger adaptive baselines would be necessary to establish generality.

Provider accuracy. FABLE consumes thresholded predicate results; it does not make an incorrect detector correct. False positives or false negatives can create, advance, or miss partial instances. The reported end-to-end recall therefore combines the behavior of the authored workflow, the configured providers, and the orchestration policy. It should not be interpreted as an isolated measurement of any one component.

Automatic behavior that remains meaningful. The limitations do not reduce the prototype to a manually stepped script. Within a configured workflow, the semantic runtime automatically maintains candidate instances, applies bindings, advances graph

state, derives new frontiers, and triggers replanning. This closed loop is the concrete mechanism the chapter contributes. The appropriate claim is that the mechanism is demonstrated in a curated testbed, not that it already constitutes a broadly usable system.

Path toward a fuller realization. A stronger system would combine the later declarative language with a larger and independently maintained provider ecosystem, explicit validation of provider semantics, heterogeneous physical deployment, stronger adaptive baselines, and evaluation of concurrency and failure recovery. Such work would test whether the frontier abstraction remains useful once the manually curated parts of the present prototype are relaxed.

4.13 Conclusion and Future Work

This chapter explored whether the progress of a complex event can guide which sensing and inference computation is active. FABLE represents possible event occurrences as partial instances and exposes their currently enabled authored predicates as active frontiers. When a predicate result advances an instance, the prototype automatically derives the next frontier and activates the predefined provider work associated with it.

The implementation is best understood as a research prototype and orchestration testbed. Its nine workflows were separately authored in Python, and its provider chains were registered in advance. For many predicates, the applicable chain was effectively predetermined. A bounded ranking procedure selected among the few remaining source, provider, or placement realizations when a choice existed. FABLE therefore automated progress and activation within a developer-defined envelope; it did not provide a general CE language or synthesize arbitrary pipelines.

Across 89 recorded traces, the progress-driven configuration achieved the highest observed recall while using less measured computation than the Produce-All baseline. A controlled network-disconnection experiment further showed that some workflows could

continue when another configured realization remained available. These results establish feasibility for the curated workflows and testbed, not generality to unseen events or deployments.

The core research lesson is consequently narrower than the claim of a complete distributed runtime: semantic progress can be made operational as a signal for starting and stopping configured evidence-producing work. Future work must determine whether this idea remains effective with independently authored event specifications, larger provider catalogs, heterogeneous physical devices, concurrent workloads, and stronger adaptive baselines. The post-defense refactor's declarative CE language is one step toward that broader realization, but it is not part of the evaluation reported in this chapter.

Within the dissertation, FABLE provides an exploratory execution counterpart to *IncidentLens* and *privMediator*. Together, the three works illustrate how event semantics and application context can influence the interpretation of observations, the information disclosed to applications, and the computation activated during event recognition. Chapter 5 synthesizes these roles and discusses the additional work required for an integrated event-centric sensing system.

CHAPTER 5

Conclusion and Future Work

This dissertation began from a gap between the abstractions exposed by complex event processing and the realities of distributed sensing. Complex event processors are most naturally described as operating over meaningful primitive event streams, yet distributed sensing applications often begin with incomplete, heterogeneous, and potentially sensitive observations. Before an application can use a complex event, a sensing system may need to determine how observations should be interpreted, what representation may cross the application boundary, and what sensing and computation are currently useful for recognition.

The central argument of the dissertation is that these decisions should be explicit and revisable rather than selected entirely before recognition begins. The dissertation uses *complex event construction* to describe the process of turning distributed observations into an application-usable event abstraction through three complementary pillars. Event interpretation determines which event explanations remain plausible and how observations relate to them. Application-visible representation determines which form of event-relevant information may be disclosed to an application. Distributed execution determines which sensing and inference computations should run as an event and its deployment conditions change.

The three principal contributions study these pillars separately. *IncidentLens* con-

structs and revises explanations for latent incidents from heterogeneous urban effects. *privMediator* constructs application-visible outputs that satisfy both utility requirements and configured contextual constraints. *FABLE* explores how progress through a specified, developer-authored event workflow can guide which configured sensing and inference work is useful next. Studying the pillars independently makes it possible to isolate their assumptions, mechanisms, and evaluation criteria, while recognizing that the supporting artifacts have different scopes and levels of maturity. A deployed sensing infrastructure, however, may need to address all three pillars at once. A smart city responding to an emerging incident may need to distinguish among several explanations, disclose different information to emergency services, infrastructure operators, researchers, and public applications, and reorganize sensing and computation as the incident area and available resources change. The dissertation develops the individual ideas and mechanisms needed for such a system, while their holistic integration remains future work.

5.1 Summary of Contributions

Event interpretation from distributed effects. *IncidentLens* addresses incidents whose types, sources, extents, progression, and observation membership are not directly given. It normalizes multimodal reports into semantic effects and maintains competing incident hypotheses whose support changes as new observations arrive. Incident-specific templates express source priors, propagation behavior, clustering, composition, and stability within a shared online engine. This design improves event-level discovery and false-positive control relative to methods that treat observations independently or cluster them using generic spatial and temporal proximity. More broadly, it demonstrates that a system can preserve and revise uncertainty about the event itself rather than forcing each observation into an immediate local classification.

Application-visible representation under utility and information-use constraints. *privMediator* addresses the assumption that authorization for an application

implies entitlement to a predetermined sensor stream. It separates an application's utility contract from the technical representation used to satisfy it and models sensor-processing operators through typed input, output, utility, transformation, and residual-disclosure effects. Candidate pipelines are checked against configured contextual-integrity rules and ranked by characterized residual exposure. No release is made when no feasible candidate satisfies both the application requirements and the configured rules. The resulting architecture makes transformed release and refusal explicit system outcomes. It also clarifies the role of learned models in privacy mediation. Models may assist with bounded interpretation or candidate generation, while final authority remains with inspectable contracts, rules, and selection logic.

Distributed execution guided by event progress. *FABLE* explores the assumption that configured computations for event recognition need not remain continuously active. At the time of the reported evaluation, each of nine complex-event workflows was authored separately in Python, including its predicates, participant roles, ordering, temporal conditions, and applicable provider work. The prototype represents possible occurrences as partial, entity-bound instances and identifies their currently enabled conditions as an *active frontier*. When a predicate result is accepted, the prototype advances the corresponding instance and activates the predefined provider work associated with its next frontier. The registered provider catalog is small and often contains one effectively predetermined chain; when several sources, provider variants, or placements are feasible, the prototype ranks only those limited realizations. Experiments on 89 recorded traces use a controlled MQTT/replay testbed to examine the recognition–computation tradeoff of progress-driven activation and behavior during a logical network disconnection. These experiments show that event progress can serve as an orchestration signal within the curated workflows; they do not establish a general-purpose event language, large-space pipeline planner, or production distributed runtime.

5.2 Lessons Learned

Although the contributions differ in domain, mechanism, and maturity, several lessons recur.

5.2.1 Application requirements should be independent of technical realization

A predetermined pipeline often conflates what an application requires with how the system currently provides it. The dissertation separates these concerns at three levels. An incident hypothesis is independent of any one observation or modality. An application utility contract is independent of any one media or semantic representation. In the *FABLE* exploration, an authored complex-event predicate identifies the semantic condition to be established, while registered provider work describes a configured way to produce evidence for that condition. This separation permits a system design to revise a technical realization without redefining the application's semantic objective.

The separation also supports system evolution. A new urban data source can support an existing incident hypothesis without changing the incident definition. A new privacy operator can become a candidate representation without changing the application task. In principle, additional predicate providers can offer new realizations of an existing recognition condition. The thesis-era *FABLE* prototype exercises this last separation only within a small, manually registered catalog, so general provider extensibility remains a design direction rather than an evaluated result. In each case, a stable semantic interface can limit the scope of change.

5.2.2 Interpretation, representation, and execution should remain revisable

The appropriate event explanation, application-visible output, or configured execution choice often cannot be selected before observations arrive. *IncidentLens* maintains competing incident hypotheses and revises them as additional observations support or con-

tradict possible explanations. *privMediator* constructs multiple type-compatible pipelines and selects an output only after utility, residual disclosure, and application context have been evaluated. In its curated workflows, *FABLE* changes which predefined provider work is active as partial event occurrences advance and, when a limited choice exists, ranks a small number of feasible sources, providers, or placements using the available deployment state.

Maintaining revisable decisions has costs. Candidate spaces must be bounded, retained state must expire, and decisions require provenance. Premature selection can nevertheless be more damaging. It can merge unrelated observations into one incident, disclose a more revealing output than an application requires, or activate an expensive provider before any possible event occurrence needs it. Across the contributions, alternatives are therefore paired with mechanisms such as pruning, rejection, expiration, or cancellation, although their implementations and evaluations differ in scope.

5.2.3 Negative outcomes are part of correctness

Sensing systems are often evaluated primarily by the positive outputs they produce. This dissertation treats several negative outcomes as first-class decisions. An incident hypothesis may remain unconfirmed when available observations are inadequate. An application request may be rejected when no output is both useful and permitted. An event occurrence may expire when its temporal requirements can no longer be satisfied, and a provider may be stopped when its result can no longer advance any active occurrence.

Such outcomes are not equivalent to implementation failure. They prevent unsupported explanations, inappropriate information release, and irrelevant computation. Evaluation should therefore report refusal, expiration, and false-positive behavior alongside the quality of successful outputs. Applications also need interfaces for responding to negative decisions rather than assuming that every request produces data.

5.2.4 Learned components require structured boundaries

The precursor systems showed that language and vision-language models can supply useful world knowledge, extract semantic observations, and propose technical actions. They also exposed variability, incomplete calibration, and limited auditability. The principal contributions therefore place learned components behind closed schemas and deterministic checks. Models may emit bounded observation labels, assist with offline template authoring, or provide advisory norm judgments. They do not directly create unrestricted incident records or override hard privacy rules. In the thesis-era *FABLE* prototype, workflows and available provider behavior are registered by the developer rather than invented by a learned model at runtime.

This pattern does not remove model error. It narrows the effect of error and preserves the information needed to inspect a decision. The appropriate balance between learned flexibility and structured authority is likely to remain a central design question as sensing systems adopt increasingly capable foundation models.

5.2.5 Complex-event construction should expose provenance

Each principal contribution exposes a different form of decision trace. Incident hypotheses retain the observations and propagation assumptions that support them. Privacy candidates retain operator sequences, residual vectors, induced information flows, and rule outcomes. The *FABLE* testbed exposes partial-workflow progress, frontier changes, provider activations, and predicate results during controlled runs. These traces support debugging and evaluation, but they are also important for systems that affect safety, privacy, and resource allocation. The *FABLE* experiments do not establish a durable, general-purpose provenance subsystem; rather, they illustrate which execution decisions a fuller system would need to make inspectable. An application-usable event abstraction is more trustworthy when an operator can determine how it was constructed, which observations and rules affected it, and which assumptions remain uncertain.

5.3 Toward Holistic Complex-Event Construction in Smart Cities

The dissertation presents complementary contributions rather than an implemented end-to-end platform. Smart city sensing provides a concrete setting in which their concerns would need to be addressed together. Consider an emerging urban emergency observed through environmental sensors, cameras, traffic measurements, infrastructure reports, and public reports. The observations may initially support several incident explanations and possible origins. Different applications may require different event attributes or supporting observations under different legal, organizational, and social constraints. As the possible incident develops across a city, the relevant sensing regions, recognition models, network paths, and computing resources may also change.

An integrated architecture would connect the three pillars through a shared application requirement while preserving their distinct responsibilities. An event interpretation layer could maintain competing incident hypotheses and identify the observations or inferences that would distinguish them. A distributed execution layer could translate those recognition needs into sensing and inference demands, select compatible providers, and organize their execution across available resources. An application-visible representation layer could govern which observations, intermediate inferences, or event results may cross each trust and application boundary.

The interactions would be bidirectional rather than a simple sequence of post-processing stages. An incident hypothesis could determine which observations should be sought and which locations and time periods remain relevant. Available providers and sensing coverage could determine which hypotheses can currently be tested and how much confidence should be placed in missing observations. Information-use constraints could prohibit a provider, placement, or application-facing output even when it would otherwise offer high recognition utility. A less revealing provider might be selected when it satisfies the same recognition requirement, while an internal inference might be permitted to advance an event without allowing its supporting raw observations to leave

a trusted domain.

Several challenges prevent this composition from being straightforward. Incident hypotheses are open-world and uncertain, while the thesis-era *FABLE* experiments use bounded, separately authored workflows and a curated provider catalog. Contextual information-flow rules refer to actors, purposes, spaces, and transmission principles that are not normally represented in execution contracts. Joint decision making across event likelihood, disclosure, quality, latency, energy, and bandwidth creates large candidate spaces whose objectives may be only partially ordered. Observations collected to test one hypothesis may also create new disclosure risks or support secondary event inferences. The dissertation isolates the three pillars so that each challenge can be studied independently. A holistic architecture must preserve their distinct assumptions and guarantees while allowing the decisions to influence one another.

5.4 Limitations

The principal limitation of the dissertation is that the three pillars are studied through separate artifacts in different sensing settings and at different levels of maturity. *IncidentLens* studies latent incident interpretation in urban sensing, *privMediator* studies application-visible outputs in privacy-sensitive shared environments, and *FABLE* uses a research prototype and controlled MQTT/replay testbed to explore progress-driven execution for curated tactical workflows. The results provide different forms of evidence for making each decision explicit and revisable, but they do not establish the behavior, scalability, or correctness of a single system that jointly reasons about interpretation, information use, and execution. In particular, the *FABLE* evaluation does not establish usability for arbitrary event authors, general scheduling performance, or operation across arbitrary physical deployments. The smart city architecture described above is therefore a synthesis of the dissertation's abstractions and findings, not an evaluated end-to-end contribution.

The contributions also share several individual limits. First, their semantic vocabularies and contracts are authored for bounded domains. Incident templates cannot cover every urban phenomenon, operator effects can omit unknown inferences, and *FABLE*'s workflow definitions and provider mappings are manually prepared for a small set of scenarios. Structured boundaries make omissions visible but do not eliminate them.

Second, evaluation depends on imperfect proxies. Real incident labels are incomplete and spatially coarse. Privacy rules and participant judgments represent configured and perceived appropriateness rather than universal norms. Controlled replay and network emulation do not reproduce every failure mode of a long-lived deployment. For *FABLE* in particular, logical sensing and compute nodes are exercised on a controlled replay platform; this supports repeatable comparisons but is not equivalent to a longitudinal physical deployment. The dissertation uses synthetic data, weak labels, probes, ablations, and controlled experiments to triangulate behavior, but broader longitudinal deployments remain necessary.

Third, the contributions assume trusted control components. *IncidentLens* assumes the integrity of observation metadata and extraction infrastructure. *privMediator* assumes a trusted mediator, operator catalog, rule library, and enforcement boundary. The *FABLE* prototype assumes trusted workflow configuration, provider registrations, and testbed components; it does not evaluate adversarial nodes, providers, or control messages. Compromised sensors, deceptive applications, adversarial providers, and side channels require complementary security mechanisms.

Finally, the contributions make selective use of learning but do not solve continual semantic evolution. New incident types, changing social norms, novel sensor inferences, and previously unseen provider behavior can invalidate existing templates and contracts. Safe mechanisms for detecting specification drift and incorporating new knowledge are therefore as important as adaptation within a fixed, authored specification.

5.5 Future Research Directions

5.5.1 Integrated smart city event construction

The most direct continuation of this dissertation is an end-to-end smart city sensing architecture that combines the three pillars around shared event and application requirements. Such a system would maintain competing incident explanations, derive recognition demands from unresolved hypotheses, select and execute compatible sensing operations, and govern each internal and external information flow. An evaluation would need to measure not only event quality, disclosure, and resource use independently, but also how a decision in one pillar changes the feasible choices and outcomes in the others.

A useful first step would be a bounded urban emergency application with a small family of incident hypotheses, explicit application roles, a typed provider catalog, and a controlled set of information-flow rules. This setting would permit experiments on whether joint reasoning improves recognition timeliness or resource use, whether privacy constraints change provider selection and hypothesis confidence, and whether active sensing creates information that is useful for one application but inappropriate for another.

5.5.2 Joint hypothesis and observation planning

IncidentLens currently reasons primarily over observations that have already arrived. A natural extension is active observation acquisition. Incident hypotheses could identify which observation would most reduce uncertainty, and an event-driven runtime could decide whether the expected reduction justifies activating a sensor or model. This would connect abductive event interpretation with resource-aware sensing and permit the system to distinguish lack of supporting observations from lack of sensing coverage.

5.5.3 Coverage-aware event confidence

All three contributions would benefit from stronger representations of observability. Incident confidence should account for where sensors could have observed an effect, not only whether an effect was reported. Negative and duration predicates in a fuller realization of *FABLE* require operational coverage to distinguish "not observed" from "observed to be absent." Privacy mediation may also need to account for alternate sensors or correlated outputs that undermine the intended minimization. A shared coverage model could make these distinctions explicit across interpretation, execution, and disclosure.

5.5.4 Verified operator and provider contracts

Typed contracts are central to *privMediator*, while *FABLE*'s registered provider descriptions play a related but more limited role in connecting authored predicates to configured implementations. Their correctness is largely established through authoring, tests, and empirical profiles. Future systems could combine static analysis, runtime monitoring, differential testing, and learned anomaly detection to verify that an operator or provider behaves consistently with its declared input, output, resource, quality, and disclosure effects. Contract violations could trigger conservative risk escalation, provider quarantine, or a new execution decision.

5.5.5 Multi-party and evolving information norms

The privacy chapter evaluates a configurable rule library for a bounded scenario set. Shared environments can contain subjects with conflicting preferences and changing roles, and some norms cannot be reduced to one sender and recipient flow. Future mediation should support collective constraints, authority delegation, appeals, temporary exceptions, and explicit uncertainty about normative judgments. The least-disclosing feasible output may also depend on what an application has learned previously, requiring history-sensitive disclosure accounting rather than independent decisions for each request.

5.5.6 Decentralized and resilient event execution

The *FABLE*testbed uses a central orchestration process to coordinate its configured workflows. Highly disrupted or peer-to-peer environments may not have continuous access to such a coordinator. Future work could distribute partial-instance ownership, allow local frontier decisions under bounded autonomy, and reconcile event state after network partitions. This would first require explicit and validated continuation-state contracts beyond what the thesis-era prototype establishes. Decentralized execution must also address conflicting versions, duplicate work, privacy domains, and strategic or faulty participants.

5.5.7 Human interaction with complex-event construction

Operators and affected users need ways to inspect and correct constructed events without understanding every internal model. Incident systems could expose competing explanations and the observations that would distinguish them. Privacy systems could explain why an output was selected or why a request was refused. Event runtimes could show which recognition requirement is blocking progress and what resource or coverage limitation is responsible. Designing these interfaces is not only a usability problem. Human feedback can become a structured input for revising hypotheses, rules, and contracts.

5.6 Concluding Remarks

Complex events are often presented as patterns recognized after sensing and preprocessing have already occurred. This dissertation instead treats the process connecting distributed observations to an application-usable event abstraction as a systems problem. Event interpretation determines how observations belong together and what occurrence they may describe. Application-visible representation determines which form of event-relevant information may cross an application boundary. Distributed execution determines which sensing and inference work is useful as an event unfolds.

The three contributions do not constitute a fully integrated platform, and they provide different forms of evidence for their respective pillars. *IncidentLens* and *privMediator* develop systems for revising event explanations and application-visible representations within their evaluated domains. The *FABLE* prototype shows, for nine curated workflows, that partial event progress can drive changes in predefined provider activity within a controlled testbed. Future sensing infrastructures, particularly smart cities and other dynamic distributed environments, will need to make these decisions together. A single urban incident may require the system to revise competing explanations, provide different outputs to applications with different responsibilities and constraints, and reorganize computation as the event and deployment conditions change.

As sensing deployments become more heterogeneous, autonomous, and embedded in shared physical spaces, the distance between raw observations and application intent will continue to grow. Bridging that distance requires systems that reason not only about whether an event occurred, but also about how the event was interpreted, what representation was appropriate, and what work was necessary to establish it. Making those choices explicit and connecting them through a common event abstraction is a step toward sensing infrastructures that are more adaptive, accountable, privacy-aware, and worthy of trust.

References

ALERTCalifornia. 31

Caltrans Streaming Video Locations. 31

Citizen: Keeping you safe & informed. 31

Current weather and forecast - OpenWeatherMap. 31

Data: CAMEO. 5

The GDELT Project. 31

Real-time Air Quality Monitoring by PurpleAir. 31

Semantic Sensor Network Ontology. 5

(2026). Prolific. Online participant recruitment platform. 54, 74

Abdi, N., Zhan, X., Ramokapane, K. M., and Such, J. (2021). Privacy Norms for Smart Home Personal Assistants. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–14, New York, NY, USA. Association for Computing Machinery. 76

Agrawal, R. and Srikant, R. (2000). Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 439–450. 55

Akidau, T., Balikov, A., Bekiroğlu, K., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., and Whittle, S. (2013). Millwheel: fault-tolerant stream processing at internet scale. *Proc. VLDB Endow.*, 6(11):1033–1044. 99

Ali, M. I., Mileo, A., Parreira, J. X., Fischer, M., Kolozali, S., Farajidavar, N., Gao, F., Iggena, T., Pham, T.-L., Nechifor, C.-S., Puschmann, D., et al. (2016). Citypulse: Large scale data analytics framework for smart cities. 43

Alshehri, A., Spielman, J., Prasad, A., and Yue, C. (2022). Exploring the privacy concerns of bystanders in smart homes from the perspectives of both owners and bystanders. *Proceedings on Privacy Enhancing Technologies*. 75

Amir, A., Kolchinsky, I., and Schuster, A. (2022). Dlacep: A deep-learning based framework for approximate complex event processing. In *Proceedings of the 2022 International Conference on Management of Data*, pages 340–354. 98

Android Developers. Permissions on Android. https://developer.android.com/guide/topics/permissions/overview. Accessed: 2026-06-15. 50

Apple Inc. Requesting Access to Protected Resources. https://developer.apple.com/documentation/uikit/requesting-access-to-protected-resources. Accessed: 2026-06-15. 50

Apthorpe, N., Shvartzshnaider, Y., Mathur, A., Reisman, D., and Feamster, N. (2018). Discovering smart home internet of things privacy norms using contextual integrity. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 2(2):1–23. 7, 76

Arthur, R., Boulton, C. A., Shotton, H., and Williams, H. T. (2018). Social sensing of floods in the uk. *PloS one*, 13(1):e0189327. 41

Avnur, R. and Hellerstein, J. M. (2000). Eddies: Continuously adaptive query processing. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 261–272. 99

Ayyubi, H., Thomas, C., Chum, L., Lokesh, R., Chen, L., Niu, Y., Lin, X., Feng, X., Koo, J., Ray, S., et al. (2024). Beyond grounding: Extracting fine-grained event hierarchies across modalities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17664–17672. 43

Barth, A., Datta, A., Mitchell, J. C., and Nissenbaum, H. (2006). Privacy and contextual integrity: Framework and applications. In *2006 IEEE symposium on security and privacy (S&P'06)*, pages 15–pp. IEEE. 53, 76

Bartocci, E., Bortolussi, L., Loreti, M., and Nenzi, L. (2017). Monitoring mobile and spatially distributed cyber-physical systems. In *Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*, pages 146–155. 97

Brickell, J. and Shmatikov, V. (2008). The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 70–78. 55

Bucchi, M., Grez, A., Quintana, A., Riveros, C., and Vansummeren, S. (2021). Core: a complex event recognition engine. *arXiv preprint arXiv:2111.04635*. 97

California, S. o. PeMS Data Source | Caltrans. 31

Canel, C., Kim, T., Zhou, G., Li, C., Lim, H., Andersen, D. G., Kaminsky, M., and Dulloor, S. (2019). Scaling video analytics on constrained edge nodes. *Proceedings of Machine Learning and Systems*, 1:406–417. 98

Cartwright, M., Cramer, J., Mendez, A. E. M., Wang, Y., Wu, H.-H., Lostanlen, V., Fuentes, M., Dove, G., Mydlarz, C., Salamon, J., et al. (2020). Sonyc-ust-v2: An urban sound tagging dataset with spatiotemporal context. *arXiv preprint arXiv:2009.05188*. 41

Caven, P., Zhang, Z., Abbott, J., Ma, X., and Camp, L. (2024). Comparing the use and usefulness of four iot security labels. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–31. 76

Chakraborty, S., Shen, C., Raghavan, K. R., Shoukry, Y., Millar, M., and Srivastava, M. (2014). {ipShield}: A framework for enforcing {Context-Aware} privacy. In *11th USENIX symposium on networked systems design and implementation (NSDI 14)*, pages 143–156. 77

Charfi, I., Miteran, J., Dubois, J., Atri, M., and Tourki, R. (2013). Optimized spatio-temporal descriptors for real-time fall detection: comparison of support vector machine and adaboost-based classification. *Journal of Electronic Imaging*, 22(4):041106–041106. 69

Chawla, S., Zheng, Y., and Hu, J. (2012). Inferring the root cause in road traffic anomalies. In *2012 IEEE 12th International Conference on Data Mining*, pages 141–150. IEEE. 41

Chen, C., Zhou, D., Ye, Y., Li, T. J.-j., and Yao, Y. (2025). Clear: Towards contextual llm-empowered privacy policy analysis and risk generation for large language model applications. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*, pages 277–297. 78

Chi, H., Zeng, Q., Du, X., and Luo, L. (2019). Pfirewall: Semantics-aware customizable data flow control for home automation systems. *arXiv preprint arXiv:1910.07987*. 77

Chow, F. K., Kosović, B., and Chan, S. (2008). Source inversion for contaminant plume dispersion in urban environments using building-resolving simulations. *Journal of applied meteorology and climatology*, 47(6):1553–1572. 41

Cobb, C., Bhagavatula, S., Garrett, K. A., Hoffman, A., Rao, V., and Bauer, L. (2021). "i would have to evaluate their objections": Privacy tensions between smart home device owners and incidental users. *Proceedings on Privacy Enhancing Technologies*. 75

Colnago, J., Feng, Y., Palanivel, T., Pearman, S., Ung, M., Acquisti, A., Cranor, L. F., and Sadeh, N. (2020). Informing the design of a personalized privacy assistant for the internet of things. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13. 76

Das, A., Degeling, M., Smullen, D., and Sadeh, N. (2018). Personalized privacy assistants for the internet of things: Providing users with notice and choice. *IEEE Pervasive Computing*, 17(3):35–46. 50, 76

Davies, N., Taft, N., Satyanarayanan, M., Clinch, S., and Amos, B. (2016). Privacy mediators: Helping iot cross the chasm. In *Proceedings of the 17th international workshop on mobile computing systems and applications*, pages 39–44. 77

Demers, A., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., and White, W. Cayuga: A General Purpose Event Monitoring System. 97

Demers, A. J., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., White, W. M., et al. (2007). Cayuga: A general purpose event monitoring system. In *Cidr*, volume 7, pages 412–422. 42

Dempster, A. P. (2008). Upper and lower probabilities induced by a multivalued mapping. In *Classic works of the Dempster-Shafer theory of belief functions*, pages 57–72. Springer. 42

Emami-Naeini, P., Agarwal, Y., Cranor, L. F., and Hibshi, H. (2020). Ask the experts: What should be on an iot privacy and security label? In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 447–464. IEEE. 50, 76

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., and Balakrishnan, H. (2008). The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. 41

European Parliament and Council of the European Union (2009). Directive 2009/136/EC amending Directive 2002/58/EC. `https://eur-lex.europa.eu/eli/dir/2009/136/oj/eng`. Amendment relevant to cookie consent under Article 5(3); accessed 2026-06-15. 50

Federal Emergency Management Agency (2017a). National Incident Management System. Technical report, U.S. Department of Homeland Security. Defines "incident" as an occurrence, natural or human-caused, that necessitates a response to protect life or property. 16

Federal Emergency Management Agency (2017b). National Incident Management System. Technical report, U.S. Department of Homeland Security. Defines incident complex as two or more individual incidents in the same general area assigned to a single Incident Commander or Unified Command. 18, 46

Federal Emergency Management Agency (2018). Incident Complexity and Type. States that incident complexity is assessed on a scale from Type 5, least complex, to Type 1, most complex. 17, 46

Federal Emergency Management Agency (2021). NIMS Incident Complexity Guide: Planning, Preparedness and Training. Technical report, U.S. Department of Homeland Security. Provides guidance for assessing incident complexity and using incident complexity types. 17, 46

Feng, Y., Yao, Y., and Sadeh, N. (2021). A design space for privacy choices: Towards meaningful privacy control in the internet of things. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–16. 76

Fernandes, E., Paupore, J., Rahmati, A., Simionato, D., Conti, M., and Prakash, A. (2016). {FlowFence}: Practical data protection for emerging {IoT} application frameworks. In *25th USENIX security symposium (USENIX Security 16)*, pages 531–548. 77

Fortmann, T., Bar-Shalom, Y., and Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, 8(3):173–184. 42

Foster, P., Sigtia, S., Krstulovic, S., Barker, J., and Plumbley, M. D. (2015). Chimehome: A dataset for sound source recognition in a domestic environment. In *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5. IEEE. 69

Fremont, D. J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A. L., and Seshia, S. A. (2019). Scenic: a language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 63–78. 9

Gao, G., Dong, Y., Wang, R., and Zhou, X. (2024). Edgevision: Towards collaborative video analytics on distributed edges for performance maximization. *IEEE Transactions on Multimedia*, 26:9083–9094. 98

Geeng, C. and Roesner, F. (2019). Who's in control? interactions in multi-user smart homes. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–13. 76

Guo, X., Peng, M., Hao, X., Zou, X., Wang, Q., Ruan, S., and Liang, Y. (2026). Agentsense: Llms empower generalizable and explainable web-based participatory urban sensing. In *Proceedings of the ACM Web Conference 2026*, pages 5439–5450. 44

Gyllstrom, D., Wu, E., Chae, H.-J., Diao, Y., Stahlberg, P., and Anderson, G. (2006). SASE: Complex Event Processing over Streams. arXiv:cs/0612128. 97

Haghighi, I., Jones, A., Kong, Z., Bartocci, E., Gros, R., and Belta, C. (2015). Spatel: a novel spatial-temporal logic and its applications to networked systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 189–198. 43

Hall, D. L. and Llinas, J. (2002). An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23. 42

Han, C., Chang, C., Srivastava, S., Lu, Y., and Lo, E. (2025a). Scalable complex event processing on video streams. *Proceedings of the ACM on Management of Data*, 3(3):1–29. 98

Han, L., Dong, G., Ouyang, X., Kaplan, L., Cerutti, F., and Srivastava, M. (2025b). Toward foundation models for online complex event detection in cps-iot: A case study. In *Proceedings of the 2nd International Workshop on Foundation Models for Cyber-Physical Systems & Internet of Things*, pages 1–6. 98

Han, L. and Srivastava, M. B. (2024). An empirical evaluation of neural and neuro-symbolic approaches to real-time multimodal complex event detection. *arXiv preprint arXiv:2402.11403*. 98

He, W., Reitinger, N., Almogbil, A., Chiang, Y.-S., Pierson, T. J., and Kotz, D. (2024). Contextualizing interpersonal data sharing in smart homes. 76

Hobbs, J. R. and Pan, F. (2004). An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing*, 3(1):66–85. 5

Huang, L., Cassidy, T., Feng, X., Ji, H., Voss, C., Han, J., and Sil, A. (2016). Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 258–268. 43

Huot, F., Hu, R. L., Goyal, N., Sankar, T., Ihme, M., and Chen, Y.-F. (2022). Next day wildfire spread: A machine learning dataset to predict wildfire spreading from remote-sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13. 41

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. (2024). Gpt-4o system card. *arXiv preprint arXiv:2410.21276*. 70

Jana, S., Narayanan, A., and Shmatikov, V. (2013). A scanner darkly: Protecting user privacy from perceptual applications. In *2013 IEEE symposium on security and privacy*, pages 349–363. IEEE. 77

Jia, Y. J., Chen, Q. A., Wang, S., Rahmati, A., Fernandes, E., Mao, Z. M., and Prakash, A. (2017). Contexiot: Towards providing contextual integrity to appified iot platforms. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, volume 2, pages 2–2. San Diego. 77

Jiang, A. H., Wong, D. L.-K., Canel, C., Tang, L., Misra, I., Kaminsky, M., Kozuch, M. A., Pillai, P., Andersen, D. G., and Ganger, G. R. (2018a). Mainstream: Dynamic {Stem-Sharing} for {Multi-Tenant} video processing. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 29–42. 98

Jiang, J., Ananthanarayanan, G., Bodik, P., Sen, S., and Stoica, I. (2018b). Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 conference of the ACM special interest group on data communication*, pages 253–266. 98

Jiang, Y., Chao, Q., Chen, Y., Li, X., Liu, S., and Cong, G. (2024). Urbanllm: Autonomous urban activity planning and management with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1810–1825. 43

Jin, H., Liu, G., Hwang, D., Kumar, S., Agarwal, Y., and Hong, J. I. (2022). Peekaboo: A hub-based approach to enable transparency in data processing within smart homes. In *2022 IEEE symposium on security and privacy (SP)*, pages 303–320. IEEE. 77

Jocher, G. (2020). YOLOv5 by Ultralytics. 9

Kang, D., Bailis, P., and Zaharia, M. (2018). Blazeit: optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv preprint arXiv:1805.01046*. 98

Kang, D., Emmons, J., Abuzaid, F., Bailis, P., and Zaharia, M. (2017). Noscope: optimizing neural network queries over video at scale. *arXiv preprint arXiv:1703.02529*. 98

Keats, A., Yee, E., and Lien, F.-S. (2007). Bayesian inference for source determination with applications to a complex urban environment. *Atmospheric environment*, 41(3):465–479. 41

Khan, M., Sarhaddi, F., Zuniga, A., Flores, H., Tarkoma, S., and Nurmi, P. (2026). Governance at the edge: Agent-driven privacy mediation for mobile and iot data. In *Proceedings of the 27th International Workshop on Mobile Computing Systems and Applications*, pages 85–90. 78

Khan, Z., Balch, T., and Dellaert, F. (2005). Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE transactions on pattern analysis and machine intelligence*, 27(11):1805–1819. 42

Khandelwal, R., Linden, T., Harkous, H., and Fawaz, K. (2021). {PriSEC}: A privacy settings enforcement controller. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 465–482. 78

Kim, W., Lee, S., Chang, Y., Lee, T., Hwang, I., and Song, J. (2021). Hivemind: social control-and-use of iot towards democratization of public spaces. In *Proceedings of the 19th annual international conference on mobile systems, applications, and services*, pages 467–482. 76

Kulldorff, M., Heffernan, R., Hartman, J., Assunçao, R., and Mostashari, F. (2005). A space–time permutation scan statistic for disease outbreak detection. *PLoS medicine*, 2(3):e59. 41

Langheinrich, M. (2002). A privacy awareness system for ubiquitous computing environments. In *UbiComp 2002: Ubiquitous Computing: 4th International Conference Göteborg, Sweden, September 29–October 1, 2002 Proceedings 4*, pages 237–245. Springer. 76

Li, M., Li, S., Wang, Z., Huang, L., Cho, K., Ji, H., Han, J., and Voss, C. (2021). The future is not one-dimensional: Complex event schema induction by graph modeling for event prediction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5203–5215. 43

Li, S., Zhao, R., Li, M., Ji, H., Callison-Burch, C., and Han, J. (2023). Open-domain hierarchical event schema induction by incremental prompting and verification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5677–5697. 43

Li, T. and Li, N. (2009). On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–526. 55

Li, T., Reiman, K., Agarwal, Y., Cranor, L. F., and Hong, J. I. (2022). Understanding challenges for developers to create accurate privacy nutrition labels. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–24. 76

Li, W., Sun, L., Guan, Z., Zhou, X., and Sap, M. (2025a). 1-2-3 check: Enhancing contextual privacy in llm via multi-agent reasoning. In *Proceedings of the The First Workshop on LLM Security (LLMSEC)*, pages 115–128. 78

Li, Y., Padmanabhan, A., Zhao, P., Wang, Y., Xu, G. H., and Netravali, R. (2020). Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 359–376. 98

Li, Y., Qiu, W., Shezan, F. H., Cai, K., van Dam, M., Austin, L., Lie, D., and Tian, Y. (2025b). Breaking the illusion: Automated reasoning of gdpr consent violations. *arXiv preprint arXiv:2512.22789*. 78

Li, Z., Xia, L., Tang, J., Xu, Y., Shi, L., Xia, L., Yin, D., and Huang, C. (2024). Urbangpt: Spatio-temporal large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 5351–5362. 43

Liao, H., Li, Y., Wang, C., Guan, Y., Tam, K., Tian, C., Li, L., Xu, C., and Li, Z. (2024). When, where, and what? a benchmark for accident anticipation and localization with large language models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 8–17. 44

Liu, B., Andersen, M. S., Schaub, F., Almuhimedi, H., Zhang, S. A., Sadeh, N., Agarwal, Y., and Acquisti, A. (2016). Follow my recommendations: A personalized privacy assistant for mobile app permissions. In *Twelfth symposium on usable privacy and security (SOUPS 2016)*, pages 27–41. 76

Liu, W., Luo, W., Lian, D., and Gao, S. (2018). Future frame prediction for anomaly detection–a new baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6536–6545. 41

Liu, X., Ghosh, P., Ulutan, O., Manjunath, B., Chan, K., and Govindan, R. (2019). Caesar: cross-camera complex activity recognition. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 232–244. 43, 98

Ma, M., Bartocci, E., Lifland, E., Stankovic, J., and Feng, L. (2020a). Sastl: Spatial aggregation signal temporal logic for runtime monitoring in smart cities. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, pages 51–62. IEEE. 43

Ma, M., Bartocci, E., Lifland, E., Stankovic, J., and Feng, L. (2020b). SaSTL: Spatial Aggregation Signal Temporal Logic for Runtime Monitoring in Smart Cities. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, pages 51–62. ISSN: 2642-9500. 97

Mahler, R. P. (2003). Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic systems*, 39(4):1152–1178. 42

Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., and Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM special interest group on data communication*, pages 270–288. 99

Marky, K., Gerber, N., Pelzer, M. G., Khamis, M., and Mühlhäuser, M. (2022). "you offer privacy like you offer tea": Investigating mechanisms for improving guest privacy in iot-equipped households. *Proceedings on Privacy Enhancing Technologies*. 75

Maroti, M., Simon, G., Ledeczi, A., and Sztipanovits, J. (2004). Shooter localization in urban terrain. *Computer*, 37(8):60–61. 41

Mehrotra, S., Kobsa, A., Venkatasubramanian, N., and Rajagopalan, S. R. (2016). Tippers: A privacy cognizant iot environment. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6. IEEE. 77

Mohan, P., Padmanabhan, V. N., and Ramjee, R. (2008). Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. 41

Murray, D. G., McSherry, F., Isaacs, R., Isard, M., Barham, P., and Abadi, M. (2013). Naiad: a timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 439–455. 99

Nissenbaum, H. (2020). *Privacy in context: Technology, policy, and the integrity of social life*. Stanford University Press. 7, 57

Ntouni, O., Banelas, D., and Giatrakos, N. (2025). Neuroflinkcep: Neurosymbolic complex event recognition optimized across iot platforms. *Proceedings of the VLDB Endowment*, 18(12):5355–5358. 98

Oh, S., Russell, S., and Sastry, S. (2004). Markov chain monte carlo data association for general multiple-target tracking problems. In *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, volume 1, pages 735–742. IEEE. 42

Okutani, I. and Stephanedes, Y. J. (1984). Dynamic prediction of traffic volume through kalman filtering theory. *Transportation Research Part B: Methodological*, 18(1):1–11. 42

Pan, J., Yuan, Z., Zhang, X., and Chen, D. (2022). Youhome system and dataset: Making your home know you better. In *2022 IEEE International Symposium on Smart Electronic Systems (iSES)*, pages 414–420. IEEE. 69

Pawar, K. and Attar, V. (2022). Deep learning based detection and localization of road accidents from traffic surveillance videos. *ICT Express*, 8(3):379–387. 41

Payne, H. J. and Tignor, S. C. (1978). Freeway incident-detection algorithms based on decision trees with states. *Transportation Research Record*, (682). 40

Rahat, T. A., Long, M., and Tian, Y. (2022). Is your policy compliant? a deep learning-based empirical study of privacy policies' compliance with gdpr. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, pages 89–102. 78

Reid, D. (2003). An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control*, 24(6):843–854. 42

Ritchie, S. G. and Cheu, R. L. (1993). Simulation of freeway incident detection using artificial neural networks. *Transportation Research Part C: Emerging Technologies*, 1(3):203–217. 40

Rodriguez, D., Yang, I., Del Alamo, J. M., and Sadeh, N. (2024). Large language models: a new approach for privacy policy analysis at scale. *arXiv preprint arXiv:2405.20900*. 78

Romero, F., Li, Q., Yadwadkar, N. J., and Kozyrakis, C. (2021). {INFaaS}: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 397–411. 98

Saide, P. E., Peterson, D. A., da Silva, A., Anderson, B., Ziemba, L. D., Diskin, G., Sachse, G., Hair, J., Butler, C., Fenn, M., et al. (2015). Revealing important nocturnal and day-to-day variations in fire smoke emissions through a multiplatform inversion. *Geophysical Research Letters*, 42(9):3609–3618. 41

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. 41

Salamon, J., Jacoby, C., and Bello, J. P. (2014). A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044. 41

Shah, A. P., Lamare, J.-B., Nguyen-Anh, T., and Hauptmann, A. (2018). Cadp: A novel dataset for cctv traffic camera based accident analysis. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–9. IEEE. 41

Shen, H., Chen, L., Jin, Y., Zhao, L., Kong, B., Philipose, M., Krishnamurthy, A., and Sundaram, R. (2019). Nexus: A gpu cluster engine for accelerating dnn-based video analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 322–337. 98

Sikder, A. K., Babun, L., Celik, Z. B., Aksu, H., McDaniel, P., Kirda, E., and Uluagac, A. S. (2022). Who's controlling my device? multi-user multi-device-aware access control system for shared smart home environment. *ACM Transactions on Internet of Things*, 3(4):1–39. 76

Simon, G., Maróti, M., Lédeczi, Á., Balogh, G., Kusy, B., Nádas, A., Pap, G., Sallai, J., and Frampton, K. (2004). Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12. 41

Sudo, A., Kashiyama, T., Yabe, T., Kanasugi, H., Song, X., Higuchi, T., Nakano, S., Saito, M., and Sekimoto, Y. (2016). Particle filter for real-time human mobility prediction following unprecedented disaster. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–10. 42

Sultani, W., Chen, C., and Shah, M. (2018). Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6479–6488. 41

Team, C. Carla. 9

Thakkar, P. K., He, S., Xu, S., Huang, D. Y., and Yao, Y. (2022). "it would probably turn into a social faux-pas": Users' and bystanders' preferences of privacy awareness

mechanisms in smart homes. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–13. 75

Tian, Y., Zhang, N., Lin, Y.-H., Wang, X., Ur, B., Guo, X., and Tague, P. (2017). {SmartAuth}:{User-Centered} authorization for the internet of things. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 361–378. 77

Topcuoglu, H., Hariri, S., and Wu, M.-Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3):260–274. 99

Vilamala, M. R., Xing, T., Taylor, H., Garcia, L., Srivastava, M., Kaplan, L., Preece, A., Kimmig, A., and Cerutti, F. (2023). Deepprobcep: A neuro-symbolic approach for complex event processing in adversarial settings. *Expert Systems with Applications*, 215:119376. 43, 98

Vo, B.-N. and Ma, W.-K. (2006). The gaussian mixture probability hypothesis density filter. *IEEE Transactions on signal processing*, 54(11):4091–4104. 42

Vo, B.-N., Singh, S., and Doucet, A. (2005). Sequential monte carlo methods for multitarget filtering with random finite sets. *IEEE Transactions on Aerospace and electronic systems*, 41(4):1224–1245. 42

Vo, B.-T., Vo, B.-N., and Cantoni, A. (2007). Analytic implementations of the cardinalized probability hypothesis density filter. *IEEE transactions on signal processing*, 55(7):3553–3567. 42

Wang, B., Garcia, L. A., and Srivastava, M. (2024). Privacyoracle: Configuring sensor privacy firewalls with large language models in smart built environments. In *2024 IEEE Security and Privacy Workshops (SPW)*, pages 239–245. IEEE. 78

Wang, L., Khan, U., Near, J., Pang, Q., Subramanian, J., Somani, N., Gao, P., Low, A., and Song, D. (2022). {PrivGuard}: Privacy regulation compliance made easier. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3753–3770. 77

White, J., Thompson, C., Turner, H., Dougherty, B., and Schmidt, D. C. (2011). Wreckwatch: Automatic traffic accident detection and notification with smartphones. *Mobile Networks and Applications*, 16(3):285–303. 41

Wong, Y., Chen, S., Mau, S., Sanderson, C., and Lovell, B. C. (2011). Patch-based probabilistic image quality assessment for face selection and improved video-based face recognition. In *IEEE Biometrics Workshop, Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 81–88. IEEE. 69

X Corp. (2026). X. `https://x.com`. Accessed: 2026-05-24. 31

Xiao, C., Zhou, J., Xiao, Y., Huang, J., and Xiong, H. (2024). Refound: Crafting a foundation model for urban region understanding upon language and visual foundations. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3527–3538. 44

Xing, T., Garcia, L., Vilamala, M. R., Cerutti, F., Kaplan, L., Preece, A., and Srivastava, M. (2020). Neuroplex: learning to detect complex events in sensor networks through knowledge injection. In *Proceedings of the 18th conference on embedded networked sensor systems*, pages 489–502. 43, 98

Xing, T., Vilamala, M. R., Garcia, L., Cerutti, F., Kaplan, L., Preece, A., and Srivastava, M. (2019). Deepcep: Deep complex event processing using distributed multimodal information. In *2019 IEEE international conference on smart computing (SMARTCOMP)*, pages 87–92. IEEE. 98

Xue, J., Leung, Y., and Fung, T. (2017). A bayesian data fusion approach to spatiotemporal fusion of remotely sensed images. *Remote Sensing*, 9(12):1310. 42

Yadav, P. and Curry, E. (2019). Vidcep: Complex event processing framework to detect spatiotemporal patterns in video streams. In *2019 IEEE International conference on big data (big data)*, pages 2513–2522. IEEE. 98

Yadav, P., Salwala, D., Das, D. P., and Curry, E. (2020). Knowledge graph driven approach to represent video streams for spatiotemporal event pattern matching in complex event processing. *International Journal of Semantic Computing*, 14(03):423–455. 98

Yao, Y., Wang, X., Xu, M., Pu, Z., Atkins, E., and Crandall, D. (2020). When, where, and what? a new dataset for anomaly detection in driving videos. *arXiv preprint arXiv:2004.03044*. 41

Yi, J., Acer, U. G., Kawsar, F., and Min, C. (2024). Argus: Enabling cross-camera collaboration for video analytics on distributed smart cameras. *IEEE Transactions on Mobile Computing*, 24(1):117–134. 98

Yu, L., Du, B., Hu, X., Sun, L., Han, L., and Lv, W. (2021). Deep spatio-temporal graph convolutional network for traffic accident prediction. *Neurocomputing*, 423:135–147. 41

Yuan, Z., Zhou, X., and Yang, T. (2018). Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 984–992. 41

Zeng, E., Mare, S., and Roesner, F. (2017). End user security and privacy concerns with smart homes. In *thirteenth symposium on usable privacy and security (SOUPS 2017)*, pages 65–80. 75

Zhang, C., Liu, L., Lei, D., Yuan, Q., Zhuang, H., Hanratty, T., and Han, J. (2017a). Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 595–604. 41

Zhang, C., Zhou, G., Yuan, Q., Zhuang, H., Zheng, Y., Kaplan, L., Wang, S., and Han, J. (2016). Geoburst: Real-time local event detection in geo-tagged tweet streams. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 513–522. 41

Zhang, H., Agarwal, Y., and Fredrikson, M. (2022a). Teo: Ephemeral ownership for iot devices to provide granular data control. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, pages 302–315. 76

Zhang, H., Ananthanarayanan, G., Bodik, P., Philipose, M., Bahl, P., and Freedman, M. J. (2017b). Live video analytics at scale with approximation and {Delay-Tolerance}. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 377–392. 98

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2022b). Bytetrack: Multi-object tracking by associating every detection box. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 1–21. Springer. 9

Zhang, Z., Cao, Y., Ye, C., Ma, Y., Liao, L., and Chua, T.-S. (2024). Analyzing temporal complex events with large language models? a benchmark towards temporal, long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1588–1606. 43

Zhou, H., Goel, M., and Agarwal, Y. (2024). Bring privacy to the table: Interactive negotiation for privacy settings of shared sensing devices. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–22. 51, 77

Zhou, Z., Wang, Y., Xie, X., Chen, L., and Liu, H. (2020). Riskoracle: A minute-level citywide traffic accident forecasting framework. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1258–1265. 41