SARUS: PRIVACY-PRESERVING MULTI-VENDOR PERCEPTION FUSION VIA HOMOMORPHIC ENCRYPTION

Munawar Hasan[1,2], **Apostol Vassilev**[1]

[1]National Institute of Standards and Technology, USA
[2]Michigan Technological University, USA

{munawar.hasan, apostol.vassilev}@nist.gov
munawarh@mtu.edu

ABSTRACT

Cooperative perception enables autonomous vehicles (AVs) to improve situational awareness by aggregating detection outputs from multiple agents and sensing platforms, often via a shared fusion service in multi-vendor deployments. However, sharing such outputs at inference time exposes proprietary model behavior and sensitive environmental information, creating significant privacy and security concerns. In this paper, we present Sarus, a privacy-preserving framework for multi-vendor perception fusion via homomorphic encryption (HE), enabling aggregation without revealing individual vendor outputs. Each vendor encodes detections as compact Gaussian moment vectors over a shared spatial lattice and transmits encrypted payloads to a fusion server, which aggregates them directly in the encrypted domain. The fused result is then decrypted and reconstructed into final detections through class-wise bin merging.

We analyze the computational complexity, showing linear scaling for vendor payload construction and $O(BV)$ server-side fusion with the number of occupied bins B and vendors V, while postprocessing scales as $O(B + \sum_{c \in \mathcal{C}} B_c^2)$, where $\mathcal{C}$ denotes the set of object classes and B_c is the number of occupied bins for class c. Experiments demonstrate linear scaling in practice with only a bounded constant-factor overhead from HE, with decryption dominating postprocessing cost. Experiments on the KITTI dataset using camera (YOLOv8 [1]) and LiDAR (PointPillars, PV-RCNN) detectors show that Sarus improves scene-level coverage by effectively aggregating complementary detections, particularly in distance-dependent regimes where individual modalities degrade. These results indicate that privacy-preserving multi-vendor perception fusion is feasible for real-time deployment when statistical compression and spatial sparsity are jointly exploited. The demonstration code and dataset for this project is open source, available on Github[2].

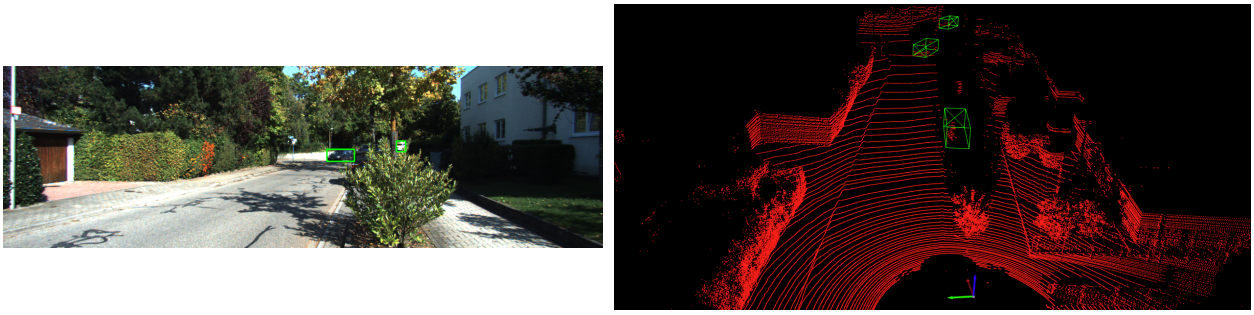
***K*eywords** Cooperative Perception · Privacy-Preserving Inference, Multi-Vendor Fusion · Autonomous Vehicles · Homomorphic Encryption · Secure Aggregation

1 Introduction

Cooperative perception enables multiple agents, such as autonomous vehicles, roadside infrastructure, and perception service providers, to share complementary observations of a common environment. By aggregating information from diverse viewpoints and sensing modalities, cooperative perception improves robustness under occlusion, extends sensing range beyond line-of-sight, and enhances reliability in challenging conditions such as adverse weather or

[1]Certain equipment, instruments, software, or materials, commercial or noncommercial, are identified in this paper to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials, equipment, or software identified are necessarily the best available for the purpose.

[2]https://github.com/mhasan08/sarus



(a) Camera-only perception: YOLOv8 detects two cars.

(b) LiDAR-only perception: PV-RCNN. detects three cars.

Figure 1: Complementarity of heterogeneous perception outputs on KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) [9]: The camera detector captures visually salient vehicles, while the LiDAR detector recovers additional 3D structure, including a nearby vehicle missed in the image due to partial occlusion. This example motivates privacy-preserving cooperative fusion: multiple agents or vendors may collectively improve scene coverage, but direct sharing of raw sensor data or detailed proprietary outputs can expose sensitive information.

dense traffic [1, 2, 3, 4, 5]. Beyond spatial diversity, cooperative perception also leverages *temporal and contextual diversity*, where observations collected across time or from different agents contribute to a more stable and consistent understanding of dynamic scenes. This enables improved tracking of objects, early detection of hazards, and more reliable estimation of scene structure compared to isolated perception systems. Cooperative perception transforms perception from a purely local task into a *distributed sensing and aggregation problem*, where multiple heterogeneous sources jointly contribute to a unified representation of the environment.

In practical deployments, cooperative perception is often realized through a centralized or edge-based *fusion server* [6, 7], which collects perception outputs from multiple sources and aggregates them into a unified scene representation. Each participant processes its local sensor data independently and transmits a compact representation—such as detected objects, bounding boxes, or intermediate features—to the fusion server. The server then combines these inputs to produce a consistent and enriched understanding of the environment, which can be redistributed to participating agents or used for downstream decision-making. Multi-modal fusion further reinforces the role of the fusion server as an aggregation point for heterogeneous data representations. Each vendor or agent may process distinct sensor modalities or employ modality-specific models, producing outputs that differ in structure, scale, and uncertainty characteristics. The fusion process must therefore reconcile these heterogeneous representations into a consistent spatial interpretation of the scene. By integrating these distributed observations, the fusion server enables a more complete and accurate representation of the scene than any single agent could achieve independently. Real-world deployments, such as the University of Michigan Smart Intersection Project, demonstrate this paradigm by aggregating multi-sensor observations to construct real-time intersection-level scene understanding [8]. This paradigm shifts perception from isolated sensing to distributed scene understanding.

Importantly, cooperative perception operates at the *inference level*, where agents exchange the outputs of their local perception pipelines rather than raw sensor data or model parameters. Each participant independently processes its sensor inputs and produces detection results—such as object locations, classes, and confidence scores—which are then shared for aggregation. This design avoids the need for centralized training or access to raw data, making it well-suited for real-time deployment in dynamic environments. In Figure 1, YOLOv8 [10] detects two vehicles in the camera image, while PV-RCNN [11] identifies three using LiDAR (Light Detection and Ranging) data. The missed detection by YOLOv8 corresponds to the closest vehicle, which is occluded by vegetation. This example highlights how inference-level cooperative perception can leverage complementary modalities to recover missed objects. However, operating at the inference level introduces unique challenges. Unlike training-time collaboration frameworks such as federated learning, where model updates can be aggregated without exposing individual predictions, cooperative perception requires sharing fine-grained outputs that directly reflect model behavior on specific scenes. These outputs inherently encode both environmental information and model-specific characteristics, making them highly sensitive. As a result, protecting inference-time data becomes critical for preserving privacy, confidentiality, and competitive integrity in multi-vendor settings.

Despite its advantages, cooperative perception introduces a fundamental challenge between *utility and privacy*. Vendor detections encode not only information about the environment, but also implicit knowledge about proprietary models, training data, and system behavior. Sharing raw perception outputs in cooperative perception systems exposes multiple security and privacy risks. In particular, plaintext detection sharing enables the following attack vectors:

1. **Model Extraction:** Repeated access to detection outputs allows an adversarial aggregation service or participating vendor to infer properties of proprietary perception models. By observing confidence scores, bounding box behavior, and response patterns across diverse inputs, an attacker can approximate model decision boundaries or reconstruct surrogate models [12, 13, 14]. This compromises vendor intellectual property. Further, such compromise can also reveal vendor-specific design choices and performance characteristics, enabling competitors to reverse-engineer or benchmark proprietary systems.
2. **Data Leakage at Inference:** Detection outputs may reveal sensitive information about the observed environment, including the presence, location, and behavior of objects or individuals. In infrastructure-assisted or multi-agent settings, sharing such outputs in plaintext can inadvertently expose private or regulated data [12, 15].
3. **Side-Channel Leakage:** Auxiliary information such as timing, output sparsity, and detection patterns can leak additional information beyond the explicit content of detections. For example, variations in detection latency or confidence distributions may reveal scene complexity, sensor characteristics, internal model behavior, or model format [12, 16] further amplifying privacy risks.
4. **Cross-Vendor Inference.** In multi-vendor settings, access to multiple vendors' outputs for the same scene enables comparative analysis, allowing adversaries to infer relative strengths, weaknesses, or biases of individual models. Such cross-analysis can expose competitive information and facilitate targeted attacks against specific vendors.

This challenge is further amplified in *multi-vendor* settings, where participating entities may be competitors or operate under distinct trust and regulatory constraints. Existing cooperative perception systems largely assume access to raw or intermediate detection outputs in plaintext, making them incompatible with privacy-preserving deployment scenarios. Consequently, there is a pressing need for mechanisms that enable *secure aggregation of perception outputs* without revealing vendor-specific information.

In this paper, we present Sarus[3], a privacy-preserving framework for multi-vendor perception fusion based on homomorphic encryption (HE). Instead of sharing raw detections, each vendor encodes its observations into compact *moment-based statistical representations*. In this representation, detections are summarized through aggregate quantities such as confidence mass, confidence-weighted object center, and spatial spread, which capture object location, uncertainty, and confidence without exposing the raw detection outputs. These representations are encrypted using CKKS (Cheon–Kim–Kim–Song) [17] and transmitted to a fusion server, which performs aggregation directly in the encrypted domain without accessing plaintext data. The fused result is then decrypted by an authorized party and used to reconstruct the final detections.

A key aspect of our design is the use of a shared spatial lattice that discretizes the scene into bins, enabling scalable aggregation without explicit cross-vendor matching. By combining moment-based encoding with spatial binning, Sarus achieves linear scaling in the number of occupied bins while maintaining robustness to minor spatial misalignment. Importantly, the use of homomorphic encryption introduces only a bounded constant-factor overhead, preserving practical feasibility. We summarize our contributions below:

- We introduce a homomorphically encrypted moment-based fusion framework that enables aggregation of multi-vendor perception outputs without revealing raw detections.
- We propose a spatial binning mechanism that ensures scalable fusion with complexity $O(B \cdot V)$, where B is the number of occupied bins and V the number of vendors.
- We provide a comprehensive performance evaluation covering vendor payload generation, server-side homomorphic fusion, and postprocessing.
- We demonstrate that homomorphic encryption incurs only bounded overhead, with linear scaling in practice and decryption dominating vendor-side cost.
- We show empirically on KITTI [9] that Sarus effectively recovers complementary detections across camera and LiDAR modalities.

[3] Sarus draws inspiration from the Sarus crane—a species native to India and Southeast Asia—known for its coordinated movement and broad environmental awareness, mirroring the collaborative perception paradigm underlying our framework.

Our results show that privacy-preserving multi-vendor perception fusion is feasible in real-time settings when statistical compression and spatial sparsity are jointly exploited, enabling secure and scalable cooperative perception in heterogeneous environments. We clarify that Sarus does not handle authentication, identity management, schema validation, or proof-carrying compliance for cooperative perception. Instead, it assumes that submitted payloads have passed an admission check before encrypted fusion. This admission check includes both agent legitimacy and adherence to a shared syntax and public specification for the submitted perception output. Such an admission layer may be built from proof-carrying data [18], vehicular message security and credential mechanisms such as IEEE 1609.2 [19], and succinct zero-knowledge compliance mechanisms such as Hermes Seal [20].

2 Related Work

Cooperative perception enables multiple agents to share observations and improve scene understanding beyond the capabilities of individual sensors. Prior work has explored vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) collaboration for object detection and tracking, demonstrating improved robustness under occlusion and extended sensing range [1, 2, 3]. These approaches typically rely on exchanging raw sensor data [4, 21, 22], intermediate features [23, 24, 25], or detection outputs [26], and assume a trusted setting in which such information can be shared in plaintext. Infrastructure-assisted systems further extend this paradigm by aggregating observations from roadside units and vehicles to construct a unified scene representation, often via a centralized fusion module. Modern perception systems increasingly incorporate multiple sensing modalities, such as cameras, LiDAR, and radar, to improve robustness and accuracy [27]. Multi-modal fusion methods [28, 29, 30, 31, 32, 33, 34] combine complementary information from heterogeneous sensors to enhance detection performance under challenging conditions [4, 35]. In cooperative settings, this heterogeneity is further amplified, as different agents may employ distinct sensor configurations and model architectures. Existing fusion strategies typically operate on feature-level or detection-level representations and require direct access to underlying data, making them difficult to deploy in privacy-sensitive multi-vendor environments.

Privacy-preserving machine learning techniques, such as federated learning (FL) [36, 37, 38, 39], enable collaborative model training without sharing raw data. In FL, participants exchange model updates rather than inference outputs, reducing exposure of local datasets. However, FL may not be suitable for cooperative perception since, FL cannot operate at *inference time*, requiring aggregation of instance-level predictions rather than model parameters [40, 41]. As a result, existing FL-based approaches do not directly address the privacy risks associated with sharing detection outputs across vendors. Secure aggregation protocols and homomorphic encryption have been widely studied for enabling computation over encrypted data [42, 43, 44]. HE schemes, such as CKKS [17], support approximate arithmetic on encrypted vectors and have been applied in domains such as secure inference and privacy-preserving analytics [45, 46]. Prior work has explored encrypted aggregation in distributed learning and statistics; however, these approaches typically focus on scalar or low-dimensional data and do not address the challenges of structured perception outputs, such as spatial alignment, object-level aggregation, and multi-modal heterogeneity.

Existing cooperative perception systems typically assume plaintext sharing of detection outputs, while privacy-preserving learning approaches focus on training or intermediate representations rather than inference-time collaboration. While prior work has explored cooperative perception and privacy-preserving computation independently, a scalable framework for privacy-preserving multi-vendor perception fusion at inference time remains largely unexplored, particularly for structured detection outputs. Sarus bridges this gap by introducing an encrypted moment-based representation and a spatial binning strategy that enable scalable homomorphic aggregation of structured perception outputs without revealing vendor-specific information.

3 Data Representation, System, and Threat Model

We start by presenting notations used in this paper. Table 1 summarizes the abbreviations and mathematical notations.

Table 1: Symbols and Mathematical Notations

$\mathcal{V}$	Vendors, $\mathcal{V} \in \{v_1 \dots v_V\}$ such that $\lvert\mathcal{V}\rvert = V$
$\mathsf{Enc}_{\mathsf{HE}}(\cdot)$, $\mathsf{Dec}_{\mathsf{HE}}(\cdot)$	Homomorphic encryption and decryption
$\mathcal{P}_v$	Encrypted payload from vendor $v \in \mathcal{V}$
$\mathcal{C}$	Set of object classes
$\mathcal{F}_{\text{common}}$	Common spatial reference frame.
$\mathbf{b}$	Bounding box with coordinates x_1, y_1, x_2, y_2.
(w, h)	Width and height of bounding box.
μ, σ^2	Mean and variance.
α	Vendor trust weight.

3.1 Data Representation

Definition 1 (Common Spatial Frame)**.** *Let $\mathcal{F}_{common} \subseteq \mathbb{R}^d$ denote a shared spatial reference frame in which detections from all vendors are expressed. Let $\mathcal{O}$ denote a physical object in the environment, and for each vendor $v \in \mathcal{V}$, let Φ_v denote the observation operator that maps $\mathcal{O}$ to the vendor's sensor observation $\mathcal{I}_v$, i.e.,*

$$\Phi_v : \mathcal{O} \to \mathcal{I}_v.$$

Further, let $\mathcal{T}_v$ denote a transformation that maps observations to the common spatial frame, then we have:

$$\mathcal{T}_v : \mathcal{I}_v \to \mathcal{F}_{common}.$$

We assume that for any object $\mathcal{O}$ observed by multiple vendors, the composed mappings $\mathcal{T}_v \circ \Phi_v(\mathcal{O})$ are spatially consistent, i.e., they lie within a bounded neighborhood in $\mathcal{F}_{common}$.

Φ_v captures sensing and perception effects (e.g., viewpoint, modality, and model behavior), while $\mathcal{T}_v$ accounts for geometric alignment through calibration and localization.

Definition 2 (Spatial Consistency Assumption)**.** *For any physical object $\mathcal{O}$ observed by vendors v and v', the corresponding coordinates (x_v, y_v) and $(x_{v'}, y_{v'})$ in $\mathcal{F}_{common}$ satisfy:*

$$\|(x_v, y_v) - (x_{v'}, y_{v'})\| \leq \epsilon,$$

for some bounded alignment error $\epsilon > 0$ determined by calibration and localization accuracy.

The common spatial frame $\mathcal{F}_{\text{common}}$ can take several forms depending on the deployment setting. In many autonomous driving systems, detections are projected into a bird's-eye-view (BEV) representation aligned with the ground plane, enabling consistent spatial reasoning across agents. Alternatively, an ego-centric coordinate frame centered at a reference vehicle may be used, where all detections are expressed relative to the vehicle's pose. In infrastructure-assisted settings, detections may be aligned to a global or map-based coordinate system using GPS and localization pipelines. These transformations are standard in cooperative perception systems and rely on well-established calibration and pose estimation techniques.

3.2 System Model

We consider a multi-vendor cooperative perception setting consisting of a set of vendors such that $|\mathcal{V}| = V$ and a centralized fusion server. Each vendor $v \in \mathcal{V}$ operates an independent perception pipeline over its local sensor observations and participates in collaborative inference through secure aggregation.

Vendors: Each vendor v observes the environment through local sensors, producing observations $\mathcal{I}_v$ (e.g., images or point clouds). Using its proprietary perception model, the vendor extracts detection outputs and encodes them into a compact moment-based representation over a shared spatial frame $\mathcal{F}_{\text{common}}$. The resulting payload is encrypted using a homomorphic encryption scheme $\mathsf{Enc}_{\mathsf{HE}}(\cdot)$ before transmission.

Fusion Server: The fusion server receives encrypted payloads from the participating vendors and performs aggregation directly in the encrypted domain. Its role is limited to computing aggregated statistics over ciphertexts; it does not require access to plaintext detections, model parameters, or intermediate vendor representations.

Output Reconstruction: After aggregation, the encrypted fused result is returned to an authorized party (e.g., a participating vendor or a trusted client), which decrypts the payload using $\mathsf{Dec}_{\mathsf{HE}}(\cdot)$ and reconstructs final detections through spatial and statistical postprocessing.

Communication Model: Communication occurs in two stages: (i) vendors transmit encrypted payloads to the fusion server, and (ii) the fusion server returns an aggregated encrypted result. Note, no plaintext perception outputs are exchanged between vendors or revealed to the server.

Assumptions: We assume that all vendors express detections in a shared spatial reference frame $\mathcal{F}_{\text{common}}$ (see Definition 1). Vendors do not share raw observations or model parameters, and encryption keys are not accessible to the server.

3.3 Threat Model

Building on the system model in Section 3.2, Our threat model focuses on protecting the confidentiality of vendor perception outputs at inference time.

Adversarial Fusion Server: *Honest-but-curious*—It correctly follows the prescribed aggregation protocol but may attempt to infer information about individual vendor inputs from the received data and intermediate computations. In particular, the server has access to all transmitted payloads and aggregation results, and may perform arbitrary offline analysis on observed data. The server does not possess decryption keys and cannot directly access plaintext representations.

Curious Vendors: *Mutually untrusted*— May attempt to infer information about other participants through the fusion process. However, vendors only observe their own inputs and the final fused output after decryption, and do not have access to intermediate per-vendor contributions.

Adversarial Capabilities: The adversary may — (i) observe all encrypted payloads transmitted to the fusion server, (ii) access aggregated ciphertexts and final fused outputs, and (iii) perform statistical or inference attacks on observed data. The adversary is not able to— (i) break the underlying homomorphic encryption scheme, (ii) access secret keys or decrypt intermediate ciphertexts, and (iii) tamper with the protocol execution (i.e., no active attacks).

Security Goals: Our objective is to ensure that no party learns any information about individual vendor detections beyond what is revealed by the final fused output. In particular, we aim to:

- Protect the confidentiality of vendor detection outputs, including object locations and confidence scores.
- Prevent inference of proprietary model behavior from shared data.
- Ensure that intermediate aggregation steps do not leak additional information.

Out of Scope: We assume that all admitted vendors express detections in a shared spatial reference frame $\mathcal{F}_{\text{common}}$ (see Definition 1) and follow the agreed payload schema and public fusion specification. Admission control, authentication, and compliance checking are treated as prerequisite mechanisms rather than functions provided by Sarus.

We do not consider active adversaries that deviate from the protocol, collusion between multiple parties, or side-channel attacks arising from implementation-specific leakage (e.g., timing or hardware effects). These extensions are left for future work. A formal treatment of inference-time privacy with explicit leakage bounds can be developed under standard semantic security assumptions of homomorphic encryption scheme; we defer a detailed treatment to future work.

Intuition: At a high level, Sarus ensures that the fusion server operates exclusively on encrypted representations of vendor data. Since all vendor payloads are encrypted under a semantically secure homomorphic encryption scheme, the server cannot access individual detections or intermediate statistics in plaintext. Note that we use the term semantic security in its standard cryptographic sense: ciphertexts should not reveal any efficiently computable information about the underlying plaintext, except what is already implied by public information [47, 48]. Aggregation is performed through homomorphic operations, and only the final fused result is revealed after decryption. Thus, the server's view is computationally indistinguishable from that of an observer with access only to encrypted data and the final output.

4 Sarus

The key challenge in privacy-preserving perception fusion is to aggregate object detections from multiple vendors without revealing proprietary model outputs or intermediate representations of those outputs. Sharing bounding boxes or prediction confidences of the perception stack in plaintext to vendors exposes sensitive perception information, therefore violating the trust-neutral collaboration model. To address this challenge, Sarus converts each detection into a

representation whose sufficient statistics can be aggregated using only linear operations. This design enables the fusion server to combine perception outputs directly in encrypted form using homomorphic encryption, while preserving the geometric information required to reconstruct the fused detections.

Specifically, each bounding box is represented as a Gaussian splat and encodes its weighted spatial statistics as a moment vector. In this work, Gaussian splatting refers to representing each detection as a spatial Gaussian contribution over the common fusion grid, where the Gaussian center corresponds to the detected object location and the covariance/spread captures localization uncertainty. This allows detections from multiple vendors to be accumulated as smooth statistical evidence rather than as discrete bounding boxes. The moment vectors are accumulated within spatial bins and homomorphically aggregated across vendors. After decryption, the fused moments are inverted to recover the parameters of the fused Gaussian representation, which is then converted back into a bounding box estimate. This representation allows collaborative perception fusion while maintaining strict privacy guarantees for vendor detections.

4.1 Detection Representation via Gaussian Splats

Bounding boxes represent discrete geometric primitives whose fusion typically requires non-linear operations such as intersection tests, non-maximum suppression, or heuristic box merging strategies. Such operations are incompatible with secure aggregation under homomorphic encryption, which efficiently supports only linear computations. To address this limitation, Sarus converts each bounding box detection into a continuous spatial representation using a *Gaussian splat*. The Gaussian representation models the detected object by its spatial center and an associated uncertainty that reflects the spatial extent of the bounding box. This representation provides two important advantages: (1) Gaussian splats allow detections from multiple vendors to be combined in a statistically meaningful manner through aggregation of their spatial statistics, and (2) the sufficient statistics of a Gaussian distribution can be expressed as linear moment sums, enabling secure fusion through homomorphic addition without exposing the underlying detections. Using these two properties Sarus transforms perception fusion into a linear aggregation problem, which can be performed directly on encrypted data while preserving the information required to reconstruct fused object detections.

A detection produced by an object detector is represented by an axis-aligned bounding box $\mathbf{b} = [x_1, y_1, x_2, y_2]$. Geometrically, the bounding box defines a spatial region as follows:

$$\mathbf{b} = \{(x, y) \mid x_1 \leq x \leq x_2,\ y_1 \leq y \leq y_2\}.$$

Let the width and height of the bounding box be $w = x_2 - x_1$, and $h = y_2 - y_1$ respectively and its geometric center be $c_x = \frac{x_1+x_2}{2}$ and $c_y = \frac{y_1+y_2}{2}$ respectively. Since the exact object center within the detected box is uncertain, we model spatial uncertainty by assuming that the true object center is uniformly distributed within the bounding box. This assumption should be interpreted as a local non-informative prior over the admissible support region, not as a uniform distribution over the entire scene. In particular, the support is constrained by the detected bounding box, the image plane, and the physically feasible region associated with the candidate object; therefore, physically invalid locations are excluded [49, 50, 51]. Let X and Y denote the horizontal and vertical object coordinates, respectively. Then under uniform uncertainty assumption, we have:

$$X \sim \mathrm{Uniform}(x_1, x_2), \qquad Y \sim \mathrm{Uniform}(y_1, y_2).$$

The corresponding probability density functions are given by:

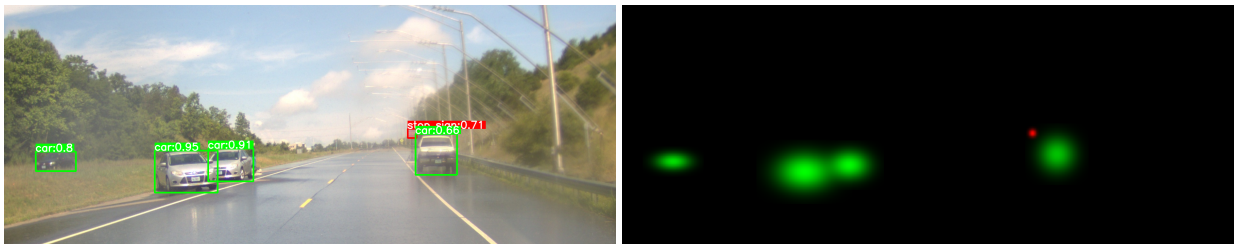
$$f_X(x) = \begin{cases} \frac{1}{w}, & x_1 \leq x \leq x_2 \\ 0, & \text{otherwise} \end{cases} \qquad f_Y(y) = \begin{cases} \frac{1}{h}, & y_1 \leq y \leq y_2 \\ 0, & \text{otherwise.} \end{cases}$$

Assuming independence between the two spatial coordinates, the joint density inside the bounding box becomes

$$f_{\mathbf{b}}(x, y) = \begin{cases} \frac{1}{wh}, & (x, y) \in \mathbf{b} \\ 0, & \text{otherwise.} \end{cases}$$

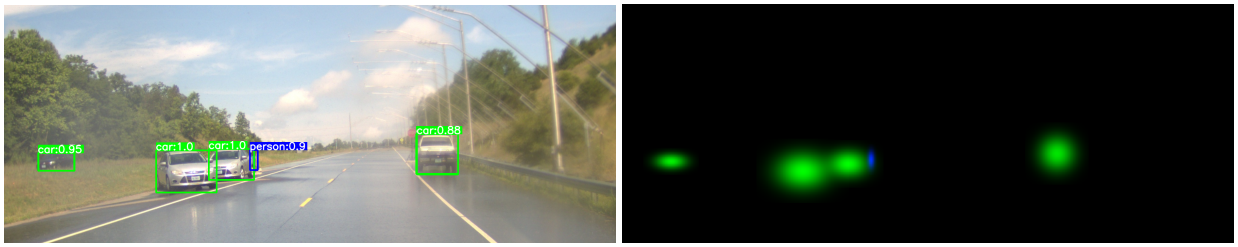
The mean and variance of the induced spatial distribution are therefore: $\mu_x = \mathbb{E}[X] = c_x$, $\mu_y = \mathbb{E}[Y] = c_y$ and $\sigma_x^2 = \mathrm{Var}(X) = \frac{w^2}{12}$ and $\sigma_y^2 = \mathrm{Var}(Y) = \frac{h^2}{12}$. While the uniform model captures the spatial extent of the bounding box, Sarus represents detections using a Gaussian splat whose parameters preserve the mean of the distribution while allowing a tunable spatial spread. This spacial spread is controlled by a scaling parameter $\kappa > 0$, hence:

$$\sigma_x^2 = \left(\kappa \frac{w}{2}\right)^2, \qquad \sigma_y^2 = \left(\kappa \frac{h}{2}\right)^2. \tag{1}$$



(a) Stop sign detected $59.46m$, pedestrian undetected $47.07m$. (b) Gaussian Splat for the detection.

Figure 2: RT-DETR detection.



(a) Stop sign undetected $59.46m$, pedestrian detected $47.07m$. (b) Gaussian Splat for the detection.

Figure 3: DETR101 detection.

Note, when $\kappa = \frac{1}{\sqrt{3}}$, equation (1) matches the variance induced by the uniform spatial uncertainty of the bounding box. Under this representation, each detection is approximated by the Gaussian distribution:

$$\mathcal{N}\big((\mu_x, \mu_y), \mathrm{diag}(\sigma_x^2, \sigma_y^2)\big), \tag{2}$$

which we refer to as a *Gaussian splat*: a continuous spatial model of the detection preserving the geometric information implied by the bounding box and used for encrypted moment aggregation. Figure 2 and Figure 3 show detection in the image space (Figure 2a and Figure 3a) and their respective gaussian splats (Figure 2b and Figure 3b).

Moment-Based Detection Encoding: Let a detection $\mathbf{b}$ with class confidence p be modeled by the Gaussian distribution of equation (2). Let α be the trust weight associated with some vendor $v \in \mathcal{V}$, such that the effective contribution weight of the detection is defined as: $w = \alpha p$. Then Sarus encodes each detection using a vector of weighted spatial moments:

$$\mathbf{m} = w \cdot \left[1,\ \mu_x,\ \mu_x^2,\ \sigma_x^2,\ \mu_y,\ \mu_y^2,\ \sigma_y^2\right]. \tag{3}$$

Each component the vector in equation (3) represents a sufficient statistic of the spatial distribution and is linear with respect to aggregation across detections. Consequently, moment vectors from multiple detections can be accumulated through simple summation.

Aggregated Moments. Let $\mathcal{H}$ denote an object hypothesis, defined as a set of detections that are hypothesized to correspond to the same underlying object instance. Formally, $\mathcal{H}$ is a subset of detections whose spatial coordinates and class predictions are mutually consistent under the fusion model. Let $\mathbf{m}_k$ denote the moment vector produced by the k^{th} detection in this set, where $k \in \mathcal{H}$. The aggregated statistics of the hypothesis are obtained by:

$$\mathbf{s} = \sum_{k \in \mathcal{H}} \mathbf{m}_k.$$

Expanding the vector yields

$$\mathbf{s} = \left[S_w,\ S_{w\mu_x},\ S_{w\mu_x^2},\ S_{w\sigma_x^2},\ S_{w\mu_y},\ S_{w\mu_y^2},\ S_{w\sigma_y^2}\right]. \tag{4}$$

These aggregated quantities correspond to the sufficient statistics of the Gaussian representation associated with the hypothesis $\mathcal{H}$.

Lemma 1 (Linearity and Sufficiency)**.** *Let each contributing detection $k \in K$ produce a moment vector*

$$\mathbf{m}_k = \big[w_k,\ w_k\mu_{x,k},\ w_k\mu_{x,k}^2,\ w_k\sigma_{x,k}^2,\ w_k\mu_{y,k},\ w_k\mu_{y,k}^2,\ w_k\sigma_{y,k}^2\big],$$

where $w_k = \alpha_{v(k)}p_k$ is the weighted confidence of the detection for vendor $v \in \mathcal{V}$, such that these moment vectors are aggregated through summation to obtain

$$\mathbf{s} = \sum_k \mathbf{m}_k = [S_w,\ S_{w\mu_x},\ S_{w\mu_x^2},\ S_{w\sigma_x^2},\ S_{w\mu_y},\ S_{w\mu_y^2},\ S_{w\sigma_y^2}].$$

Then:

1. ***(Linearity)** The aggregated moment vector $\mathbf{s}$ can be computed using only component-wise additions. Consequently, if the vectors $\mathbf{m}_k$ are encrypted, homomorphic addition yields the exact plaintext sums after decryption.*
2. ***(Sufficiency)** The fused Gaussian parameters $(\mu_x, \mu_y, \sigma_x^2, \sigma_y^2)$ can be recovered from $\mathbf{s}$ as*

$$\mu_x = \frac{S_{w\mu_x}}{S_w}, \qquad \mu_y = \frac{S_{w\mu_y}}{S_w},$$

$$\sigma_x^2 = \frac{S_{w\sigma_x^2} + S_{w\mu_x^2}}{S_w} - \mu_x^2, \qquad \sigma_y^2 = \frac{S_{w\sigma_y^2} + S_{w\mu_y^2}}{S_w} - \mu_y^2.$$

Hence, the 7 aggregated moments are sufficient to reconstruct the fused Gaussian representation of the detections.

The design choice of local aggregation of moment contributions at the vendor before encryption provides the following advantages:

1. **Reduced encrypted payload size:** Without aggregation, each detection may contribute up to four encrypted moment vectors (due to bilinear assignment, refer section 4.1.1), resulting in $O(N)$ encrypted messages for N detections. With aggregation, the payload scales with the number of object hypotheses $|\mathcal{H}|$, typically $|\mathcal{H}| \ll N$ (see Figure 4). For example, if a vendor produces $N = 200$ detections, the naive approach may require up to 800 encrypted vectors. In practice many detections fall within the same spatial hypothesis region and therefore contribute to a common aggregated moment vector, considerably reducing the number of encrypted vectors (e.g., $|\mathcal{H}| \approx 30$ hypotheses.)
2. **Lower encryption cost:** Encryption is performed once per hypothesis rather than once per detection contribution, reducing the number of cryptographic operations.
3. **Improved network scalability:** Across multiple vendors the reduction becomes substantial. For instance, with 50 vendors each producing 200 detections, the naive approach would transmit roughly $50 \times 800 \approx 40000$ ciphertexts, whereas aggregation reduces this to approximately $50 \times 30 \approx 1500$ ciphertexts.
4. **Efficient homomorphic fusion:** Because moment aggregation is linear (Lemma 1), encrypted hypothesis statistics can be combined across vendors using homomorphic addition.

We now introduce the spatial binning and soft assignment procedure, which defines the sets of detections that contribute to each object hypothesis $\mathcal{H}$. Detections are associated with nearby grid regions based on the location of their Gaussian centers, and their moment vectors are distributed to neighboring bins using bilinear weights. The resulting moment contributions are then aggregated according to Eq. 4 leveraging the linearity of the moment representation.

4.1.1 Spatial Binning and Soft Assignment

Direct aggregation of detections across the entire image would allow spatially distant objects to influence each other, potentially producing unstable fusion results. To ensure that only spatially consistent detections are combined, the image is partitioned into a class-specific spatial grid and aggregation is performed independently within each grid cell. Let S_c and s_c be spacial anchor and stride for a given class c, then Algorithm 1 creates such a grid. Since grid creation can be precomputed for a given image size and class, we safely ignore Algorithm 1 in complexity calculation in section 4.5.1.

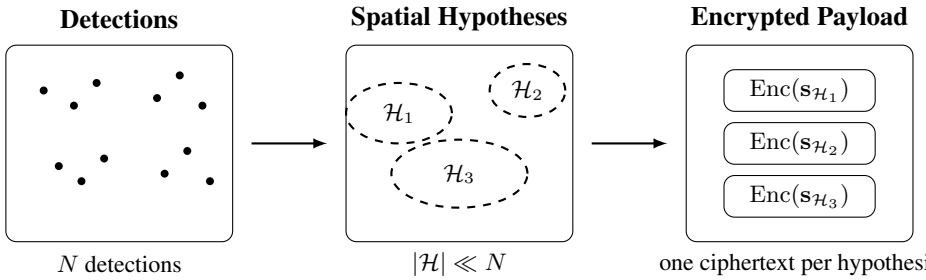


Figure 4: Illustration of vendor-side moment aggregation in Sarus. A large set of detections is first grouped into a smaller number of spatial object hypotheses, and only one encrypted aggregated moment vector is transmitted per hypothesis.

Algorithm 1 Class-Specific Spatial Grid Generation

Require: List of classes: C, List of spatial anchors for each class: S, List of strides for each class: s, Image width: W, Image height: H.

$G \leftarrow \phi$ ▷ *Initialize grid for each class c in C*
for $c = 1$ to $|C|$ **do**
  $G[c] \leftarrow \phi$ ▷ *Initialize G for current class c*
  $n_x \leftarrow \lceil \frac{W - \frac{S[c]}{2}}{s[c]} \rceil$, $n_y \leftarrow \lceil \frac{H - \frac{S[c]}{2}}{s[c]} \rceil$ ▷ *Number of bins for class $c \in C$*
  for $j = 1$ to n_y **do**
    $y = \frac{S[c]}{2} + (j-1) \cdot s[c]$
    for $i = 1$ to n_x **do**
      $x = \frac{S[c]}{2} + (i-1) \cdot s[c]$
      $G[c]$.append$((x, y))$ ▷ *Append grid center for each bin*
    end for
  end for
end for
Return G

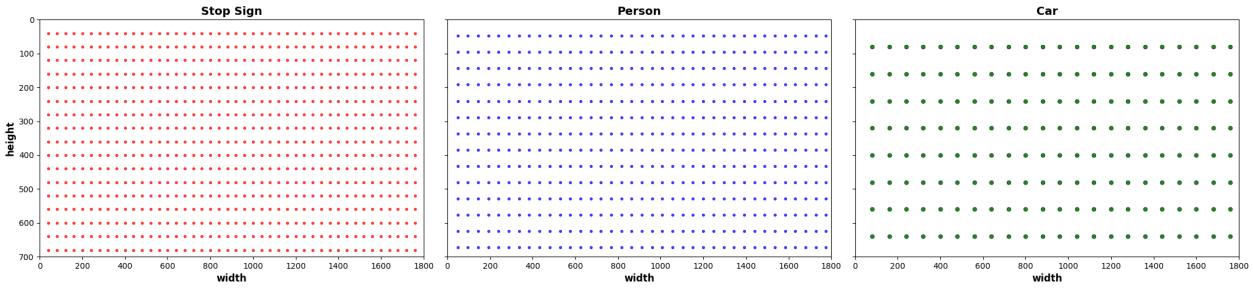


Figure 5: Bin centers using spatial anchors for image with width: 1800 and height 700. Stop sign: 748 bins ($S = 80, s = 40$), Person: 518 bins ($S = 96, s = 48$) and Car: 176 bins ($S = 160, s = 80$).

Soft Bin Assignment: Let (μ_x, μ_y) denote the center of the Gaussian splat derived from a bounding box. Then instead of assigning the detection to a single bin, its contribution is distributed to the four neighboring bins using bilinear interpolation. Let (x_i, y_j) denote the center of bin (i, j). Then fractional offsets of the detection center within the local grid cell are: $t_x = \frac{\mu_x - x_i}{s_c}$ and $t_y = \frac{\mu_y - y_j}{s_c}$ and the four neighboring bins are: $\{(i, j), (i+1, j), (i, j+1), (i+1, j+1)\}$ with respective bilinear interpolation weights given by:

$$\omega_{00} = (1 - t_x)(1 - t_y),\ \omega_{10} = t_x(1 - t_y),\ \omega_{01} = (1 - t_x)t_y,\ \omega_{11} = t_x t_y,$$

where, $\omega_{00} + \omega_{10} + \omega_{01} + \omega_{11} = 1$, such that the total contribution of the detection is preserved.

Moment Accumulation. For each bin receiving a contribution, the weighted moment vector $\mathbf{m}_k$ (introduced earlier) is accumulated as: $\mathbf{s}_{c,(i,j)} \leftarrow \mathbf{s}_{c,(i,j)} + \omega_{k \to (i,j)}\, \mathbf{m}_k$, where (i, j) is the recieving bin. The same update is applied

for the neighboring bins i.e., $(i+1, j)$, $(i, j+1)$, and $(i+1, j+1)$ using their corresponding weights. Since each detection contributes to at most four bins, the update cost per detection is constant. Consequently, the construction of per-class-bin moment accumulators scales linearly with the number of detections produced by a vendor and preserving smooth spatial transitions across bin boundaries, which improves the stability of the fused detections.

4.2 Vendor Payload Construction

Let vendor $v \in \mathcal{V}$ produce a set of detections $\mathcal{D}_v$ such that: $\mathcal{D}_v = \{d_{v,1}, d_{v,2}, \ldots, d_{v,N_v}\}$, where N_v is the total number of detections produced by v. Each detection $d_{v,k}$, where $k \in \{1, \ldots, N_v\}$, is first converted into its Gaussian splat representation as described in Section 4.1, yielding parameters $(\mu_{x,v,k}, \mu_{y,v,k}, \sigma_{x,v,k}, \sigma_{y,v,k})$ for the k^{th} detection. Let $p_{v,k}$ be the detection confidence for the k^{th} detection and the vendor trust weight α_v, then the detection weight is defined as: $w_{v,k} = \alpha_v p_{v,k}$. The detection is then encoded as the moment vector as follows:

$$\mathbf{m}_{v,k} = w_{v,k} \left[1,\ \mu_{x,v,k},\ \mu^2_{x,v,k},\ \sigma^2_{x,v,k},\ \mu_{y,v,k},\ \mu^2_{y,v,k},\ \sigma^2_{y,v,k}\right].$$

Using the soft assignment procedure described previously, detection $d_{v,k}$ contributes to at most four neighboring bins with weights $\omega_{k\to(i,j)}$. The vendor accumulates these contributions locally:

$$\mathbf{s}_{v,c,(i,j)} \leftarrow \mathbf{s}_{v,c,(i,j)} + \omega_{k\to(i,j)}\, \mathbf{m}_{v,k}.$$

This accumulation is performed independently for each object class c and spatial bin (i, j). After processing all detections, vendor v obtains a set of per-bin moment accumulators as follows:

$$\mathbf{s}_{v,c,(i,j)} = \left[S_w,\ S_{w\mu_x},\ S_{w\mu_x^2},\ S_{w\sigma_x^2},\ S_{w\mu_y},\ S_{w\mu_y^2},\ S_{w\sigma_y^2}\right].$$

These aggregated statistics constitute sufficient information to reconstruct the fused Gaussian parameters after aggregation.

Payload Structure: After local aggregation, vendor v constructs a payload indexed by the class-bin key i.e., $\texttt{key} \leftarrow (c, (i, j))$. Let $\mathsf{Enc}_{\mathsf{HE}}(\cdot)$ denote homomorphic encryption scheme which supports linear operation in the encrypted domain (Sarus uses CKKS[17]-based homomorphic encryption scheme that provides additive homomorphism). To enable privacy-preserving multi-vendor fusion, each moment vector is encrypted using $\mathsf{Enc}_{\mathsf{HE}}(\cdot)$. For each occupied key, the vendor stores three quantities:

$$\begin{aligned} \mathbf{c}_{v,\texttt{key}} &= \mathsf{Enc}_{\mathsf{HE}}(\mathbf{s}_{v,\texttt{key}}), \\ M_{v,\texttt{key}} &= \sum_{k \in \mathcal{H}_{v,\texttt{key}}} \omega_{k\to\texttt{key}}\, (\alpha_v p_{v,\texttt{key}}), \\ C_{v,\texttt{key}} &= \sum_{k \in \mathcal{H}_{v,\texttt{key}}} \omega_{k\to\texttt{key}}\, \alpha_v, \end{aligned} \tag{5}$$

where, $\mathcal{H}_{v,\texttt{key}}$ denotes the set of detections from v whose soft assignment contributes to key, and $\omega_{k\to\texttt{key}}$ is the corresponding bilinear assignment weight. The quantity $M_{v,\texttt{key}}$ represents the aggregated weighted confidence mass for key and $C_{v,\texttt{key}}$ represents the aggregated weight count. The payload is the set defined as:

$$\mathcal{P}_v = \left\{ \left(\mathbf{c}_{v,\texttt{key}}, M_{v,\texttt{key}}, C_{v,\texttt{key}}\right) \;\middle|\; \texttt{key} = (c, (i, j)) \right\},$$

for all occupied class-bin keys. Algorithm 2 presents the complete pseudocode that a vendor $v \in \mathcal{V}$ uses to generate their respective payloads.

4.3 Encrypted Multi-Vendor Fusion by Server

Let $\mathcal{V}$ denote the set of participating vendors such that each vendor $v \in \mathcal{V}$ transmits a payload, $\mathcal{P}_v = \left\{(\mathbf{c}_{v,\texttt{key}}, M_{v,\texttt{key}}, C_{v,\texttt{key}})\right\}$ where $\texttt{key} = (c, (i, j))$ denotes the class-bin key, $\mathbf{c}_{v,\texttt{key}}$ is the encrypted aggregated moment vector, and $M_{v,\kappa}$ and $C_{v,\texttt{key}}$ are the associated statistics defined above. The fusion server first computes the union of all keys contributed by vendors i.e.,

$$\mathcal{K} = \bigcup_{v \in \mathcal{V}} \text{keys}(\mathcal{P}_v),$$

where, each $\texttt{key} \in \mathcal{K}$ corresponds to a spatial hypothesis.

Algorithm 2 Vendor Payload Construction

Require: Detection: $\mathcal{D}$, `image_width` : W, `image_height` : H
Ensure: Each `detection` $\in \mathcal{D}$:
`detection`: {`probability`, `class_id`, `bbox` : $[x_1, y_1, x_2, y_2]$, `spacial_anchor` : S, `stride` : s}

for each detection $d \in \mathcal{D}$ **do**
  $p \leftarrow d$.get(`probability`), `cls` $\leftarrow d$.get(`class_id`), $\mathbf{b} \leftarrow d$.get(`bbox`)
  $(c_x, c_y, \sigma_x^2, \sigma_y^2) \leftarrow$ **BoxToGaussianParams**$(\mathbf{b}, \kappa)$
  $\mathbf{m} \leftarrow [\alpha p, \alpha c_x, \alpha c_x^2, \alpha\sigma_x^2, \alpha c_y, \alpha c_y^2, \alpha\sigma_y^2]$
  $\mathcal{B} \leftarrow$ **SoftAssignBins**$(c_x, c_y, d$.get(`spacial_anchor`)$, d$.get(`stride`)$, W, H)$
  for each $(\texttt{bin}, \omega) \in \mathcal{B}$ **do**
    `key` $\leftarrow$ (`cls`, `bin_id`)
    `accumulator`[`key`] $\leftarrow$ `accumulator`[`key`] $+ \omega \cdot \mathbf{m}$
    `mass_by_key`[`key`] $\leftarrow$ `mass_by_key`[`key`] $+ \omega \cdot (\alpha p)$
    `count_by_key`[`key`] $\leftarrow$ `count_by_key`[`key`] $+ \omega \cdot \alpha$
  end for
end for
`cipher_by_key` $\leftarrow \emptyset$
for each `key` $\in$ keys(`accumulator`) **do**
  `ciphertext_bytes` $\leftarrow \mathrm{Enc}_{\mathrm{HE}}$(`accumulator`[`key`]) ▷ Homomorphic encryption
  `cipher_by_key`[`key`] $\leftarrow$ `ciphertext_bytes`
end for
Return (`cipher_by_key`, `mass_by_key`, `count_by_key`)

BoxToGaussianParams$(\mathbf{b}, \kappa)$**:**
$x_1, y_1, x_2, y_2 \leftarrow \mathbf{b}$
$w \leftarrow (x_2 - x_1)$, $h \leftarrow (y_2 - y_1)$
$c_x \leftarrow \frac{x_1+x_2}{2}$, $c_y \leftarrow \frac{y_1+y_2}{2}$
$\sigma_x^2 \leftarrow \left(\kappa \cdot \frac{w}{2}\right)^2$, $\sigma_y^2 \leftarrow \left(\kappa \cdot \frac{h}{2}\right)^2$
Return $(c_x, c_y, \sigma_x^2, \sigma_y^2)$

SoftAssignBins$(\mu_x, \mu_y, S, s, W, H)$**:**
$(n_x, n_y, \_) \leftarrow$ **SpatialGridGenerator**(W, H, S, s)
$i \leftarrow \lfloor(\mu_x - S/2)/s\rfloor + 1$,$j \leftarrow \lfloor(\mu_y - S/2)/s\rfloor + 1$
$i \leftarrow \max(1, \min(i, n_x - 1))$, $\quad j \leftarrow \max(1, \min(j, n_y - 1))$
$x_i \leftarrow S/2 + (i-1)s$, $\quad y_j \leftarrow S/2 + (j-1)s$
$t_x \leftarrow (\mu_x - x_i)/s$, $\quad t_y \leftarrow (\mu_y - y_j)/s$
$t_x \leftarrow \min(1, \max(0, t_x))$, $\quad t_y \leftarrow \min(1, \max(0, t_y))$
$\mathcal{B} \leftarrow \emptyset$
$b_{00} \leftarrow (i, j)$, $\quad \omega_{00} \leftarrow (1 - t_x)(1 - t_y)$
$b_{10} \leftarrow (i + 1, j)$, $\quad \omega_{10} \leftarrow t_x(1 - t_y)$
$b_{01} \leftarrow (i, j + 1)$, $\quad \omega_{01} \leftarrow (1 - t_x)t_y$
$b_{11} \leftarrow (i + 1, j + 1)$, $\quad \omega_{11} \leftarrow t_x t_y$
for each $(\texttt{bin\_id}, \omega) \in \{(b_{00}, \omega_{00}), (b_{10}, \omega_{10}), (b_{01}, \omega_{01}), (b_{11}, \omega_{11})\}$ **do**
  if $\omega > 0$ **then**
    add $(\texttt{bin\_id}, \omega)$ to $\mathcal{B}$
  end if
end for
$Z \leftarrow \sum_{(\texttt{bin\_id},\omega)\in\mathcal{B}} \omega$
if $Z > 0$ **then**
  for each $(\texttt{bin\_id}, \omega) \in \mathcal{B}$ **do**
    $\omega \leftarrow \omega/Z$
  end for
end if
return $\mathcal{B}$

Homomorphic Moment Fusion: For each key, the server aggregates encrypted moment vectors across vendors using the additive homomorphism of the encryption scheme:

$$\mathbf{c}_{\texttt{key}}^{\text{fused}} = \bigoplus_{v \in \mathcal{V}_{\texttt{key}}} \mathbf{c}_{v,\texttt{key}} \tag{6}$$

where $\mathcal{V}_{\texttt{key}} \subseteq \mathcal{V}$ denotes the set of vendors that contributed to key, and $\oplus$ denotes homomorphic ciphertext addition. Since the moment representation is linear (refer Lemma 1), hence, the operation in equation (6) produces an encryption of the fused moment vector:

$$\mathbf{c}_{\texttt{key}}^{\text{fused}} = \mathsf{Enc}\left(\sum_{v \in \mathcal{V}_{\texttt{key}}} \mathbf{s}_{v,\texttt{key}} \right).$$

The accompanying statistics are aggregated as follows:

$$M_{\texttt{key}}^{\text{fused}} = \sum_{v \in \mathcal{V}_{\texttt{key}}} M_{v,\texttt{key}}, \qquad C_{\texttt{key}}^{\text{fused}} = \sum_{v \in \mathcal{V}_{\texttt{key}}} C_{v,\texttt{key}}.$$

Fused Payload: The resulting fused representation maintained by the server is given by the following set:

$$\mathcal{P}^{\text{fused}} = \left\{ (\mathbf{c}_{\texttt{key}}^{\text{fused}}, M_{\texttt{key}}^{\text{fused}}, C_{\texttt{key}}^{\text{fused}}) \;\middle|\; \texttt{key} = (c,(i,j)) \right\},$$

which contains encrypted fused moment vectors and the associated statistics for every occupied key. Algorithm 3 presents the pseudocode that the server uses to create fused payload.

4.4 Fused Payload to Detection Reconstruction

Vendor $v \in \mathcal{V}$ operates on the fused payload shared by the server: $\mathcal{P}^{\text{fused}} = \left\{ (\mathbf{c}_{\texttt{key}}^{\text{fused}}, M_{\texttt{key}}^{\text{fused}}, C_{\texttt{key}}^{\text{fused}}) \right\}$, where $\texttt{key} = (c,(i,j))$ denotes the class-bin key, $\mathbf{c}_{\texttt{key}}^{\text{fused}}$ is the encrypted fused moment vector, and $M_{\texttt{key}}^{\text{fused}}$ and $C_{\texttt{key}}^{\text{fused}}$ are the associated statistics obtained during the fusion stage.

Moment Decryption and Parameter Recovery: For each key, v decrypts the fused moment vector:

$$\mathbf{s}_{\texttt{key}} = \mathsf{Dec}_{\mathsf{HE}}\left(\mathbf{c}_{\texttt{key}}^{\text{fused}}\right).$$

Let

$$\mathbf{s}_{\texttt{key}} = \left[S_w, S_{w\mu_x}, S_{w\mu_x^2}, S_{w\sigma_x^2}, S_{w\mu_y}, S_{w\mu_y^2}, S_{w\sigma_y^2} \right].$$

Using the moment inversion relations derived in Lemma 1, the Gaussian parameters of the fused detection are recovered as:

$$\mu_x = \frac{S_{w\mu_x}}{S_w}, \qquad \mu_y = \frac{S_{w\mu_y}}{S_w},$$

$$\sigma_x^2 = \frac{S_{w\sigma_x^2} + S_{w\mu_x^2}}{S_w} - \mu_x^2, \qquad \sigma_y^2 = \frac{S_{w\sigma_y^2} + S_{w\mu_y^2}}{S_w} - \mu_y^2.$$

The corresponding standard deviations are

$$\sigma_x = \sqrt{\max(\sigma_x^2, \epsilon)}, \qquad \sigma_y = \sqrt{\max(\sigma_y^2, \epsilon)}.$$

where $\epsilon > 0$ is a small numerical constant used to ensure non-negative variance and maintain numerical stability during moment inversion.

Bounding Box Reconstruction: The recovered Gaussian parameters define a spatial extent for the fused detection. This Gaussian representation is converted into a bounding box $\mathbf{b} = [x_1, y_1, x_2, y_2]$, such that $x_1 = \mu_x - \lambda\sigma_x$, $y_1 = \mu_y - \lambda\sigma_y$, $x_2 = \mu_x + \lambda\sigma_x$ and $y_2 = \mu_y + \lambda\sigma_y$, where λ controls the spatial coverage of the reconstructed box.

Class-wise Spatial Hypothesis Grouping: Since soft assignment distributes moment contributions across neighboring bins, the same physical object may appear in multiple adjacent class-bin entries. To recover a consistent set of detections, entries are grouped by object class and spatial consistency is analyzed between neighboring bins. For each class c, let $\mathcal{E}_c = \left\{ (\texttt{key}, \mu_x, \mu_y, \sigma_x, \sigma_y, \mathbf{b}, \mathbf{s}_{\texttt{key}}) \right\}$ denote the set of reconstructed bin-level entries.

Algorithm 3 HE-Encrypted Multi-Vendor Fusion by Server

Require: Vendor Payloads $\mathcal{V} = \{p_1, \ldots, p_V\}$ where $|\mathcal{V}| = V$
Ensure: Each vendor payload $p_v, \forall v \in \mathcal{V}$:
`cipher_by_key`: $\{\texttt{key} : (\texttt{class\_id}, \texttt{bin\_id}), \texttt{ciphertext\_bytes}\}$ ▷ HE-encrypted $\mathbf{v}$
`mass_by_key`: $\{\texttt{key} : (\texttt{class\_id}, \texttt{bin\_id}), \texttt{float}\}$ ▷ Confidence mass
`count_by_key`: $\{\texttt{key} : (\texttt{class\_id}, \texttt{bin\_id}), \texttt{float}\}$ ▷ Detection count

`cipher_list` $\leftarrow \emptyset$, `mass_list` $\leftarrow \emptyset$, `count_list` $\leftarrow \emptyset$, `all_keys` $\leftarrow \emptyset$
`fused_cipher_by_key` $\leftarrow \emptyset$, `fused_mass_by_key` $\leftarrow \emptyset$, `fused_count_by_key` $\leftarrow \emptyset$
for $v \in \mathcal{V}$ **do**
    `cipher_list`.append(p_v[`cipher_by_key`])
    `mass_list`.append(p_v[`mass_by_key`]),
    `count_list`.append(p_v[`count_by_key`])
end for
for each `cipher` $\in$ `cipher_list` **do**
    `all_keys` $\leftarrow$ `all_keys` $\cup$ keys(`cipher`) ▷ Collect all unique (`class_id`, `bin_id`) keys across vendors payloads
end for
if `all_keys` $= \emptyset$ **then**
    raise error "No ciphertexts available"
end if
for each `key` $\in$ `all_keys` **do**
    `fused_cipher_by_key`[`key`] $\leftarrow \perp$, `fused_mass_by_key`[`key`] $\leftarrow 0$, `fused_count_by_key`[`key`] $\leftarrow 0$
    `accumulator` $\leftarrow$ `None`
    for each `cipher` $\in$ `cipher_list` **do**
        $c \leftarrow$ `cipher`.get(`key`)
        if $c \neq$ `None` **then**
            if `accumulator` $=$ `None` **then**
                `accumulator` $\leftarrow c$
            else
                `accumulator` $\leftarrow$ `accumulator` $+ c$ ▷ HE addition based on key (`class_id`, `bin_id`)
            end if
        end if
    end for
    `fused_cipher_by_key`[`key`] $\leftarrow$ `accumulator`
    for each `mass` $\in$ `mass_list` **do**
        `fused_mass_by_key`[`key`] $\leftarrow$ `fused_mass_by_key`[`key`] $+$ `mass`.get(`key`)
    end for
    for each `count` $\in$ `count_list` **do**
        `fused_count_by_key`[`key`] $\leftarrow$ `fused_count_by_key`[`key`] $+$ `count`.get(`key`)
    end for
end for
Return (`fused_cipher_by_key`, `fused_mass_by_key`, `fused_count_by_key`)

Spatial Consistency Graph: For each class c, we construct an undirected graph given by: $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c^{\text{graph}})$, where the vertex set $\mathcal{V}_c = \mathcal{E}_c$ consists of the reconstructed bin-level entries for class c (i.e., each vertex $\texttt{key} \in \mathcal{V}_c$ represents a local hypothesis $\mathcal{H}_{\texttt{key}}$ associated with a class-bin entry). An edge is introduced between two vertices $\texttt{key}_a, \texttt{key}_b \in \mathcal{V}_c$ if their corresponding bins are spatially adjacent and their associated Gaussian statistics satisfy spatial consistency constraints. Specifically, an edge $(\texttt{key}_a, \texttt{key}_b) \in \mathcal{E}_c^{\text{graph}}$ is added if following condition holds:

- Adjacency: the bin indices (i, j) of $\texttt{key}_a$ and $\texttt{key}_b$ satisfy: $|i_a - i_b| \leq 1$ and $|j_a - j_b| \leq 1$.
- Let γ_x and γ_y be tunable gating constants that control the allowable spatial deviation between neighboring hypotheses. Then, the difference between Gaussian centers is bounded relative to their spatial uncertainty, as given below:

$$\begin{aligned} |\mu_x^{(a)} - \mu_x^{(b)}| &\leq \gamma_x \min(\sigma_x^{(a)}, \sigma_x^{(b)}), \\ |\mu_y^{(a)} - \mu_y^{(b)}| &\leq \gamma_y \min(\sigma_y^{(a)}, \sigma_y^{(b)}). \end{aligned} \tag{7}$$

- The hypotheses exhibit sufficient geometric overlap and statistical proximity, defined as follows:

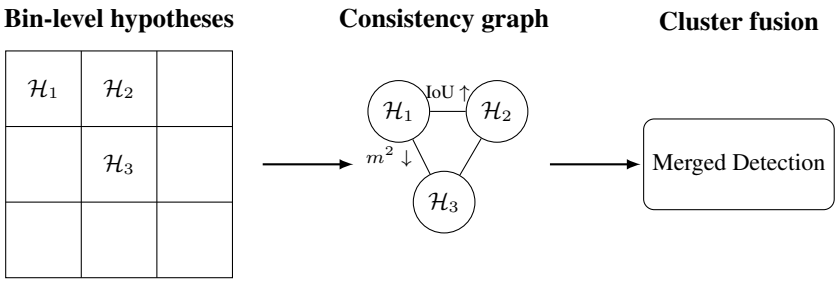


Figure 6: Cross-bin hypothesis merging. Neighboring bins produce Gaussian hypotheses that may correspond to the same physical object. A spatial consistency graph is constructed between hypotheses and connected components are merged together producing the final fused detection.

– *Geometric overlap:* The intersection-over-union (IoU) between the bounding boxes satisfies:

$$\mathrm{IoU}(\mathbf{b}^{(a)}, \mathbf{b}^{(b)}) \geq \tau_{\mathrm{high}}$$

– *Statistical proximity:* The squared Mahalanobis distance between the Gaussian parameters satisfies:

$$m^2\big((\mu^{(a)}, \Sigma^{(a)}), (\mu^{(b)}, \Sigma^{(b)})\big) \leq \tau_m \quad \text{and} \quad \mathrm{IoU}(\mathbf{b}^{(a)}, \mathbf{b}^{(b)}) \geq \tau_{\min}.$$

where τ_{high} and $\tau_{\min}$ denote high and minimum overlap thresholds, respectively, and τ_m controls statistical similarity.

Let $\mathcal{K}_c = \{K_1, K_2, \dots\}$ denote the set of connected components of $\mathcal{G}_c$, where each $K \subseteq \mathcal{V}_c$ is a set of vertices (i.e., class-bin keys). Each component K therefore represents a merged object hypothesis formed by grouping spatially-consistent local hypotheses $\{\mathcal{H}_{\mathtt{key}} \mid \mathtt{key} \in K\}$. Figure 6 depicts the construction of the spatial consistency graph and subsequent cluster fusion. Local hypotheses $\mathcal{H}_{\mathtt{key}}$ arising from neighboring bins are linked based on geometric overlap and statistical proximity, and connected components are aggregated to produce the final detection.

Cluster Fusion and Confidence Recovery. For each connected component $K \in \mathcal{K}_c$, merged hypothesis is constructed by aggregating the moment vectors of all participating entries, hence we have:

$$\mathbf{s}^K = \sum_{\mathtt{key} \in K} \mathbf{s}_{\mathtt{key}}.$$

The Gaussian parameters $(\mu_x^K, \mu_y^K, \sigma_x^K, \sigma_y^K)$ and the corresponding bounding box $\mathbf{b}^K$ are then recovered from $\mathbf{s}^K$ using the moment inversion and Gaussian-to-bounding-box mapping described previously.

The associated confidence statistics are obtained from the fused payload by aggregating the corresponding quantities across the cluster:

$$\hat{m}^K = \sum_{\mathtt{key} \in K} M_{\mathtt{key}}^{\mathrm{fused}}, \qquad \hat{c}^K = \sum_{\mathtt{key} \in K} C_{\mathtt{key}}^{\mathrm{fused}}.$$

The fused confidence score for the hypothesis is therefore: $\hat{p}^K = \frac{\hat{m}^K}{\hat{c}^K}$. The final fused detection set is given by:

$$\mathcal{O} = \left\{ (\mu_x^K, \mu_y^K, \sigma_x^K, \sigma_y^K, \mathbf{b}^K, \hat{m}^K, \hat{p}^K) \;\middle|\; K \in \mathcal{K}_c,\ c \in \mathcal{C} \right\},$$

such that each element corresponds to a reconstructed object obtained from merging a cluster of spatially consistent hypotheses. This completes the reconstruction pipeline, mapping encrypted multi-vendor observations to a consistent set of fused object detections. Algorithm 4 summarizes the complete methodology discussed in this section in pseudocode form. Figure 7 shows the obtained Gaussian splat (Figure 7a) after HE merging and the respective detection obtained (Figure 7b) after cluster fusion.

4.5 Complexity Analysis

4.5.1 Vendor Payload Construction

Let N denote the number of detections produced by a vendor and let B_v denote the number of occupied class-bin keys for that vendor i.e.,$\mathtt{key} = (c, (i, j))$ after spatial binning.

(a) Gaussian splat after HE merging of RT-DETR and DETR101. (b) Bounding box obtained after cluster fusion.

Figure 7: HE Fusion: RT-DETR and DETR101.

1. **Per-Detection Processing:** For each detection, Sarus performs (i) Gaussian parameter computation, (ii) moment vector construction, and (iii) soft assignment to at most four neighboring bins. Each operation takes constant time, yielding:
$$T_{\text{preprocess}} = O(N).$$
2. **Per-Key Aggregation:** Each detection contributes to at most four keys, and the corresponding moment vectors are accumulated into per-key aggregates. Since the number of updates per detection is bounded by a constant, the total cost is:
$$T_{\text{aggregation}} = O(N).$$
3. **Payload Creation:** The payload is constructed by iterating over all occupied keys, resulting in:
$$T_{\text{payload}} = O(B_v),$$
where typically $B_v \ll N$ due to spatial aggregation.

Combining the above steps, the total time complexity is:

$$T_{\text{vendor}} = O(N + B_v) = O(N). \tag{8}$$

This linear complexity is achieved through local aggregation, which avoids per-detection payload expansion.

4.5.2 Encrypted Multi-Vendor Fusion

Let $V = |\mathcal{V}|$ denote the number of participating vendors and let B denotes the union of keys across all vendors.

1. **Key Collection:** The fusion server first computes the union of keys contributed by all vendors i.e., $\mathcal{K} = \bigcup_{v \in \mathcal{V}} \text{keys}(\mathcal{P}_v)$. If each vendor contributes at most B occupied keys, then collecting the union requires:
$$T_{\text{keys}} = O(BV).$$
2. **Per-Key Fusion:** For each key $\texttt{key} \in \mathcal{K}$, the server scans the vendor payloads and combines the available entries. Ignoring the cost of encryption-specific operations, each per-key update is constant time: one addition for the fused moment accumulator and constant-time updates for the associated plaintext statistics. Since there are at most B keys and at most V vendors contributing to each key, the total cost is:
$$T_{\text{fusion}} = O(BV).$$
3. **Fused Payload Construction:** After aggregation, the server stores one fused entry per occupied key, yielding:
$$T_{\text{output}} = O(B).$$

Combining the above steps, the total time complexity of encrypted multi-vendor fusion by the server is:

$$T_{\text{fusion}} = O(BV + B) = O(BV). \tag{9}$$

$O(BV)$ result comes from the fact that Sarus fuses aggregated per-key payloads, not individual detections. Without vendor-side aggregation, the server would instead process per-detection contributions, leading to a larger dependence on the number of detections. Hence, the server-side fusion complexity scales linearly with both the number of occupied class-bin keys and the number of participating multi-vendors.

Algorithm 4 Encrypted Fused Payload to Fused Detection

Require: Server fused payload $\mathcal{P}^{\text{fused}}$:
`fused_cipher_by_key`: $\{\texttt{key} : (\texttt{class\_id}, \texttt{bin\_id}), \texttt{ciphertext\_bytes}\}$ ▷ Fused encrypted $\mathbf{v}$
`fused_mass_by_key`: $\{\texttt{key} : (\texttt{class\_id}, \texttt{bin\_id}), \texttt{float}\}$ ▷ Fused confidence mass
`fused_count_by_key`: $\{\texttt{key} : (\texttt{class\_id}, \texttt{bin\_id}), \texttt{float}\}$ ▷ Fused detection count
`bin_by_class` $\leftarrow \emptyset$ ▷ Group bins by class
$\mathcal{O} \leftarrow \emptyset$ ▷ Fused detections
for each $(\texttt{key}, \texttt{ciphertext\_bytes}) \in \texttt{fused\_cipher\_by\_key}$ **do**
  $(\texttt{class\_id}, \texttt{bin\_id}) \leftarrow \texttt{key}$
  $\mathbf{s} \leftarrow \mathsf{Dec}_{\mathsf{HE}}(\texttt{ciphertext\_bytes})$ ▷ Homomorphic decryption
  if $\mathbf{s}[0] \leq \epsilon$ **then**
    `continue`
  end if
  $(\mu_x, \mu_y, \sigma_x, \sigma_y) \leftarrow$ **InvertMoment**$(\mathbf{s})$
  $\mathbf{b} \leftarrow$ **GaussianToBBox**$(\mu_x, \mu_y, \sigma_x, \sigma_y, \lambda)$
  if $\text{len}(\texttt{bin\_by\_class}[\texttt{class\_id}]) = 0$ **then**
    $\texttt{bin\_by\_class}[\texttt{class\_id}] \leftarrow [\texttt{class\_id}, \texttt{bin\_id}, \mathbf{s}[0], \mu_x, \mu_y, \sigma_x, \sigma_y, \mathbf{b}, \mathbf{s}]$
  else
    $\texttt{bin\_by\_class}[\texttt{class\_id}].\text{append}([\texttt{class\_id}, \texttt{bin\_id}, \mathbf{s}[0], \mu_x, \mu_y, \sigma_x, \sigma_y, \mathbf{b}, \mathbf{s}])$
  end if
end for
for each $(c, \mathcal{E}_c) \in \texttt{bin\_by\_class}$ **do**
  $\mathcal{O}_c \leftarrow$ **Merge**$(\mathcal{E}_c, \texttt{fused\_mass\_by\_key}, \texttt{fused\_count\_by\_key})$
  $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{O}_c$
end for
Return $\mathcal{O}$

InvertMoments$(\mathbf{s})$**:**
$[S_w, S_{wc_x}, S_{wc_x^2}, S_{w\sigma_x^2}, S_{wc_y}, S_{wc_y^2}, S_{w\sigma_y^2}] \leftarrow \mathbf{s}$
$\mu_x \leftarrow S_{wc_x}/S_w$, $\mu_y \leftarrow S_{wc_y}/S_w$
$v_x \leftarrow (S_{w\sigma_x^2} + S_{wc_x^2})/S_w - \mu_x^2$, $v_y \leftarrow (S_{w\sigma_y^2} + S_{wc_y^2})/S_w - \mu_y^2$
$v_x \leftarrow \max(v_x, \epsilon)$, $v_y \leftarrow \max(v_y, \epsilon)$
$\sigma_x \leftarrow \sqrt{v_x}$, $\sigma_y \leftarrow \sqrt{v_y}$
Return $(\mu_x, \mu_y, \sigma_x, \sigma_y)$

GaussianToBBox$(\mu_x, \mu_y, \sigma_x, \sigma_y, \lambda)$**:**
$x_1 \leftarrow (\mu_x - \lambda\sigma_x)$, $y_1 \leftarrow (\mu_y - \lambda\sigma_y)$
$x_2 \leftarrow (\mu_x + \lambda\sigma_x)$, $y_2 \leftarrow (\mu_y + \lambda\sigma_y)$
Return $[x_1, y_1, x_2, y_2]$

Merge$(\texttt{entries}, \texttt{fused\_mass\_by\_key}, \texttt{fused\_count\_by\_key})$**:**
Require: Per-bin entries $\texttt{entries} = \{e_i\}$ for a fixed class
$\mathcal{O}_{\text{cls}} \leftarrow \emptyset$
$n \leftarrow |\texttt{entries}|$
Initialize adjacency list $\texttt{Adj}[0 \ldots n-1]$
for all pairs (i, j) with $i < j$ **do** ▷ Ensure undirected graph
  if $\texttt{entries}[i].\texttt{bin\_id}$ and $\texttt{entries}[j].\texttt{bin\_id}$ are not neighbors **then**
    continue
  end if
  $\Delta x \leftarrow |\texttt{entries}[i].\mu_x - \texttt{entries}[j].\mu_x|$
  $\Delta y \leftarrow |\texttt{entries}[i].\mu_y - \texttt{entries}[j].\mu_y|$
  $\sigma_{x,\min} \leftarrow \min(\texttt{entries}[i].\sigma_x, \texttt{entries}[j].\sigma_x)$
  $\sigma_{y,\min} \leftarrow \min(\texttt{entries}[i].\sigma_y, \texttt{entries}[j].\sigma_y)$

if $\Delta x > \gamma_x \sigma_{x,\min}$ **or** $\Delta y > \gamma_y \sigma_{y,\min}$ **then** ▷ Center distance gating
  continue
end if
$IoU \leftarrow \text{IoU}(\texttt{entries}[i].\texttt{bbox}, \texttt{entries}[j].\texttt{bbox})$ ▷ IoU calculation
$m^2 \leftarrow \text{Maha}^2((\mu_i, \sigma_i), (\mu_j, \sigma_j))$ ▷ Mahalanobis distance
if $IoU \geq \tau_{\text{strong}}$ **or** $(m^2 \leq \tau_m \wedge IoU \geq \tau_{\text{floor}})$ **then** ▷ Geometric overlap and statistical proximity
  add edge $i \leftrightarrow j$ to `Adj` ▷ Add edge from i to j
end if
end for
$\mathcal{G} \leftarrow (\texttt{Vertex}, \texttt{Edge})$ constructed from `Adj` ▷ Undirected graph: `Edge` between bins (`Vertex`) if adjacent
$\mathcal{K} \leftarrow$ **ConnectedComponents**$(\mathcal{G})$ ▷ Perform Breadth First Search on $\mathcal{G}$
for all clusters $K \in \mathcal{K}$ **do**
  if $|K| = 1$ **then**
    $e_k \leftarrow \texttt{entries}[k]$ ▷ $k \leftarrow$ the single element of K
    $M_k \leftarrow \texttt{fused\_mass\_by\_key}[(e_k.\texttt{class\_id}, e_k.\texttt{bin\_id})]$
    $A_k \leftarrow \texttt{fused\_count\_by\_key}[(e_k.\texttt{class\_id}, e_k.\texttt{bin\_id})]$
    $\hat{p}_k \leftarrow M_k / A_k$
    $\mathcal{O}_{\text{cls}}.\text{append}(e_k.\mu_x, e_k.\mu_y, e_k.\sigma_x, e_k.\sigma_y, e_k.\mathbf{b}, M_k, \hat{p}_k)$
    continue
  end if
  $\mathbf{s}^K \leftarrow \sum_{k \in K} \texttt{entries}[k].\mathbf{s}$
  $(\mu_x^K, \mu_y^K, \sigma_x^K, \sigma_y^K) \leftarrow$ **InvertMoments**$(\mathbf{s}^K)$
  $\mathbf{b}^K \leftarrow$ **GaussianToBBox**$(\mu_x^K, \mu_y^K, \sigma_x^K, \sigma_y^K, \lambda)$
  if $(\sigma_x^K > \kappa_\sigma \max_{k \in K}(\sigma_{x,k})$ **or** $\sigma_y^K > \kappa_\sigma \max_{k \in K}(\sigma_{y,k}))$ **or** $(\text{Area}(b^K) > \kappa_A \max_{k \in K}(\text{Area}(b_k)))$ **then**
    for $k \in K$ **do**
      $e_k \leftarrow \texttt{entries}[k]$
      $M_k \leftarrow \texttt{fused\_mass\_by\_key}[(e_k.\texttt{class\_id}, e_k.\texttt{bin\_id})]$
      $A_k \leftarrow \texttt{fused\_count\_by\_key}[(e_k.\texttt{class\_id}, e_k.\texttt{bin\_id})]$
      $\hat{p}_k \leftarrow M_k / A_k$
      $\mathcal{O}_{\text{cls}}.\text{append}(e_k.\mu_x, e_k.\mu_y, e_k.\sigma_x, e_k.\sigma_y, e_k.\mathbf{b}, M_k, \hat{p}_k)$
    end for
    continue
  end if
  $\hat{m} \leftarrow \sum_{k \in K} \texttt{fused\_mass\_by\_key}[(\texttt{entries}[k].\texttt{class\_id}, \texttt{entries}[k].\texttt{bin\_id})]$ ▷ Confidence mass
  $\hat{c} \leftarrow \sum_{k \in K} \texttt{fused\_count\_by\_key}[(\texttt{entries}[k].\texttt{class\_id}, \texttt{entries}[k].\texttt{bin\_id})]$ ▷ Aggregation weight
  $\hat{p} \leftarrow \frac{\hat{m}}{\hat{c}}$ ▷ Fused confidence
  $\mathcal{O}_{\text{cls}}.\text{append}(\mu_x^K, \mu_y^K, \sigma_x^K, \sigma_y^K, \mathbf{b}^K, \hat{m}, \hat{p})$ ▷ Append the merged object
end for

4.5.3 Fused Payload to Detection Reconstruction

Let B denote the number of occupied class-bin keys in the fused payload, and let B_c denote the number of reconstructed bin-level entries for class c, such that:

$$\sum_{c \in \mathcal{C}} B_c = B.$$

1. **Per-Key Reconstruction:** For each fused $\texttt{key} = (c, (i, j))$, Sarus decrypts the fused moment vector, inverts the moments to recover $(\mu_x, \mu_y, \sigma_x, \sigma_y)$, and reconstructs the corresponding bounding box. Ignoring the cost of decryption, each of these operations takes constant time. Therefore, reconstructing all bin-level entries requires:
$$T_{\text{reconstruct}} = O(B).$$
2. **Class-wise Grouping:** The reconstructed entries are grouped by class before graph-based merging. This requires a single pass over the fused entries, giving:
$$T_{\text{group}} = O(B).$$
3. **Spatial Consistency Graph Construction:** For each class c, Sarus constructs a spatial consistency graph over the B_c reconstructed entries. In the worst case, all pairs of entries for that class are examined to determine

whether an edge should be added. Since each pairwise consistency check (bin adjacency, center gating, IoU, and Mahalanobis distance) takes constant time, the graph construction cost is:

$$T_{\text{graph}}^{(c)} = O(B_c^2).$$

Summing over all classes yields

$$T_{\text{graph}} = O\left(\sum_{c\in\mathcal{C}} B_c^2\right),$$

which, is upper bounded by:

$$T_{\text{graph}} = O(B^2).$$

4. **Connected Components:** For each class-specific graph, connected components are computed using breadth-first search (BFS). BFS runs in time linear in the number of vertices and edges:

$$T_{\text{cc}}^{(c)} = O(B_c + E_c),$$

where E_c is the number of graph edges for class c. Since $E_c = \mathcal{O}(B_c^2)$ in the worst case, this becomes:

$$T_{\text{cc}}^{(c)} = O(B_c^2).$$

Therefore,

$$T_{\text{cc}} = O\left(\sum_{c\in\mathcal{C}} B_c^2\right) = O(B^2).$$

5. **Cluster Fusion:** Each connected component K is fused by summing its moment vectors, recovering Gaussian parameters, and computing the associated confidence statistics. Since each reconstructed entry belongs to exactly one connected component, the total work across all clusters is linear in the number of entries:

$$T_{\text{cluster}} = O(B).$$

Combining the above steps, the total time complexity of fused payload to detection reconstruction is

$$T_{\text{post}} = O\left(B + \sum_{c\in\mathcal{C}} B_c^2\right), \tag{10}$$

which, in the worst case, simplifies to: $T_{\text{post}} = O(B^2)$. Thus, reconstruction is dominated by class-wise graph construction and connected-component analysis, while the remaining stages are linear in the number of occupied fused keys. In practice, the effective cost is often substantially lower because spatial adjacency constraints limit the number of candidate pairs that can form graph edges.

Summary of Time Complexity:

1. **Vendor payload construction:** $T_{\text{vendor}} = O(N + B_v)$.
2. **Server encrypted fusion:** $T_{\text{fusion}} = O(BV)$.
3. **Post-fusion reconstruction:** $T_{\text{post}} = O(B + \sum_{c\in\mathcal{C}} B_c^2)$.

where, $N =$ detections per vendor, $V =$ vendors, $B_v =$ bins produced by one vendor, $B =$ fused bins and $B_c =$ bins per class.

5 Experiments

The dataset was collected using a real-world autonomous test vehicle equipped with a drive-by-wire system provided by Dataspeed Inc.[4]. The vehicle is instrumented with a multi-modal sensing suite, including a LiDAR sensor, an image sensor (Lucid Triton camera[5]), and a radar sensor. In this work, only the image stream is utilized for evaluating the Sarus framework. Sensor data acquisition is performed using the Robot Operating System (ROS) middleware within the perception architecture described in [52]. The experiments were conducted at the Virginia Tech Transportation Institute (VTTI) autonomous vehicle testing track [53], under controlled city-driving conditions.

[4] `https://www.dataspeedinc.com/`

[5] `https://thinklucid.com/product/triton-16-mp-imx273/`

The scene is designed to reflect realistic perception challenges. A stop sign is placed along the driving path of the ego vehicle and is partially occluded by a parked truck. Additional dynamic and static objects are present in the environment, including multiple vehicles and a pedestrian partially occluded by vegetation. To further emulate adverse environmental conditions, artificial rain is introduced during data collection.

To evaluate robustness under varying motion dynamics, the dataset is collected at three different vehicle speeds: 25 mph (142 frames), 35 mph (100 frames), and 55 mph (66 frames), a total of 308 frames. Increasing vehicle speed affects temporal sampling, motion blur, and relative object motion, while artificial rain introduces light scattering and attenuation effects. The combination of these factors produces realistic perception distortions that challenge detection consistency across vendors, enabling a rigorous evaluation of fusion stability under adverse and dynamic conditions.

This setup enables evaluation of perception fusion under realistic occlusion, multi-object interaction, adverse weather conditions and under varying speeds.

Multi-Vendor Perception Setup. To emulate a collaborative multi-vendor perception environment, each frame is processed using five heterogeneous object detection models: YOLOv8, YOLOv9, DETR-50, DETR-101, and RT-DETR. Each model is treated as an independent vendor providing its own set of detection outputs.

To capture varying levels of vendor participation, we evaluate 26 non-empty subsets of these models, corresponding to combinations of 2, 3, 4, and 5 vendors. Each subset represents a distinct collaborative scenario with differing degrees of redundancy and diversity in perception. For each frame and each vendor subset, fusion is performed independently, resulting in a total of:

$$308 \times 26 = 8008$$

fusion instances. This exhaustive enumeration enables a comprehensive evaluation of the proposed framework across a wide range of multi-vendor configurations. This design allows us to systematically analyze how fusion performance varies with the number and diversity of participating vendors.

5.1 Equivalence of Homomorphic and Plaintext Fusion

The primary objective of the experiment is to validate the *functional and numerical correctness* of the proposed homomorphic fusion pipeline. Specifically, the aim is to verify that fusion performed entirely in the encrypted domain reproduces the same outputs as plaintext fusion, up to negligible numerical error induced by approximate homomorphic encryption. Such a correctness validation is a necessary prerequisite before evaluating downstream perception performance on labeled benchmarks, as any accuracy gains or losses would be meaningless without first demonstrating that encryption itself does not alter fusion behavior.

Fusion Modes. For each frame and each vendor subset, fusion is executed under two computational settings:

1. **Plaintext fusion:** the fusion pipeline operates directly on unencrypted detection outputs, serving as the reference baseline
2. **Homomorphic fusion:** the identical fusion pipeline is executed over CKKS-encrypted representations of the detection payloads, where all aggregation operations are performed in the encrypted domain without access to plaintext data.

In both settings, the underlying fusion logic, including moment aggregation, spatial binning, and hypothesis merging, remains unchanged. This ensures that any observed differences in output or runtime are solely attributable to the use of homomorphic encryption.

Table 2 quantitatively compares the outputs of homomorphic fusion and plaintext fusion across different driving regimes. The results demonstrate near-perfect agreement between the two pipelines.

Across all speed settings, the intersection-over-union (IoU) between bounding boxes produced by homomorphic and plaintext fusion exceeds 0.99997 on average, with the 95^{th} percentile and maximum values reaching 1.000. This indicates that the spatial extent of the reconstructed detections is effectively identical in both settings.

The absolute deviations in bounding box centers and sizes remain within sub-pixel to pixel-level precision, with $\max|\Delta c| \leq 0.5$ pixels and $\max|\Delta s| \leq 2.0$ pixels across all experiments. Similarly, the differences in the recovered Gaussian standard deviations are negligible, with $\max|\Delta\sigma_x|$ and $\max|\Delta\sigma_y|$ remaining below 0.05 pixels.

These small discrepancies arise from the approximate nature of CKKS-based homomorphic encryption, which introduces bounded numerical error during arithmetic operations. Importantly, the magnitude of these errors is insufficient to affect the geometric interpretation or downstream decision-making of the fused detections.

Table 2: Equivalence between homomorphic and plaintext fusion across driving regimes.

Speed	**N**	**IoU(HE,Plain)**			**BBox Δ (px)**		**$\Delta\sigma$ (px)**	
		mean	p95	max	$\max \lvert\Delta c\rvert$	$\max \lvert\Delta s\rvert$	$\max \lvert\Delta\sigma_x\rvert$	$\max \lvert\Delta\sigma_y\rvert$
25 mph	3692 ($= 142 \times 26$)	0.999989	1.000	1.000	0.5	1.0	0.048	0.022
35 mph	2600 ($= 100 \times 26$)	0.999978	1.000	1.000	0.5	1.0	0.016	0.005
55 mph	1716 ($= 66 \times 26$)	0.999991	1.000	1.000	0.0	2.0	0.016	0.001

Overall, these results confirm that the proposed homomorphic fusion pipeline preserves the correctness of plaintext fusion with high numerical fidelity, validating the use of encrypted computation for privacy-preserving multi-vendor perception.

5.2 Computational Performance Evaluation

To evaluate the computational behavior of Sarus, we conduct a controlled scaling study that varies both the number of detections per scene and the number of participating vendors. Specifically, we consider detection counts:

$$N \in \{1, 5, 10, 15, 20, 25\},$$

which correspond to increasing scene complexity, ranging from sparse to dense object configurations. For each setting, we simulate a collaborative perception environment with:

$$V \in \{2, 3, 4, 5\}$$

independent vendors, where each vendor contributes its own detection outputs.

This setup allows us to systematically analyze how the computational cost of the pipeline scales with (i) the number of objects in the scene and (ii) the number of participating vendors. All measurements are reported separately for the three major stages of the pipeline: (i) vendor-side payload construction, (ii) encrypted multi-vendor fusion at the server, and (iii) fused detection reconstruction.

Unless otherwise stated, cryptographic primitives (encryption and decryption) are excluded when analyzing algorithmic scaling behavior, in order to isolate the structural complexity of the proposed method.

5.2.1 Vendor-Side Payload Construction

Runtime vs. Occupied Bins. Figure 8a shows the relationship between payload construction time and the number of occupied class-bin keys B. The results exhibit an almost perfectly linear trend, with a fitted model $T = 4.528 \cdot B + 2.446$ and $R^2 = 0.9995$.

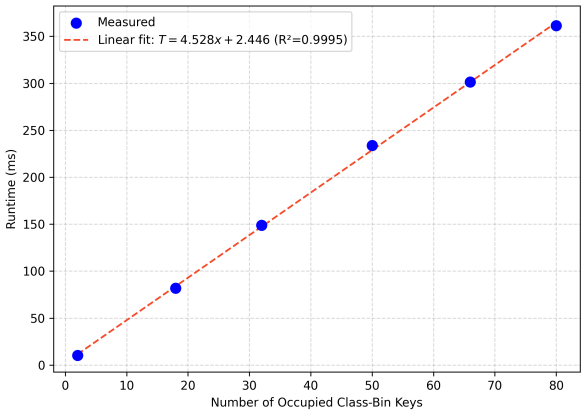
This provides strong empirical validation that vendor-side runtime scales as $O(B)$, confirming that the dominant cost arises from per-bin operations such as encryption and serialization, rather than from the raw number of detections.

Stage-wise Runtime Breakdown. Figure 8b decomposes runtime into preprocessing, encryption, and serialization. Encryption clearly dominates across all detection counts, accounting for the majority of total runtime (e.g., ≈ 250 ms out of ≈ 360 ms at $N = 25$). Preprocessing remains negligible, while serialization contributes a moderate but consistent overhead.

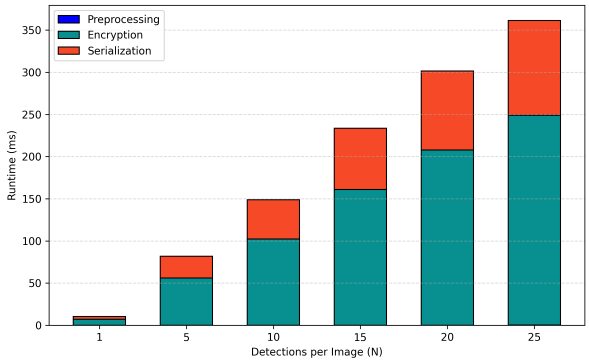
Importantly, both encryption and serialization scale proportionally with B, reinforcing that cryptographic operations are the primary performance bottleneck.

Homomorphic vs. Plaintext Runtime. Figure 8c compares homomorphic and plaintext payload construction on a logarithmic scale. While both exhibit increasing trends with N, homomorphic execution is several orders of magnitude slower due to CKKS encryption overhead. However, both curves remain approximately linear, indicating that homomorphic processing preserves the same asymptotic behavior as plaintext execution.

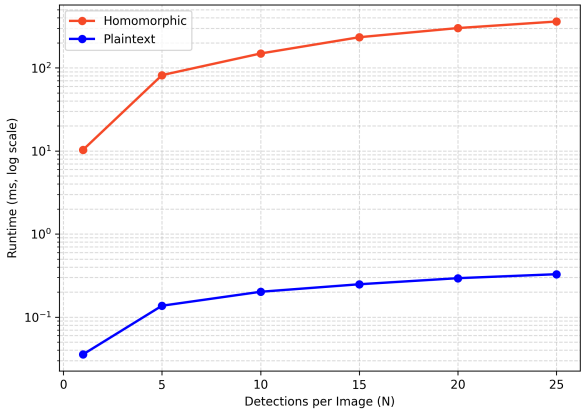
Relative Overhead of Homomorphic Encryption. Figure 8d shows the ratio of homomorphic to plaintext runtime. The overhead ranges from approximately $300\times$ at low detection counts to over $1000\times$ at higher densities. This increase is driven by the growth in occupied bins, which amplifies the number of expensive encryption operations. Despite this overhead, the linear scaling behavior ensures predictable performance, making the system amenable to optimization through batching, parameter tuning, or hardware acceleration.



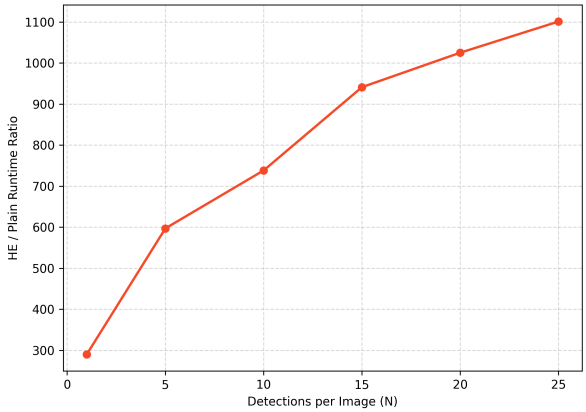
(a) Total runtime as a function of occupied class-bin keys, showing a strong linear relationship ($R^2 = 0.9995$).



(b) Stage-wise runtime breakdown highlighting encryption as the dominant cost.



(c) Comparison between homomorphic and plaintext payload construction (log scale).



(d) Relative overhead of homomorphic encryption compared to plaintext.

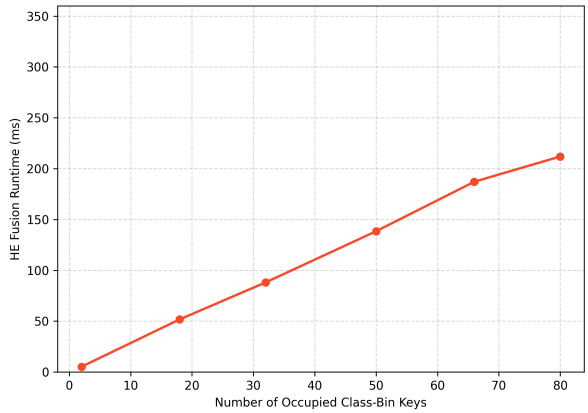
Figure 8: Vendor-side payload construction performance.

Discussion. Overall, the vendor-side evaluation confirms that payload construction scales linearly with the number of occupied bins rather than the number of detections. This distinction is critical, as spatial binning effectively compresses multiple detections into fewer encrypted representations, reducing the number of cryptographic operations.

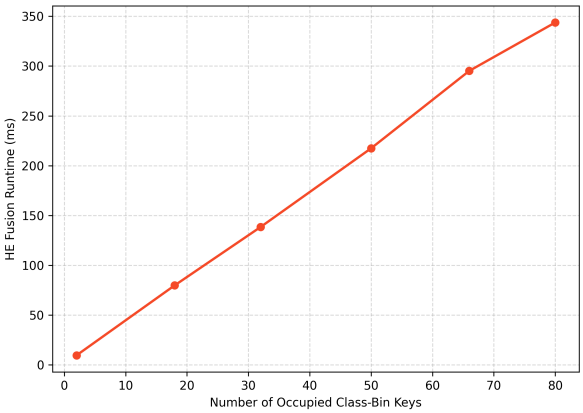
While homomorphic encryption introduces significant constant-factor overhead, the preservation of linear scaling ensures that the system remains predictable and scalable. These results validate the theoretical complexity of $O(N) + O(B)$ and highlight spatial aggregation as a key mechanism for controlling computational cost.

Network Overhead: The use of homomorphic encryption introduces a significant increase in payload size compared to plaintext representations due to ciphertext expansion and encoding overhead. In our experiments, the encrypted payload size grows from approximately 0.65 MB at $N = 1$ to over 26 MB at $N = 25$, whereas the corresponding plaintext payload remains below 10 KB.

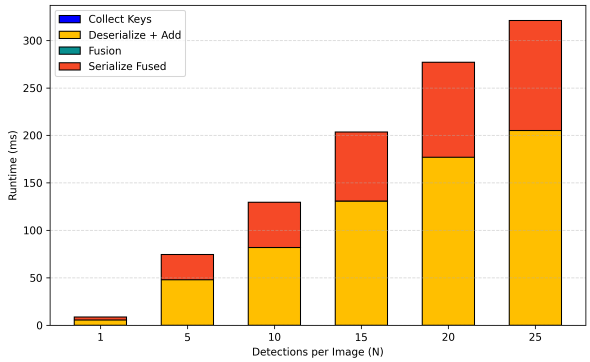
This results in a ciphertext expansion factor of approximately 3×10^3, which stabilizes as the number of detections increases. Despite this large constant-factor overhead, the payload size grows approximately linearly with the number of occupied bins, ensuring predictable network cost as a function of scene complexity. Furthermore, the bin-wise aggregation strategy limits the number of transmitted ciphertexts, preventing excessive growth in communication overhead even in dense detection scenarios. This predictable scaling behavior is critical for deployment, as it enables accurate estimation of bandwidth requirements despite the inherent overhead of homomorphic encryption.



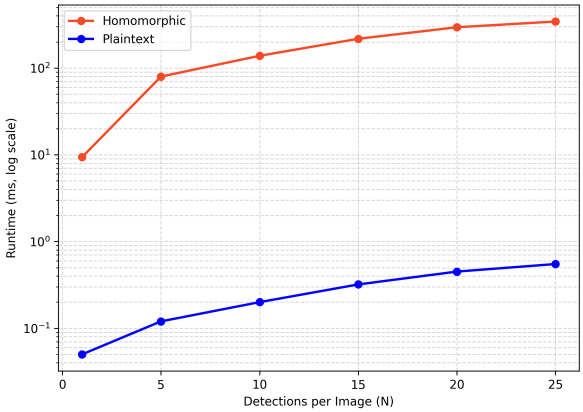
(a) HE fusion runtime as a function of occupied class-bin keys for $V = 2$.



(b) HE fusion runtime as a function of occupied class-bin keys for $V = 5$.



(c) Stage-wise breakdown highlighting ciphertext deserialization and serialization as the dominant costs.



(d) Comparison: homomorphic vs plaintext fusion (log scale), showing constant-factor overhead of encrypted computation.

Figure 9: Server-side fusion performance.

5.2.2 Server-Side Encrypted Fusion

Figures 9a and 9b show that server-side HE fusion runtime scales linearly with the number of occupied class-bin keys B. Comparing $V = 2$ and $V = 5$, the slope increases with the number of vendors, confirming that the fusion cost grows proportionally with both B and V.

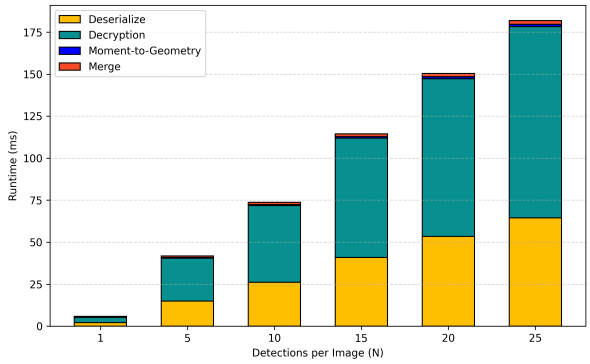
Figure 9c provides a stage-wise breakdown, showing that the dominant cost arises from ciphertext deserialization and accumulation, followed by serialization of the fused result. In contrast, key collection and fusion arithmetic contribute negligibly to the overall runtime.

Figure 9d compares homomorphic and plaintext fusion, demonstrating that while homomorphic processing introduces a significant constant-factor overhead, both exhibit similar growth trends with increasing problem size.

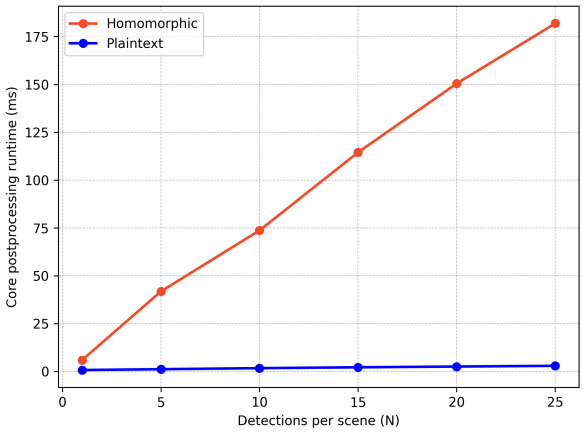
Empirically, the server-side fusion runtime follows a highly linear trend with respect to the number of occupied bins, with fitted slopes of approximately 2.70 ms/bin for $V = 2$ and 4.35 ms/bin for $V = 5$ ($R^2 = 0.9971$ and 0.9990, respectively), confirming the expected growth with both B and the number of participating vendors V.

5.2.3 Fused Detection Reconstruction

The vendor-side postprocessing results further validate the linear scaling behavior of the proposed pipeline. As shown in Figure 10, the total homomorphic runtime grows approximately linearly with the number of detections, which directly corresponds to the number of occupied bins. The stage-wise breakdown reveals that ciphertext decryption dominates the computational cost, while moment-to-geometry transformation and bin merging contribute only marginal overhead.



(a) Stage-wise breakdown of vendor-side postprocessing under homomorphic encryption. Decryption dominates runtime, while moment-to-geometry transformation and merging incur negligible cost.



(b) Comparison of core vendor postprocessing runtime between homomorphic and plaintext execution. Homomorphic processing exhibits linear scaling with a higher constant factor, while plaintext remains near-negligible.

Figure 10: Vendor-side postprocessing performance. Homomorphic processing introduces significant computational overhead due to ciphertext operations, while preserving linear scaling with respect to the number of occupied bins.

In contrast, plaintext postprocessing remains negligible across all detection counts, highlighting that the additional cost is entirely attributable to homomorphic operations rather than the underlying fusion logic. Importantly, despite the large constant-factor overhead, the absence of super-linear growth confirms that the system scales predictably as $O(B)$, making it suitable for deployment under bounded scene complexity.

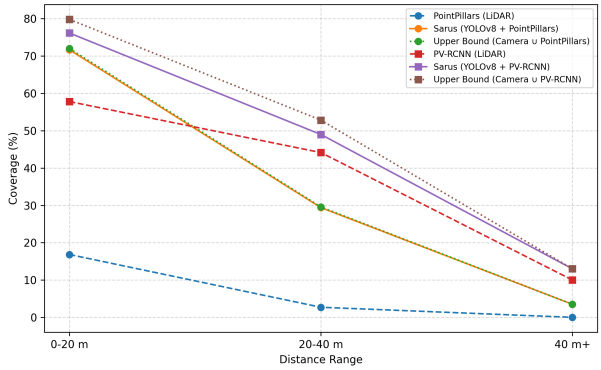
This result is particularly significant, as it demonstrates that the dominant cost arises from unavoidable cryptographic primitives rather than algorithmic inefficiencies, preserving scalability despite strong privacy guarantees.

5.3 Situational Awareness Evaluation

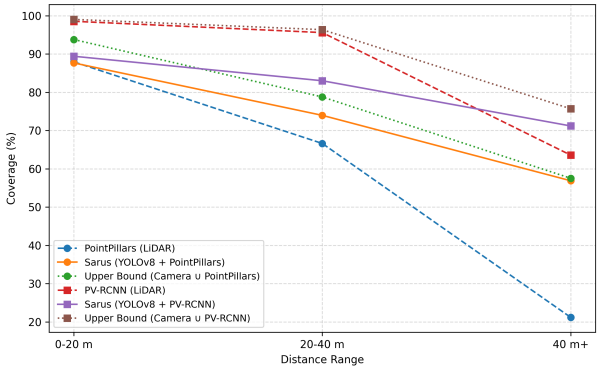
In cooperative perception settings, the primary objective is to maximize the coverage of objects in the scene by aggregating complementary detections across heterogeneous perception systems. Hence, we evaluate the *situational awareness*, rather than conventional detection accuracy. We conduct experiments on the KITTI [9] object detection benchmark and focus on the *Car* and *Pedestrian* classes, we use training split with corresponding annotations. The dataset provides synchronized RGB images and LiDAR point clouds, enabling evaluation of multi-modal perception and fusion. Camera-based detections are obtained using YOLOv8 [10, 54], while LiDAR-based detections are produced using PointPillars [55] and PV-RCNN [11]. Evaluation is done on three configurations: (i) LiDAR-only baseline, (ii) Sarus, and (iii) an *Upper Bound* corresponding to the union of camera and LiDAR detections. The LiDAR models are implemented using the OpenPCDet framework [56], which provides standardized training and inference pipelines for 3D object detection on KITTI. Pretrained models are used to ensure consistent and reproducible performance. We define *coverage* as the fraction of ground-truth objects that are matched by at least one detection. Matching is determined using an Intersection-over-Union (IoU) threshold. We report results at IoU at 0.3, which provides a more appropriate measure of situational awareness while still requiring meaningful spatial overlap. For completeness, we also observe that higher IoU thresholds lead to lower apparent coverage due to reconstruction effects, rather than failure to detect objects.

To analyze the spatial behavior of fusion, we stratify evaluation by object distance into three non-overlapping bins: $[0, 20)$ m, $[20, 40)$ m, and ≥ 40 m. This partitioning captures the distance dependent sensing characteristics of camera and LiDAR modalities. Specifically, LiDAR performance degrades with increasing range due to reduced point cloud density and sparsity, whereas camera-based detection remains comparatively robust. This allows us to isolate regimes of modality complementarity and quantify the effectiveness of fusion under varying sensing conditions.

The results in Figure 11 illustrate the distance dependent behavior of the fusion framework across both Pedestrian and Car classes. When paired with PointPillars (see Figure 11a and Figure 11b), the framework leads to substantial improvements in coverage across all distance ranges, particularly for pedestrians and at longer distances where LiDAR-based detection degrades due to reduced point cloud density. In contrast, when combined with PV-RCNN, which



(a) **Pedestrian:** Significant improvement in coverage when paired with PointPillars, and preserves performance with PV-RCNN, with gains primarily in near and mid ranges.



(b) **Car:** Substantial improvement for PointPillars and yields complementary gains at longer ranges even with PV-RCNN, reflecting distance-dependent modality complementarity.

Figure 11: Distance-based coverage analysis (IoU = 0.3). Coverage is reported across three distance ranges: $[0, 20)$ m, $[20, 40)$ m, and ≥ 40 m. Sarus improves situational awareness by aggregating complementary detections from camera and LiDAR modalities. Gains are most pronounced when detectors operate in challenging regimes, such as long-range perception and LiDAR performance degrades due to sparsity.

achieves higher baseline coverage, Sarus largely preserves the existing performance while recovering a portion of the remaining missed detections. Notably, even in this setting, consistent gains are observed at longer ranges for cars, highlighting the role of cross-modal complementarity under challenging sensing conditions. These results indicate that the effectiveness of fusion is governed by both the baseline detector performance and the distance-dependent sensing characteristics of the modalities. Overall, framework effectively aggregates complementary information across camera and LiDAR inputs, improving situational awareness while maintaining a compact, privacy-preserving representation.

6 Limitations and Future Work

Approximate Representation and Localization: Sarus relies on Gaussian moment aggregation over spatial bins to enable efficient and privacy-preserving fusion. This representation requires selecting class-specific spatial anchors S and strides s, defined with respect to a common reference frame $\mathcal{F}_{\text{common}}$ (see Definition 1). These parameters determine how detections are assigned to bins and therefore influence both communication cost and reconstruction fidelity. While the moment-based representation preserves object-level awareness, it introduces approximation during binning and reconstruction, which can degrade precise bounding-box localization. This effect is reflected in reduced performance at higher IoU thresholds. Future work will explore adaptive choices of S and s, learned or data-dependent binning strategies, and higher-order representations to better preserve geometric fidelity without sacrificing efficiency.

Dependence on Detection Quality: The effectiveness of Sarus is influenced by the quality and complementarity of the underlying perception models. When baseline detectors already achieve high coverage, the relative gains from fusion may diminish. Conversely, larger improvements are expected when modalities or vendors exhibit complementary failure modes. A promising direction is to incorporate confidence-aware or reliability-weighted fusion mechanisms that adapt dynamically to detector performance.

Admission, Compliance, and Verifiable Inputs: Sarus currently assumes that participating vendors submit payloads that are admissible for fusion: they follow the agreed schema, originate from legitimate participants, and conform to the public perception-fusion specification. In multi-vendor or adversarial settings, this assumption may not hold automatically. An important extension is to integrate Sarus with an admission and compliance layer based on credentialed vehicular communication, proof-carrying data, or succinct zero-knowledge compliance proofs. For example, a Hermes Seal–style mechanism could allow vendors to attach evidence that their submitted encrypted payloads satisfy a public specification without revealing raw sensor data, private detections, or proprietary model details. This would strengthen trust in the cooperative perception pipeline while preserving the modular role of Sarus as the encrypted fusion layer.

Key Management: In our current formulation, the homomorphic encryption context, public encryption material, and decryption authority are assumed to be established before fusion begins. This leaves open important system-design

questions, including which party generates the homomorphic encryption keys, which participants are authorized to decrypt the fused result, how secret keys are protected, and how key rotation or revocation should be handled when vendors join or leave the cooperative perception group. Future work will investigate threshold and multi-key variants of homomorphic encryption, where decryption of the fused output requires participation from multiple authorized parties rather than a single trusted key holder. Another promising direction is to combine encrypted aggregation with secure multi-party computation (MPC), enabling stronger control over key custody, joint decryption, and access policies for fused perception outputs.

Limited Modalities and Datasets: Our evaluation focuses on camera and LiDAR fusion on the KITTI dataset. While this provides a controlled and reproducible benchmark, real-world deployments may involve additional modalities, such as radar, and more diverse operating environments. Extending Sarus to broader multi-modal and multi-dataset settings, including large-scale autonomous driving benchmarks, is an important direction for future work.

System-Level Integration: Sarus is currently evaluated as a perception-level fusion module. Integrating it into full autonomous driving stacks, including tracking, prediction, planning, and decision-making, remains an open challenge. Future work will investigate how improved situational awareness from privacy-preserving fusion translates into downstream safety and robustness gains in closed-loop systems.

7 Conclusion

In this work, we presented Sarus, a privacy-preserving framework for multi-vendor cooperative perception that enables secure aggregation of inference-time detection outputs using CKKS-based homomorphic encryption. By introducing a moment-based representation over a shared spatial lattice, Sarus enables efficient fusion of structured perception outputs without requiring access to raw detections or proprietary model information. The proposed design eliminates the need for plaintext sharing, addressing key privacy and confidentiality challenges in multi-vendor perception systems.

We showed that the framework achieves scalable performance through spatial binning, with vendor payload construction scaling linearly with the number of detections and server-side fusion scaling as $O(BV)$ with respect to the number of occupied bins and vendors. Experimental results demonstrate that homomorphic encryption introduces only a bounded constant-factor overhead, while maintaining linear scaling in practice. These findings indicate that privacy-preserving multi-vendor perception fusion is feasible for real-time deployment when statistical compression and spatial sparsity are jointly exploited.

More broadly, this work highlights the importance of inference-time privacy in collaborative AI systems, where sensitive outputs rather than training data must be protected. Future work includes extending the framework to stronger adversarial models, exploring alternative cryptographic primitives with improved efficiency, and integrating the approach into real-world autonomous driving and V2X systems.

References

[1] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative Perception for Connected Autonomous Vehicles Based on 3D Point Clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 514–524, 2019. https://doi.org/doi:10.1109/ICDCS.2019.00058.

[2] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2VNet: Vehicle-to-vehicle Communication for Joint Perception and Prediction. In *European conference on computer vision*, pages 605–621. Springer, 2020. https://doi.org/10.48550/arXiv.2008.07519.

[3] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. OPV2V: An Open Benchmark Dataset and Fusion Pipeline for Perception with Vehicle-to-Vehicle Communication. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2583–2589, 2022. https://doi.org/doi:10.1109/ICRA46639.2022.9812038.

[4] Chao Xiang, Chen Feng, Xiaopo Xie, Botian Shi, Hao Lu, Yisheng Lv, Mingchuan Yang, and Zhendong Niu. Multi-Sensor Fusion and Cooperative Perception for Autonomous Driving: A Review. *IEEE Intelligent Transportation Systems Magazine*, 15(5):36–58, 2023. https://doi.org/doi:10.1109/MITS.2023.3283864.

[5] Seong-Woo Kim, Baoxing Qin, Zhuang Jie Chong, Xiaotong Shen, Wei Liu, Marcelo H. Ang, Emilio Frazzoli, and Daniela Rus. Multivehicle cooperative driving using cooperative perception: Design and experimental validation. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):663–680, 2015. https://doi.org/doi:10.1109/TITS.2014.2337316.

[6] Bin Dai, Fanglin Xu, Yuanyuan Cao, and Yang Xu. Hybrid sensing data fusion of cooperative perception for autonomous driving with augmented vehicular reality. *IEEE Systems Journal*, 15(1):1413–1422, 2021. https://doi.org/doi:10.1109/JSYST.2020.3007202.

[7] Martin Boehme, Marco Stang, Ferdin Muetsch, and Eric Sax. TalkyCars: A Distributed Software Platform for Cooperative Perception. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 701–707, 2020. https://doi.org/doi:10.1109/IV47402.2020.9304630.

[8] University of Michigan Transportation Research Institute. Smart intersection project. https://sip.umtri.umich.edu/, 2023. Accessed: 2026.

[9] KITTI. KITTI Vision Benchmark Suite. https://www.cvlibs.net/datasets/kitti/, 2012. Accessed: 2026.

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. https://doi.org/10.1109/CVPR.2016.91.

[11] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. 2021. https://doi.org/10.48550/arXiv.1912.13192.

[12] Apostol Vassilev, Alina Oprea, Alice Fordyce, Hyrum Anderson, Xander Davies, and Maia Hamin. Adversarial machine learning: A taxonomy and terminology of attacks and mitigations, 2025. National Institute of Standards and Technology Gaithersburg, MD, NIST Trustworthy and Responsible AI, NIST AI 100-2e2025 https://doi.org/10.6028/NIST.AI.100-2e2025.

[13] Si Chen, Ruoxi Jia, and Guo-Jun Qi. Improved Techniques for Model Inversion Attacks . 2020.

[14] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. A Methodology for Formalizing Model-Inversion Attacks. In *2016 IEEE 29th computer security foundations symposium (CSF)*, pages 355–370. IEEE, 2016. https://doi.org/doi:10.1109/CSF.2016.32.

[15] Sayanton V Dibbo. Sok: Model inversion attack landscape: Taxonomy, challenges, and future roadmap. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 439–456. IEEE, 2023. https://doi.org/doi:10.1109/CSF57540.2023.00027.

[16] Wiz. Wiz Discovers Flaws in GenAI Models Enabling Customer Data Theft. https://www.infosecurity-magazine.com/news/wiz-discovers-flaws-generative-ai/, 2024. Accessed: 2026.

[17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pages 409–437. Springer, 2017.

[18] Alessandro Chiesa and Eran Tromer. Proof-carrying data and hearsay arguments from signature cards. In *Innovations in Computer Science (ICS)*, 2010.

[19] IEEE Standard for Wireless Access in Vehicular Environments—Security Services for Application and Management Messages, 2022.

[20] Munawar Hasan, Apostol Vassilev, Edward Griffor, and Thoshitha Gamage. Hermes Seal: Zero-Knowledge Assurance for Autonomous Vehicle Communications. *arXiv preprint arXiv:2603.26343*, 2026. https://doi.org/10.48550/arXiv.2603.26343.

[21] Shaowu Zheng, Chong Xie, Shanhu Yu, Ming Ye, Ruyi Huang, and Weihua Li. A robust strategy for roadside cooperative perception based on multi-sensor fusion. In *2022 International Conference on Sensing, Measurement & Data Analytics in the era of Artificial Intelligence (ICSMD)*, pages 1–6. IEEE, 2022. https://doi.org/10.1109/ICSMD57530.2022.10058282.

[22] Eduardo Arnold, Mehrdad Dianati, Robert de Temple, and Saber Fallah. Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):1852–1864, 2022. https://doi.org/10.1109/TITS.2020.3028424.

[23] Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. V2X-ViT: Vehicle-to-Everything Cooperative Perception with Vision Transformer. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 107–124, Cham, 2022. Springer Nature Switzerland. https://doi.org/10.48550/arXiv.2203.10638.

[24] Yunsheng Ma, Juanwu Lu, Can Cui, Sicheng Zhao, Xu Cao, Wenqian Ye, and Ziran Wang. MACP: Efficient model adaptation for cooperative perception. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3373–3382, 2024. https://doi.org/10.48550/arXiv.2310.16870.

[25] Andreas Rauch, Felix Klanner, Ralph Rasshofer, and Klaus Dietmayer. Car2X-based perception in a high-level fusion architecture for cooperative perception systems. In *2012 IEEE Intelligent Vehicles Symposium*, pages 270–275, 2012. https://doi.org/10.1109/IVS.2012.6232130.

[26] Jinlong Li, Runsheng Xu, Xinyu Liu, Jin Ma, Zicheng Chi, Jiaqi Ma, and Hongkai Yu. Learning for Vehicle-to-Vehicle Cooperative Perception Under Lossy Communication. *IEEE Transactions on Intelligent Vehicles*, 8(4):2650–2660, 2023. https://doi.org/10.1109/TIV.2023.3260040.

[27] Chuheng Wei, Guoyuan Wu, and Matthew J. Barth. Cooperative Perception for Automated Driving: A Survey of Algorithms, Applications, and Future Directions. *Proceedings of the IEEE*, pages 1–27, 2025. https://doi.org/10.1109/JPROC.2025.3608874.

[28] Lei Zhang, Binglu Wang, Yongqiang Zhao, Yuan Yuan, Tianfei Zhou, and Zhijun Li. Collaborative Multimodal Fusion Network for Multiagent Perception. *IEEE Transactions on Cybernetics*, 55(1):486–498, 2025. https://doi.org/10.1109/TCYB.2024.3491756.

[29] Yang Zhou, Cai Yang, Ping Wang, Chao Wang, Xinhong Wang, and Nguyen Ngoc Van. ViT-FuseNet: Multimodal Fusion of Vision Transformer for Vehicle-Infrastructure Cooperative Perception. *IEEE Access*, 12:31640–31651, 2024. https://doi.org/10.1109/ACCESS.2024.3368404.

[30] Junyang He, Xiaoheng Deng, Jinsong Gui, Tao Zhang, and Xiangjian He. MDNet: Multimodal Cooperative Perception via Spatial Alignment of Modal Decision-Making. *IEEE Internet of Things Journal*, 12(11):16142–16154, 2025. https://doi.org/10.1109/JIOT.2025.3531145.

[31] Hongbo Yin, Daxin Tian, Chunmian Lin, Xuting Duan, Jianshan Zhou, Dezong Zhao, and Dongpu Cao. V2VFormer++: Multi-Modal Vehicle-to-Vehicle Cooperative Perception via Global-Local Transformer. *IEEE Transactions on Intelligent Transportation Systems*, 25(2):2153–2166, 2024. https://doi.org/10.1109/TITS.2023.3314919.

[32] Hui Zhang, Guiyang Luo, Yuanzhouhan Cao, Yi Jin, and Yidong Li. Multi-Modal Virtual-Real Fusion based Transformer for Collaborative Perception. In *2022 IEEE 13th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pages 1–6, 2022. https://doi.org/10.1109/PAAP56126.2022.10010640.

[33] Lantao Li, Kang Yang, Wenqi Zhang, Xiaoxue Wang, and Chen Sun. RG-Attn: Radian Glue Attention for Multi-modal Multi-agent Cooperative Perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1763–1772, 2025. https://doi.org/10.48550/arXiv.2501.1680.

[34] Bin Lu, Xinyu Xiao, Changzhou Zhang, Yang Zhou, Zhiyu Xiang, Hangguan Shan, and Eryun Liu. Privacy-Preserving V2X Collaborative Perception Integrating Unknown Collaborators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 5802–5810, 2025.

[35] Hanwen Jiang, Shijun Zhou, Konglin Zhu, Artur Andrzejak, and Yi Gong. A Multimodal Collaborative Perception Framework in Challenging Environments. In *2025 9th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC)*, pages 62–66, 2025. https://doi.org/10.1109/IC-NIDC67200.2025.11390536.

[36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial intelligence and statistics*, pages 1273–1282. Pmlr, 2017. https://doi.org/10.48550/arXiv.1602.05629.

[37] Peter Kairouz and H Brendan McMahan. Advances and Open Problems in Federated Learning. *Foundations and trends in machine learning*, 14(1-2):1–210, 2021. https://doi.org/10.48550/arXiv.1912.04977.

[38] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020. https://doi.org/10.48550/arXiv.1910.06378.

[39] Priyanka Mary Mammen. Federated Learning: Opportunities and Challenges. *arXiv preprint arXiv:2101.05428*, 2021. https://doi.org/10.48550/arXiv.2101.05428.

[40] Zhenrong Zhang, Jianan Liu, Xi Zhou, Tao Huang, Qing-Long Han, Jingxin Liu, and Hongbin Liu. On the Federated Learning Framework for Cooperative Perception. *IEEE Robotics and Automation Letters*, 9(11):9423–9430, 2024. https://doi.org/10.1109/LRA.2024.3457374.

[41] Mohamed K. Abdel-Aziz, Cristina Perfecto, Sumudu Samarakoon, Mehdi Bennis, and Walid Saad. Vehicular Cooperative Perception Through Action Branching and Federated Reinforcement Learning. *IEEE Transactions on Communications*, 70(2):891–903, 2022. https://doi.org/10.1109/TCOMM.2021.3126650.

[42] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. CryptoDL: Deep Neural Networks over Encrypted Data. *arXiv preprint arXiv:1711.05189*, 2017. https://doi.org/10.48550/arXiv.1711.05189.

[43] Runhua Xu, James B.D. Joshi, and Chao Li. Cryptonn: Training neural networks over encrypted data. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1199–1209, 2019. https://doi.org/10.1109/ICDCS.2019.00121.

[44] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine Learning Classification over Encrypted Data. Cryptology ePrint Archive, Paper 2014/331, 2014. https://doi.org/10.14722/ndss.2015.23241.

[45] Jungho Moon, Dongwoo Yoo, Xiaoqian Jiang, and Miran Kim. THOR: Secure Transformer Inference with Homomorphic Encryption. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 3765–3779, 2025. https://doi.org/10.1145/3719027.3765150.

[46] Dongwoo Kim and Cyril Guyot. Optimized Privacy-Preserving CNN Inference With Fully Homomorphic Encryption. *IEEE Transactions on Information Forensics and Security*, 18:2175–2187, 2023. https://doi.org/110.1109/TIFS.2023.3263631.

[47] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[48] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, 3 edition, 2020.

[49] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[50] Calibrating uncertainties in object localization task.

[51] Zining Wang, Di Feng, Yiyang Zhou, Lars Rosenbaum, Fabian Timm, Klaus Dietmayer, Masayoshi Tomizuka, and Wei Zhan. Inferring Spatial Uncertainty in Object Detection. *arXiv preprint arXiv:2003.03644*, 2020.

[52] Apostol Vassilev, Munawar Hasan, Edward Griffor, Honglan Jin, Pavel Piliptchak, Mahima Arora, and Thoshitha Gamage. On the Assessment of Sensitivity of Autonomous Vehicle Perception. *arXiv preprint arXiv:2602.00314*, 2026. https://doi.org/10.48550/arXiv.2602.00314.

[53] VTTI. Virginia Tech Transportation Institute.

[54] Ultralytics. Ultralytics YOLO. Accessed: 2026.

[55] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. 2019. https://doi.org/10.48550/arXiv.1812.05784.

[56] OpenPCDet Development Team. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet, 2020. Accessed: 2026.