Probabilistic Orchestrator for Indeterministic Multi-Agents in Real-Time Environment

Arkady Bovshover , Andrei Kojukhov , Ilya Levin *

Posted Date: 2 February 2026

doi: 10.20944/preprints202602.0030.v1

Keywords: multi-agent systems; probabilistic orchestration; uncertainty modeling; sensor fusion; real-time perception; adaptive learning

 doi:10.20944/preprints202602.0030.v1

Article

Probabilistic Orchestrator for Indeterministic Multi-Agents in Real-Time Environment

Arkady Bovshover [1], Andrei Kojukhov [2] and Ilya Levin [3,*]

Holon Institute of Technology

* Correspondence: levini@hit.ac.il; Tel.: +972-505771389

Abstract

Multi-agent perception systems must operate under fundamental asymmetries: some agents provide fast but unreliable observations, while others deliver higher-quality evidence with delay and uncertain correspondence. Traditional deterministic orchestration and rule-based fusion struggle to manage these trade-offs, often producing brittle or unstable behaviour. We introduce a probabilistic orchestration framework that treats coordination as an epistemic generation problem —constructing and updating belief states under uncertainty—rather than a selection problem. Instead of committing to a single agent's output, the orchestrator constructs a belief state that explicitly represents uncertainty, evidential provenance, and temporal relevance. Decisions are produced through latency-aware, association-weighted fusion, and uncertainty itself becomes a first-class signal governing action, deferral, and learning. Crucially, the orchestrator enables controlled teacher–student adaptation: high-confidence, well-associated stationary observations are gated into a feedback loop that improves ego perception over time while mitigating error amplification. We demonstrate the approach on an infrastructure-assisted dual-camera obstacle-recognition task. Experimental results show improved robustness to distance, occlusion, and delayed evidence compared to ego-only and deterministic orchestration baselines. By operationalizing orchestration as epistemic generation, this work provides a unifying framework for robust decision-making and safe adaptation in multi-agent systems, with implications that extend beyond perception to agentic and generative AI architectures.

Keywords: multi-agent systems; probabilistic orchestration; uncertainty modeling; sensor fusion; real-time perception; adaptive learning

1. Introduction

Consider an autonomous vehicle approaching an intersection. Its onboard camera detects a potential obstacle, but the object is distant and partially hidden behind another car. Meanwhile, a roadside camera with a better vantage point has already recognized the same obstacle with high confidence—but this information arrives with a 200-millisecond delay due to network transmission. Which source should the vehicle trust? How should it combine these conflicting signals into a safe decision? This scenario illustrates a fundamental challenge in multi-agent perception systems: different sensors offer complementary strengths, yet they also differ in accuracy, timing, and reliability. The onboard camera provides immediate feedback but struggles with distant or occluded objects. The infrastructure sensor offers superior recognition but introduces latency. Neither source alone is sufficient; the real challenge lies in orchestrating them effectively.

Current approaches to this problem typically rely on simple rules: trust the source with higher confidence, or always prefer the faster sensor for time-critical decisions. While such rule-based methods are easy to implement, they break down when conditions become complex. What happens when both sources report conflicting results with similar confidence? Or when the "reliable" sensor occasionally produces overconfident errors? In practice, these rigid approaches lead to unstable behavior—flickering detections, missed obstacles, and false alarms that undermine system reliability.

 doi:10.20944/preprints202602.0030.v1

This paper proposes a different approach: a probabilistic orchestrator that treats each sensor's output not as a definitive answer but as uncertain evidence to be weighed and combined. The key insight is that effective multi-source fusion requires reasoning about three interrelated factors: how confident is each source in its prediction, how likely is it that both sources are observing the same object, and how much should we discount information that arrives late? By modelling these factors explicitly, the orchestrator can make principled decisions even when individual sources disagree or fail. Beyond fusion, the orchestrator enables something more ambitious: continuous self-improvement. When the infrastructure sensor provides a high-confidence detection that the vehicle's own camera missed or misclassified, this becomes a learning opportunity. The orchestrator can use such cases as training signals to gradually improve the onboard model, but only when it is confident that the teaching signal is reliable and correctly matched to the same physical object.

We demonstrate and evaluate this approach using a dual-camera setup for road obstacle detection. A vehicle-mounted camera handles real-time perception, while stationary roadside cameras provide delayed but often more accurate detections. Our experiments show that probabilistic orchestration improves the detection of distant and occluded obstacles compared to either source alone, while the learning mechanism allows the onboard model to improve over time without manual retraining.

Importantly, while this work is motivated by a concrete perception problem, its scope is broader than sensor fusion. We view orchestration as the generation of an epistemic state rather than the selection of a single "correct" output. In this sense, the orchestrator does not merely aggregate predictions but constructs a temporally grounded belief that explicitly represents uncertainty, evidential provenance, and the conditions under which decisions should be deferred, revised, or used for learning. This perspective aligns the proposed framework with recent developments in agentic and generative AI, where systems are increasingly expected to reason under incomplete information, manage heterogeneous agents, and transform partial evidence into structured, actionable interpretations rather than deterministic answers.

Four core contributions are put forward in this paper:

1. Probabilistic orchestration for heterogeneous agents. We introduce an orchestration framework that fuses fast ego perception with delayed, high-confidence external evidence using calibrated confidence, association reliability, and latency-aware policies.
2. Uncertainty-aware decision policy. We define an interpretable fusion mechanism that produces an uncertainty-aware output distribution, reducing brittleness compared to deterministic selection and enabling stable behaviour under delayed evidence.
3. Controlled teacher–student adaptation. We integrate pseudo-labelling and knowledge distillation into the orchestration layer through explicit gating, enabling continual improvement while reducing the risk of error amplification.
4. Dual-camera obstacle recognition case study. We demonstrate the approach in an infrastructure-assisted perception setting and analyse robustness under distance, occlusion, and latency variability.

Figure 1 provides an overview of the proposed multi-agent architecture, showing how the probabilistic orchestrator fuses ego and delayed stationary evidence and how high-confidence teacher outputs are gated to support controlled student adaptation.

 doi:10.20944/preprints202602.0030.v1

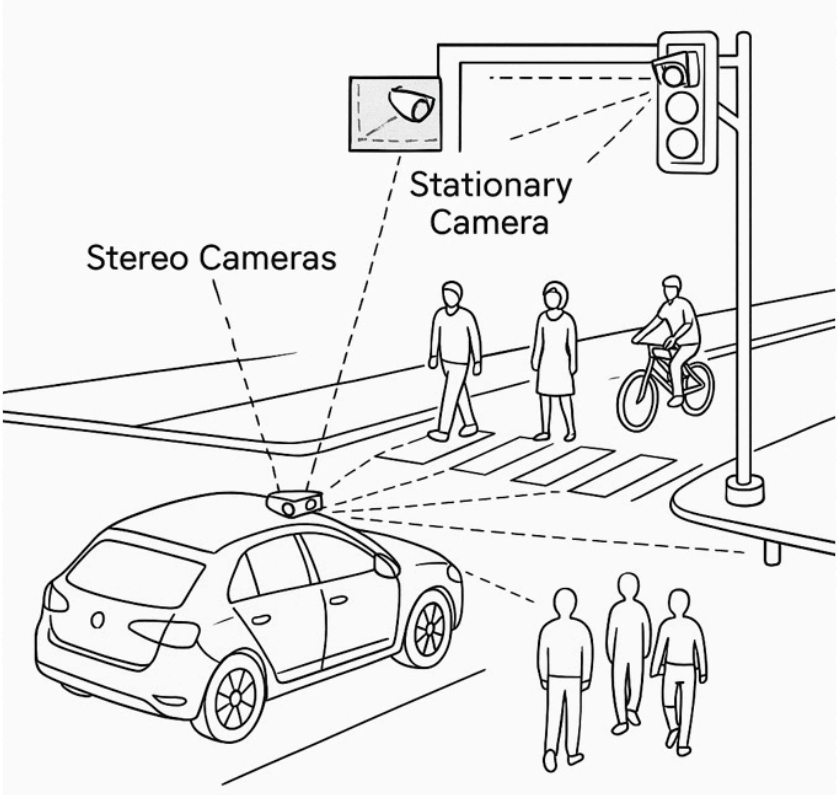

Figure 1. Dual-Camera Use Case.

In our prior work [1], we extended Rodney Brooks' subsumption architecture by replacing its traditional priority-based arbiter with a Generative Orchestrator designed for semantic mediation. Instead of arbitrating among competing agent outputs, the orchestrator interprets and integrates heterogeneous agent representations into coherent, human-interpretable actions. Specifically, it synthesizes reflexive, model-based, goal-oriented, and utility-driven epistemic perspectives using large language models to produce unified decisions. That study established orchestration as a process of semantic integration rather than hierarchical control and validated the approach through an autonomous driving case study.

The present work advances this line of research by moving from architectural demonstration to operational realization. Specifically, we address how generative orchestration can function under real-world constraints such as calibrated uncertainty, latency-aware decision deadlines, and association ambiguity across heterogeneous sensing agents.

Given the scope of the study, the paper is organized as follows. Section 2 reviews related work; Section 3 formalizes the problem and introduces the dual-camera use case; Section 4 presents the orchestrator architecture and decision policy; Section 5 describes the controlled adaptation loop; Section 6 details the implementation; Section 7 evaluates the proposed approach; Section 8 discusses generalization and broader implications; and Section 9 concludes the paper.

2. Related Works

This work engages with several intersecting research threads that address complementary aspects of complex, real-time decision systems. Specifically, it relates to prior work on multi-agent orchestration for heterogeneous decision sources, uncertainty-aware fusion under imperfect confidence and latency constraints, teacher–student learning through pseudo-labeling and knowledge distillation, and multi-camera or infrastructure-assisted perception in autonomous driving. While each of these lines of research addresses a subset of the challenges considered here, they are typically studied in isolation or combined through fixed, rule-based mechanisms. In contrast,

 doi:10.20944/preprints202602.0030.v1

our approach integrates these perspectives within a unified probabilistic orchestration framework, enabling semantic mediation among heterogeneous agents. The following sections review these research directions and clarify how our method departs from and extends existing approaches.

2.1. Multi-Agent Orchestration and Decision Coordination

Multi-agent systems are widely adopted in domains where complex tasks are decomposed into specialized components (agents) and coordinated through a controller or distributed policies [2,3]. In many deployed pipelines, orchestration remains largely deterministic—implemented as rule-based gating, hard thresholds, or state-machine logic that selects one agent's output or fuses outputs using fixed heuristics [4]. These mechanisms are attractive because they are interpretable, easy to implement, and straightforward to debug.

However, deterministic orchestration becomes brittle when agents have different error modes and timing characteristics. In real-time perception stacks, conflicts can arise when one detector produces a confident but incorrect prediction under occlusion or low resolution, while another (delayed) source produces a more accurate hypothesis. If the orchestrator treats confidence as directly comparable or treats association as binary, late-arriving signals can cause abrupt decision overrides, oscillations (decision churn), and inconsistent labels that reduce trust in the system [5].

More recent research has explored learned mechanisms for coordinating multiple decision components, including adaptive routing and decision optimization across heterogeneous models [6,7]. While these approaches move beyond fixed priority schemes, they typically assume well-calibrated confidence estimates or abstract away timing effects, which limits their suitability for real-time, safety-critical perception pipelines. In contrast, our work explicitly models both uncertainty and latency within a probabilistic orchestration framework, incorporating association reliability directly into the decision policy.

2.2. Uncertainty-Aware Fusion and Confidence Calibration

Fusion methods combine information from multiple sources to improve robustness beyond what any single sensor or model can provide. Classical approaches include Bayesian fusion, Kalman filtering, particle filtering, and probabilistic graphical models that update belief states based on uncertain observations [8]. In tracking and state estimation, uncertainty is explicit and naturally represented as part of the system state.

In deep learning–based perception systems, uncertainty is often handled implicitly, and confidence scores produced by neural networks are frequently miscalibrated, particularly under distribution shift, long-range sensing, occlusion, and rare conditions [9,10]. While calibration techniques and uncertainty estimation methods aim to improve the reliability of predicted probabilities [11–13], calibrated confidence alone is insufficient when evidence sources differ in latency and association reliability. In multi-view perception, late fusion and confidence-based selection are commonly used, but these approaches typically rely on fixed thresholds and neglect explicit timing effects. Our approach extends existing fusion strategies by treating uncertainty, association reliability, and latency as jointly interacting factors within a probabilistic orchestration framework.

2.3. Teacher–Student Learning, Pseudo-Labelling, and Knowledge Distillation

Teacher–student learning provides an effective mechanism for transferring knowledge from a high-capacity model (the teacher) to a smaller or faster model (the student), supporting efficient deployment and often improving generalization [15,16]. Knowledge distillation typically trains the student to match the teacher's soft probability outputs, sometimes in combination with ground-truth supervision [15]. Related approaches such as pseudo-labeling and self-training use teacher predictions as supervisory signals to expand training data in semi-supervised settings, reducing annotation cost and enabling adaptation to new domains [17–19].

 doi:10.20944/preprints202602.0030.v1

A central challenge in these approaches is that naïve pseudo-labeling can reinforce teacher errors, introduce confirmation bias, and amplify drift, particularly under distribution shift or when the teacher's predictions are uncertain [20]. Consequently, effective teacher–student systems rely on careful sample selection, confidence thresholds, and validation strategies. In many perception pipelines, however, teacher–student learning remains an offline training procedure, decoupled from operational inference. In contrast, our work integrates teacher–student adaptation directly into the orchestration layer, where training signals are gated based on association reliability, teacher confidence, and timing constraints, enabling controlled and continual adaptation while limiting error propagation.

2.4. Multi-Camera and Infrastructure-Assisted Perception for Autonomous Driving

Multi-camera perception is a central component of modern autonomous driving systems, encompassing camera-only pipelines, multi-view 3D reconstruction, temporal modeling, and cross-modal sensor fusion [21,22]. By combining multiple viewpoints, such systems mitigate occlusion and improve long-range recognition, but they must also address challenges related to calibration, synchronization, and cross-view association [23].

Infrastructure-assisted perception extends multi-camera approaches by incorporating stationary sensors such as roadside cameras or smart-city infrastructure to support autonomous vehicles [24,25]. These sensors can provide advantageous viewpoints, earlier visibility of occluded objects, and improved recognition of distant obstacles. However, infrastructure-based signals introduce additional constraints, including network latency and jitter, uncertain cross-view association, partial coverage, and heterogeneity in sensor quality and model performance [26]. Many existing methods treat infrastructure inputs as auxiliary signals or fuse them using deterministic rules. In contrast, our work formulates infrastructure-assisted perception as a multi-agent orchestration problem, in which delayed external evidence is integrated probabilistically based on both association reliability and timing.

2.5. Positioning of the Proposed Approach

Existing research in multi-agent systems has extensively addressed coordination mechanisms, probabilistic information fusion, and teacher–student learning paradigms, typically treating these aspects as separate design concerns. In many deployed architectures, orchestration remains largely deterministic, while learning processes are performed offline or are weakly coupled to real-time decision making.

The present work departs from this separation by introducing a probabilistic orchestration layer that integrates heterogeneous agent outputs under uncertainty and latency constraints, while simultaneously regulating teacher–student adaptation through a gated feedback mechanism. The dual-camera obstacle recognition case study demonstrates how uncertainty-aware orchestration can jointly support real-time decision fusion and controlled continual learning within a unified multi-agent perception architecture.

2.6. Implications for Generative Multi-Agent Systems

Contemporary research on generative multi-agent systems increasingly investigates architectures that coordinate heterogeneous components—such as lightweight and high-capacity language models, retrieval modules, verifiers, and planners—into coherent decision workflows [27,28]. While these systems extend classical multi-agent coordination into the generative domain, orchestration is often implemented using largely deterministic strategies, including fixed tool invocation sequences or unconditional reliance on a single high-capacity model.

In practice, such strategies are fragile under conditions characteristic of generative systems, including prompt variability, hallucinations, partial tool failures, and non-stationary task distributions [29]. In response, several recent approaches have proposed adaptive routing and

 doi:10.20944/preprints202602.0030.v1

selection mechanisms that dynamically determine which models or tools to invoke, when to verify intermediate outputs, and when to defer decisions or solicit additional evidence [30].

From this perspective, our framework suggests a more general orchestration principle applicable beyond perception. Agent outputs are treated as uncertain evidence rather than definitive decisions, and orchestration is framed as a probabilistic process that accounts for reliability and timing. In this analogy, a high-quality model or verified tool functions as a teacher, lower-latency or lower-capacity models act as students, and auxiliary verification components provide reliability signals analogous to association confidence. The orchestrator integrates these signals to perform uncertainty-aware routing and aggregation, and can selectively trigger adaptation using high-confidence feedback. Although our experiments focus on perception, this view positions probabilistic orchestration as a unifying design pattern for both perception-driven and generative multi-agent systems.

3. Use Case and Problem Formulation: Dual-Camera Road Obstacle Recognition

This section introduces the motivating use case and formalizes the multi-agent perception setting addressed in this paper. We focus on road obstacle recognition for autonomous driving, where an ego vehicle must detect obstacles under long-range, low-contrast, and partially occluded conditions. While state-of-the-art object detectors achieve strong performance under favourable conditions, their reliability degrades significantly for small or distant objects, motion blur, strong illumination changes, and occlusions caused by vehicles or infrastructure. At the same time, stationary roadside cameras can provide advantageous viewpoints and improved object visibility, but their outputs arrive with non-negligible latency due to communication and processing delays. This combination creates an ideal setting for studying orchestration across heterogeneous agents with different latency and confidence profiles.

In this work, we model obstacle recognition as an object detection task, where each obstacle hypothesis consists of a class label (or a class probability distribution) and an associated bounding box. The goal of orchestration is to generate a detection output under real-time constraints while maintaining robustness to partial, noisy, or conflicting sensory evidence, and to explicitly represent uncertainty when the available information is insufficient for confident classification or localization.

3.1. Problem Motivation and Failure Modes in Ego-Only Perception

Ego-camera perception provides low-latency situational awareness and is a common input modality for real-time driving systems. However, ego-only detection exhibits several persistent failure modes:

1. Long-range degradation: Distant obstacles occupy few pixels, reducing discriminative features and increasing confusion with background patterns.
2. Occlusion and partial visibility: Obstacles may be partially blocked by other vehicles or road geometry, resulting in fragmented detections or missed objects.
3. Illumination changes: Shadows, glare, and nighttime conditions can distort appearance and reduce detector confidence.
4. Motion blur: Rapid ego motion and vibration degrade image quality, especially at higher speeds.
5. Domain shift: Weather, camera settings, and geographic differences lead to distribution shift and degraded generalization.

These failure modes are especially critical for safety because they can cause missed detections of obstacles that require early reaction. Prior work has explored multi-sensor fusion and multi-view perception to mitigate these effects [21,22], but infrastructure-assisted perception introduces additional orchestration challenges, including latency, uncertain association, and heterogeneous model reliability.

3.2. Infrastructure-Assisted Perception Scenario

 doi:10.20944/preprints202602.0030.v1

This work considers an infrastructure-assisted perception scenario in which ego sensing is augmented by a stationary roadside camera. The setup, illustrated in Figure 1, assumes a dual-camera configuration combining an ego-mounted sensor with an external, fixed-viewpoint camera.

Stationary cameras provide several complementary advantages over ego-mounted sensors: (i) stable mounting that reduces motion blur, (ii) broader or elevated viewpoints, and (iii) earlier visibility of obstacles that may be occluded from the ego perspective. As a result, stationary sensors can act as high-confidence but temporally delayed evidence sources, supplying obstacle hypotheses that may refine or correct ego perception.

Despite these advantages, the integration of stationary camera outputs introduces several constraints that are central to the orchestration problem:

1. Latency and jitter: Sensor outputs are delayed by acquisition, processing, and network transmission, and may arrive with variable delay.
2. Association uncertainty: A stationary camera's detection must be matched to the ego camera's observation of the same physical object across different perspectives.
3. Partial coverage: Not all environments are instrumented; stationary evidence may be intermittent.
4. Heterogeneous failure modes: Stationary detectors can also fail due to weather, occlusion, or miscalibration, and their confidence may not be directly comparable to ego confidence.

These constraints motivate an orchestration layer that can reason probabilistically about the reliability and relevance of external evidence under timing uncertainty.

3.3. System Overview and Multi-Agent Perception Pipeline

We consider a pipeline with four primary agents that produce complementary outputs:

1. An ego detection agent that performs real-time object detection from the vehicle's forward-facing camera.
2. A geometric localization agent that maps detection outputs into a shared spatial representation (e.g., using homography or camera calibration).
3. A stationary detection agent that detects obstacles from a roadside camera and communicates predictions to the vehicle.
4. A cross-view association agent that establishes correspondence between the ego and the stationary detections.

These agents feed a probabilistic orchestrator that selects or fuses decisions under uncertainty and latency constraints. The orchestrator produces the final obstacle label and confidence score and may trigger a learning loop to improve the ego detector over time.

Table 1 summarizes the roles of the individual agents involved in this perception pipeline, including their primary functions, typical latency characteristics, confidence profiles, and output representations.

Agent	**Function**	**Typical Latency**	**Typical Confidence**	**Output**
A1: Ego Detector	Real-time obstacle detection on ego camera	Low	Medium (drops at distance/occlusion)	Bounding boxes + class probabilities
A2: Ego Localization	Map ego detections to ground plane / shared coordinates	Low	Medium	Localized boxes / estimated position

 doi:10.20944/preprints202602.0030.v1

A3: Stationary Detector	Obstacle detection from road-side camera	Medium	High (better long-range)	Bounding boxes + class probabilities
A4: Cross-View Association	Match detections across ego and stationary views	Medium	High when confident match exists	Association score + matched object IDs

The orchestrator consumes the heterogeneous outputs of these agents, including class probability distributions, spatial bounding boxes, localization signals, cross-view association scores, and latency metadata. Based on this information, it produces a consolidated detection result comprising a final object class label (or distribution), a selected or fused bounding box, an explicit uncertainty estimate, and a decision trace that supports explainability and audit.

3.4. Cross-View Association and Shared Coordinate Representation

A critical challenge in the dual-camera setting is determining whether an obstacle detected by the stationary camera corresponds to an obstacle detected by the ego camera. This association problem is complicated by viewpoint changes, partial visibility, and temporal misalignment due to latency.

We adopt a two-stage association approach:

1. Geometric alignment: Ego detections are projected into a shared ground-plane or world coordinate frame using a homography transformation derived from camera calibration and planar road assumptions (details in Appendix A). Stationary detections are similarly mapped to the shared frame.
2. Probabilistic matching: Candidate matches are evaluated based on:
    - spatial proximity in the shared frame (e.g., Euclidean distance between projected centers),
    - temporal consistency accounting for latency and ego motion (penalizing large),
    - semantic agreement between predicted classes (e.g., similarity of class distributions).

The association agent outputs a match confidence score used by the orchestrator. By explicitly representing association confidence, the orchestrator can treat stationary evidence as strong only when cross-view correspondence is reliable.

3.5. Learning Setup: Offline Proof-of-Concept with Feedback-Driven Adaptation

This study focuses on an offline proof-of-concept pipeline that reflects practical constraints of infrastructure-assisted perception. The ego detector runs in real time and produces predictions for each incoming frame. Stationary camera outputs arrive with delay and serve as high-confidence teacher signals when association confidence is sufficiently high.

The orchestrator uses these teacher signals to:

- curate pseudo-labelled training pairs,
- fine-tune the ego detector on examples where it is weak (e.g., distant obstacles),
- distil stationary soft labels into the ego model,
- and deploy updated ego model versions under controlled conditions.

Crucially, we do not perform uncontrolled online updates: learning is gated and validated before deployment to reduce the risk of error amplification and unsafe drift.

3.6. Scope and Assumptions

To keep the problem well-defined and operationally realistic, we adopt the following scope assumptions:

 doi:10.20944/preprints202602.0030.v1

In scope:

- Vision-based detection/classification of road obstacles using ego and stationary cameras.
- Confidence-aware and latency-aware orchestration across heterogeneous agents.
- Cross-view association and reliability scoring.
- Offline/nearline adaptation through pseudo-labeling and knowledge distillation.

Out of scope:

- End-to-end motion planning and control.
- Multi-modal fusion with lidar/radar (future work).
- High-definition mapping and full SLAM integration.
- Online model updates without validation (unsafe in practice).

We assume that cameras are calibrated and that a planar road approximation is acceptable for homography-based projection within the region of interest. We also assume that stationary camera connectivity is available, but subject to variable latency.

Taken together, the dual-camera setting, agent roles, uncertainty sources, and scope limitations introduced in this section define a perception problem characterized by heterogeneous evidence, temporal misalignment, and non-uniform reliability. Addressing this setting requires an orchestration mechanism capable of integrating delayed and uncertain information while preserving real-time responsiveness. The next section introduces the probabilistic orchestrator architecture and decision policy developed to meet these requirements.

4. Probabilistic Orchestrator Architecture and Decision Policy

This section presents the probabilistic orchestrator that coordinates heterogeneous agents under uncertainty and latency constraints. The orchestrator integrates predictions from ego and stationary detection agents, geometric localization, and cross-view association into an uncertainty-aware decision state. Unlike deterministic orchestration (e.g., hard thresholds or fixed precedence rules), the orchestrator produces calibrated confidence estimates, preserves uncertainty when evidence is weak, and supports a gated feedback loop for improving agents over time (Section 5).

As was described in the introduction, we meet challenging problems related to weak objects' recognition when only cars' cameras are leveraged for inferencing /object classification. When camera (arranged on car) is capturing video far enough (more than60-80m) then size of an object on the road (in pixels size) small enough (less than 16-32 pixels for standard 640x640 image resolution used for object classification/segmentation tasks). For so smalls object's sizes almost impossible to classify objects (cars, humans, animals, etc.) with good enough accuracy. To improve this accuracy additional (look at articles: [24,25]) stationary camera arranged near the pedestrian crossing or near the traffic lights.

Multi-view fusion with a moving car camera + a fixed intersection camera can boost recall and reduce false positives. The key is to put both cameras' detections into the same space/time, then associate them robustly. The stationary camera is capturing images, sending a compressed video stream (as h264/265 format 1-2 Mbit/s to the client (car's client app running on car's CPU + GPU) using a technology client-server communication based on 5G 1Gbit/s communication architecture. Of course, 5G adds latency and jittering, but below we'll discuss how to make time alignment of two agents: car's images and object recognition (the student agent) and object recognition by are coming from stationary camera (the Teacher agent).

The appropriate agents (Techer -Student) and matching agent that (using ground -plane fusion algorithms explained in the Appendix) provides ID tracking of Student and Techer sequence recognized objects. If there are (in the current small time slot ~100ms) two objects (still not classified) from both Teacher and Student cameras, algorithm should chouse appropriate object and its bounding box from Teacher object's queue and corresponding Student's object should be deprecated from student's queue.

 doi:10.20944/preprints202602.0030.v1

This info is getting central our agent is named Orchestrator Decision Maker. The goal of this central agent is to manage appropriate Student, Teacher and Matcher agents making object's classification as optimal as possible. Also. This Orchestrator agent is managing the fine-tuning learning process of popular data sets that are used for object's classification with autonomic cars and self-driving assistance (YOLO, Ultralytics models etc.)

The main target of our publication is the Orchestrator Decision Maker agent. Below we provide the technical description, innovative algorithms and model that allow to create an Orchestrator Decision Maker agent. We suppose that this Orchestrator concept can be used for wide class of Neuron network models, for example, Reinforce Teacher-Student models for training of robot walking and robot navigation (Reinforce Teacher-Student models are used very widely at Nvidia Issac Robot simulators). For robot walking, robot vision, and robot navigation the challenge tasks often involve two different agents are solving the same problem but with different natures: for example, first agent is calculating the next robot step based on a deterministic algorithm (stereometric vision/ 3D homography analysis etc.), but the second agent is based on Reinforce Teacher-Student network (probabilistic model) and it provides the next robot step solution in the same time. Our model of the Orchestrator agent can decide in this catch situation. We are supposed to describe the usage of the Orchestrator agent for Robotics navigation in our future article.

Resuming, our Orchestrator agent can be used as a decision maker when there are some agents that are solving the same tasks at the same time and are providing different decisions.

4.1. Orchestrator Design Goals

The orchestrator is designed to satisfy four operational goals:

- Robustness under uncertainty: reconcile conflicting predictions and preserve uncertainty when evidence is insufficient.
- Latency-aware decision making: produce real-time outputs even when higher-confidence evidence is delayed.
- Association-aware fusion: incorporate cross-view correspondence reliability as first-class evidence.
- Controlled learning enablement: generate decision traces and high-confidence supervision signals without amplifying errors.

These goals reflect the central tension in infrastructure-assisted perception: responsiveness requires fast ego decisions, while reliability improves when external evidence becomes available—provided that the association is correct and confidence is calibrated.

4.2. Teacher-Student Orchestrator Design, Flow, and Realization

4.2.1. From Student Agent to Orchestrator Agent

The Orchestrator is constructed by augmenting standard Student components, not by replacing them. The original Student agent continues to operate independently, ensuring robustness and real-time responsiveness. The Orchestrator, meanwhile, focuses on learning improvements rather than direct control.

The transformation from a standard Student agent to an Orchestrator agent involves improving the following three components:

1. Fine-tuning and transfer learning for YOLO
2. Knowledge Distillation (KD) and KD integration
3. Extended loss functions for KD-enhanced YOLO training

Each component is described in detail below.

4.2.2. Component I: Fine-Tuning and Transfer Learning for YOLO

 doi:10.20944/preprints202602.0030.v1

The Orchestrator training process begins from a pretrained YOLO checkpoint, which can be either:

- Either the public one: yolov8n.pt, yolov8s.pt, yolo11n.pt, etc.
- Or your own car-camera model: carcam_yolov8n.pt.

Then we keep those weights and train a bit more on our new dataset (real + pseudo), where real is the standard YOLO data set and "pseudo" is obtained from Teacher camera.

Transfer learning is applied such that:

- We don't re-learn basic things like "what is an edge, what is a car shape".
- We just adapt to:
  - our camera viewpoint (car + intersection geometry),
  - Tiny/far humans & cars
  - Optionally freeze early layers (so you mainly adapt high-level features & head).

4.2.3. Component II: Knowledge Distillation and Cross-View Integration

Instead of using only the hard 0/1 labels from the teacher, more advanced setups use teacher logits (full probability over classes) and a special loss to match student predictions. Out-of-the-box Ultralytics doesn't expose a simple "teacher model" argument, so this would require a custom training loop, so we'll treat this as a next step.

What is Knowledge Distillation actually is?

In normal pseudo-labelling, we do:

- Teacher says: "This is PERSON."
- Student uses a hard label: class=person, 1 vs 0.

In knowledge distillation, the teacher provides soft labels:

- Teacher says:
  - class probabilities:

P_teacher = [person: 0.82, car: 0.17, dog: 0.01, ...]

  - bounding box regression confidences

The student is trained to **match the entire probability distribution**, not just the winning class.

This contains **much richer information**, especially for **ambiguous cases** (small, far, partially occluded objects).

KD was invented for training small models from big models, and it works VERY well for your case:

- **Stationary camera** = stronger teacher (better angle, stable, less motion blur).
- **Car camera** = weaker student (far objects, motion blur, etc.).

How does KD fits dual-camera scenario?

The pipeline earlier does tracking + cross-camera association:

Car camera → low confidence → candidate sample

Stationary camera → same track → high confidence → teacher logits available

We pair each ego detection with its corresponding teacher detection (via the track ID + ground-plane fusion).

This is **paired, cross-view distillation.**

1.2.1. Component III: KD Loss Functions (Improved YOLO Loss Function)

YOLO internal loss has 3 components:

- **Classification loss**
- **Box regression loss**
- **Objectness loss**

KD adds new terms for the Orchestrator Loss function:

 doi:10.20944/preprints202602.0030.v1

1. Classification KD Loss

Formula of KD Loss component:

$$\mathcal{L}_{\text{KD-cls}} = \boldsymbol{\tau^2} \cdot \boldsymbol{KL}(\text{softmax}(\boldsymbol{z_t}/\boldsymbol{\tau}), \text{softmax}(\boldsymbol{z_s}/\boldsymbol{\tau}))$$

What the symbols mean:

Symbol	Meaning
z_t	Teacher logits (vector of un-normalized scores, one per class)
z_s	Student logits
$\text{softmax}(z/\tau)$	Probability distribution over classes at temperature τ
$KL(P, Q)$	KL divergence between distributions $P$and Q
τ	Temperature (usually 2–10),

where

KL divergence meaning:

$$\boldsymbol{KL(P \,||\, Q)} = \sum_{\boldsymbol{i}} \boldsymbol{P_i} \log \frac{\boldsymbol{P_i}}{\boldsymbol{Q_i}}$$

2. Bounding Box Distillation Loss

Formula bounding box Loss component:

$$\mathcal{L}_{\text{KD-bbox}} = \lambda_{\text{box}} \cdot SmoothL1(B_s, B_t)$$

where:

- B_s= student bounding box prediction
- B_t= teacher bounding box prediction

Boxes are usually in the format:

$$[x_c, y_c, w, h]$$

or YOLO's internal parameterization.

Meaning of bounding box distillation

The stationary teacher has a **better viewpoint** → more accurate boxes.

Thus:

- Student (car camera) learns **better geometry**, even when its own camera view is poor
- Helps especially on:
  - far pedestrians
  - side-view objects
  - partially occluded objects
  - small objects (< 16 px)

3. Objectness Distillation Loss

Formula of Object Loss component:

$$\mathcal{L}_{\text{KD-obj}} = \lambda_{\text{obj}} \cdot (p^s_{\text{obj}} - p^t_{\text{obj}})^2$$

Why is objectness KD important?

Especially in far-distance scenes:

- Ego (car) camera gets **more false positives** (reflections, poles, lights…)
- Stationary camera sees objects more clearly → gives reliable objectness

Teaching students to copy the teacher's objectness means:

- Fewer false positives
- Stronger positive signals for far but real objects
- Better small object recall

 doi:10.20944/preprints202602.0030.v1

Full Orchestrator Loss and how KD Loss components will be integrated with YOLO's normal loss

Final Orchestrator student loss:

$$L_{\text{total}} = L_{\text{YOLO}}^{cls} + L_{\text{YOLO}}^{box} + L_{\text{YOLO}}^{obj} + \alpha L_{\text{KD-cls}} + \beta L_{\text{KD-bbox}} + \gamma L_{\text{KD-obj}}$$

Purpose of each component:

Component	Purpose
YOLO classification	Student still learns hard labels
YOLO bounding-box	Student learns from its own view
YOLO objectness	Student filters own FP/TP
KD-cls	Learns teacher's "soft opinion"
KD- bounding -box	Learns teacher's geometry
KD-obj	Learns teacher's sense of objectness

4.3. Decision Objective (Expected Utility Formulation)

The policy above can be viewed as optimizing expected utility under latency constraints. We define:

- a utility function $U(\hat{y}, y)$capturing the benefit/cost of decisions (e.g., reward for correct obstacle classification, penalty for missed obstacles, penalty for false alarms),
- a latency cost $C(\Delta)$.

The orchestrator chooses a decision distribution $p_o$that maximizes:

$$\max_{p_o} \ \mathbb{E}_{y\sim p_o}[U(\hat{y}, y)] - \beta C(\Delta)$$

subject to:

- real-time constraint $\Delta \leq T_{\max}$,
- association reliability constraint $p_m \geq \theta_m$for fusion.

In practice, our implementation uses the interpretable policy in Section 4.4, which approximates this objective by controlling the mixing weight αvia match confidence and latency.

4.4. Bounding Box Selection and Fusion

The orchestrator also produces a final bounding box $\hat{b}^t$. We consider two strategies:

1. Selection (default): if stationary evidence is used ($\alpha > 0$), select the box from the higher-confidence source; otherwise, use ego box:

$$\hat{b}^t = \begin{cases} b_s^{t-\Delta} & \text{if } \alpha \geq \theta_b \\ b_e^t & \text{otherwise} \end{cases}$$

2. Fusion (optional): compute a weighted average in a shared coordinate frame:

$$\hat{b}^t = (1-\alpha) b_e^t + \alpha\, \Pi^{-1}(\Pi(b_s^{t-\Delta}))$$

where Π denotes projection into a shared representation (e.g., ground plane). In our proof-of-concept, we adopt selection for stability and simplicity; fusion can be enabled when calibration is strong.

4.5. Confidence Calibration and Uncertainty Measures

To enable probabilistic orchestration, agent confidence must be comparable and interpretable. We apply calibration to each agent's output distribution:

$$p_e'(y) = \text{Calibrate}(p_e(y)), p_s'(y) = \text{Calibrate}(p_s(y))$$

 doi:10.20944/preprints202602.0030.v1

using standard calibration procedures such as temperature scaling on validation data [10]. In addition to maximum probability, we compute uncertainty metrics that support gating and learning:

- **Entropy**: $H(p_o)$(high entropy implies uncertainty).
- **Margin**: difference between top-1 and top-2 probabilities.
- **Consistency**: agreement between ego and stationary top labels.

These quantities are recorded in the decision trace $\tau^t$and later used to select high-confidence examples for teacher–student training (Section 5).

4.6. Orchestrator as a Gating Mechanism for Controlled Learning

A key feature of the orchestrator is that it produces not only the final decision but also gating signals that determine whether a given example is suitable for learning. Specifically, an example at time $t$becomes a candidate training instance when:

- $p_m \geq \theta_m$(high association confidence),
- $\max_y p_s'(y) \geq \theta_s$(teacher confidence threshold),
- $\Delta \leq T_{\text{learn}}$(acceptable delay for labelling), and
- $\hat{u}^t \leq \theta_u$(low uncertainty).

These gates prevent low-quality teacher outputs from corrupting the student model and enable a controlled teacher–student learning loop.

4.7. Algorithm Summary

Algorithm 1: Probabilistic Orchestration (per time step t) (see Figure 2)

1. Receive ego prediction $(p_e, b_e^t)$and localization ℓ_e^t.
2. If a stationary prediction is available, receive $(p_s, b_s^{t-\Delta})$an association score p_m.
3. Calibrate distributions p_e', p_s'.
4. Compute mixing weight $\alpha = p_m \cdot \exp(-\lambda\Delta)$.
5. If $\Delta > T_{\max}$or $p_m < \theta_m$, set $p_o = p_e'$; else set $p_o = (1-\alpha)p_e' + \alpha p_s'$.
6. Output $\hat{y}^t = \arg\max_y p_o(y)$, uncertainty $\hat{u}^t = H(p_o)$, and box $\hat{b}^t$(selection or fusion).
   Store trace τ^t. If learning gates are satisfied, enqueue pseudo-label for Section 5.

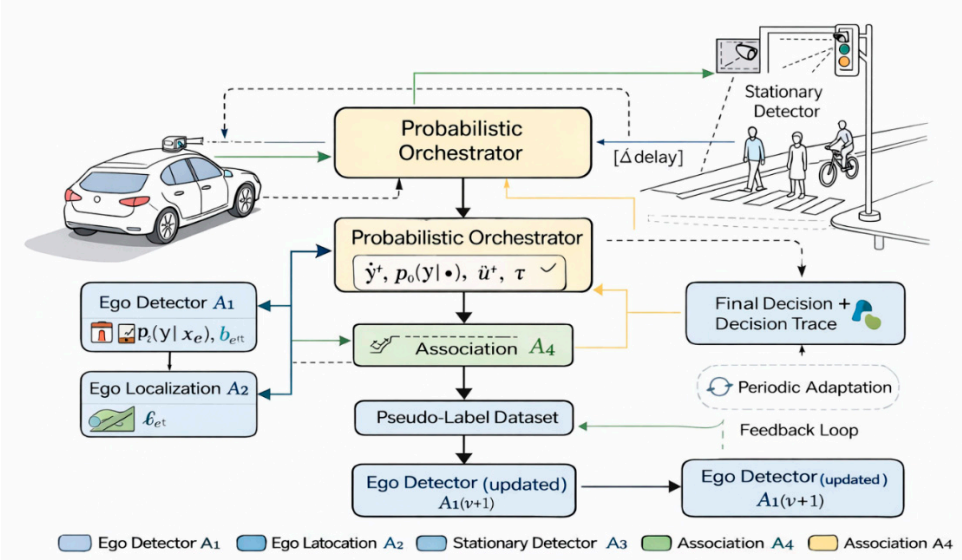

Figure 2. Orchestrator-managed teacher–student algorithm flow.

Algorithm 1 and the associated orchestration flow in Figure 2 operationalize the probabilistic framework developed throughout this section. Together, they specify how heterogeneous agent outputs—ego predictions, delayed infrastructure evidence, localization signals, and association confidence—are calibrated, weighted, and fused under explicit latency and reliability constraints.

 doi:10.20944/preprints202602.0030.v1

Beyond producing uncertainty-aware obstacle detections, the orchestrator exposes internal decision traces and gating signals that regulate when external evidence may be used to support learning. These mechanisms form the basis for the feedback-driven adaptation process described in the next section.

5. Feedback-Driven Adaptation: Teacher–Student Fine-Tuning and Knowledge Distillation

While the probabilistic orchestrator improves decision quality at inference time by fusing heterogeneous agent outputs (Section 4), its longer-term value is in enabling controlled improvement of weaker agents. In our use case, the ego detector (Agent A1) exhibits degraded performance for distant and partially occluded obstacles, whereas the stationary detector (Agent A3) typically provides higher-confidence predictions but with a delay. This asymmetry naturally motivates a teacher–student adaptation mechanism, in which stationary predictions serve as supervision to improve the ego model—provided that (i) association is reliable and (ii) teacher confidence is high.

A central challenge is that naïve self-training can amplify errors: teacher failures, mismatched associations, or delayed evidence can generate noisy pseudo-labels that degrade performance over time. We therefore design an orchestrator-gated learning loop that uses explicit quality and safety gates to decide when learning is allowed. This section describes pseudo-label curation, student fine-tuning, knowledge distillation, and controlled deployment of updated models.

5.1. Learning Loop Overview

The adaptation loop is triggered and managed by the orchestrator, which produces both fused decisions and decision traces. High-confidence teacher outputs are gated by association and uncertainty constraints before being used for student fine-tuning and knowledge distillation, as illustrated in Figure 3.

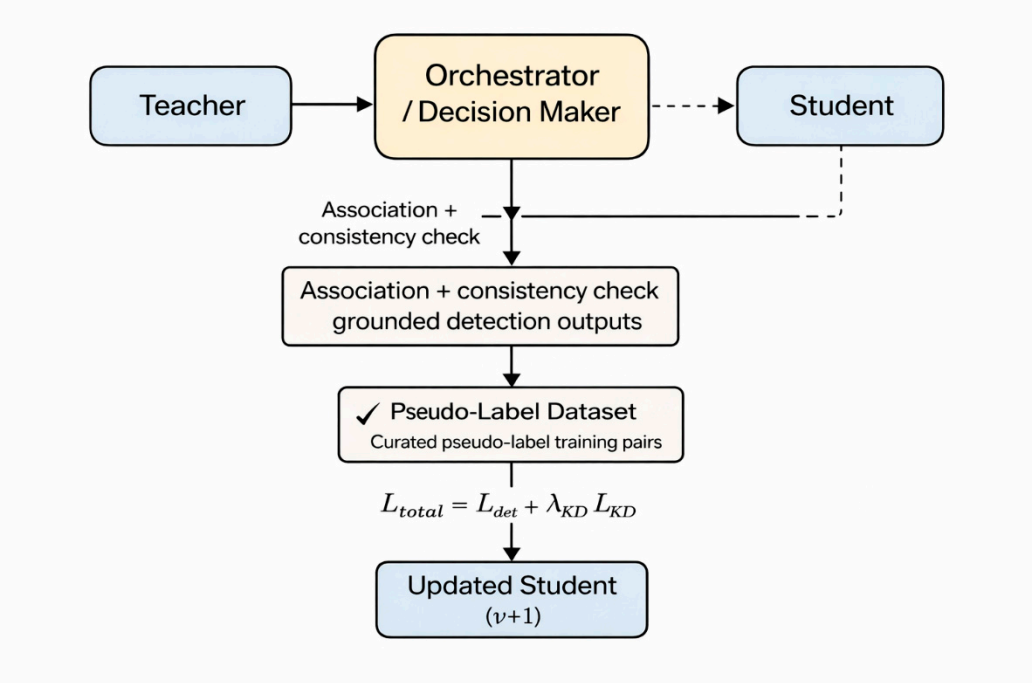

Figure 3. Orchestrator-managed teacher–student adaptation.

The learning loop consists of four stages:

1. Candidate collection: identify time steps where stationary predictions are available and association confidence is high.

 doi:10.20944/preprints202602.0030.v1

2. Pseudo-label gating: filter candidates using teacher confidence, latency constraints, and uncertainty thresholds to reduce label noise.
3. Student adaptation: fine-tune the ego detector using pseudo-labels, optionally combined with a small amount of ground-truth labeled data.
4. Controlled deployment: evaluate and register the updated model; deploy only if it improves key metrics without increasing false alarms beyond an acceptable limit.

Crucially, we do not perform uncontrolled online learning: model updates are performed offline or nearline and are deployed only after validation and gating checks.

5.2. Pseudo-Label Curation and Quality Gates

A key challenge in teacher–student learning is preventing confirmation bias and error propagation when the teacher is wrong or when the association is incorrect. We therefore use the orchestrator's gating signals (Section 4.8) to curate pseudo-labels.

Let t denote a time step, and let:

- p_m^t be association confidence from A4,
- $p_s'^t(y)$ be calibrated teacher probability distribution from A3,
- Δ^t be stationary latency,
- $\hat{u}^t$ be orchestrator uncertainty.

A training candidate is accepted if:

$$p_m^t \geq \theta_m \wedge \max_y p_s'^t(y) \geq \theta_s \wedge \Delta^t \leq T_{\text{learn}} \wedge \hat{u}^t \leq \theta_u$$

where:

- θ_m: association reliability threshold,
- θ_s: teacher confidence threshold,
- T_{learn}: maximum delay allowed for supervision,
- θ_u: uncertainty threshold to avoid ambiguous learning signals.

Optional consistency checks. To further reduce noise, we apply:

- spatial consistency: projected positions must be within radius r in the shared coordinate frame,
- temporal consistency: association must persist for k consecutive frames.

Accepted samples from the pseudo-labeled dataset:

$$\mathcal{D}_{\text{PL}} = \{(x_e^t, \tilde{y}^t, \tilde{b}^t, p_s'^t)\}$$

where x_e^t is the ego image, $\tilde{y}^t$ and $\tilde{b}^t$ are the teacher-derived label and box, and $p_s'^t$ is the teacher's soft label distribution retained for distillation.

5.3. Student Model and Training Objective

The student is the ego detector f_θ (Agent A1). The teacher is the stationary detector g_ϕ (Agent A3). The goal is to improve ego detection quality on difficult cases (distance/occlusion), while preserving responsiveness and controlling false alarms.

We optimize a combined objective:

$$\mathcal{L} = \mathcal{L}_{det} + \lambda_{KD}\mathcal{L}_{KD}$$

where:

- $\mathcal{L}_{det}$ is the standard detector loss under pseudo-label supervision,
- $\mathcal{L}_{KD}$ transfers teacher soft labels to the student,
- λ_{KD} controls distillation strength.

5.4. Detection Loss (Pseudo-Label Supervision)

For a pseudo-labeled training instance $(x_e^t, \tilde{y}^t, \tilde{b}^t)$, the detector loss is:

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{cls}}(\hat{y}, \tilde{y}) + \lambda_{\text{box}}\mathcal{L}_{\text{box}}(\hat{b}, \tilde{b}) + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}}$$

where:

- $\tilde{y}$ and $\tilde{b}$ are pseudo-label class and box from the teacher,

 doi:10.20944/preprints202602.0030.v1

- $\hat{y}$ and $\hat{b}$ are student outputs,
- $\mathcal{L}_{\text{cls}}$ is typically cross-entropy or focal loss,
- $\mathcal{L}_{\text{box}}$ is IoU/GIoU/DIoU loss,
- $\mathcal{L}_{\text{obj}}$ models objectness confidence.

Because pseudo-labels may contain residual noise, we optionally apply confidence-weighted supervision:

$$\mathcal{L}_{\text{cls}} \leftarrow w^t \cdot \mathcal{L}_{\text{cls}} \text{ where } w^t = \max_y p_s'^t(y)$$

so that high-confidence teacher predictions contribute more strongly to training.

5.5. Knowledge Distillation Loss (Soft Label Transfer)

To transfer richer information than hard pseudo-labels, we distill the teacher's soft distribution into the student output distribution. We use temperature-scaled distillation [14]:

$$\mathcal{L}_{\text{KD}} = T^2 \cdot \text{KL}(\text{Softmax}(z_s/T) \parallel \text{Softmax}(z_e/T))$$

where:

- z_s and z_e are teacher and student logits,
- T is the distillation temperature,
- $\text{KL}(\cdot)$ is the Kullback–Leibler divergence.

The distillation term encourages the student to match teacher uncertainty, which is especially valuable for borderline cases (e.g., distant obstacles) where the teacher provides a smoother probability structure than hard labels alone.

5.6. Triggering and Scheduling Adaptation

To avoid unsafe rapid adaptation, learning runs in an offline or nearline regime. The orchestrator accumulates pseudo-labels until either:

- the pseudo-label set size exceeds a minimum N_{min}, or
- a time window W elapses.

Adaptation is triggered only when:

- pseudo-label yield and diversity are sufficient (e.g., enough long-range examples),
- mean teacher confidence exceeds a threshold,
- and (optionally) drift indicators increase (e.g., rising uncertainty or reconstruction error for hard examples).

This scheduling reduces overfitting to short bursts and limits the effect of transient noise.

5.7. Controlled Deployment, Rollback, and Safety Checks

A key principle of controlled reinforcement learning is that updates must be validated before deployment. Each updated student model version $f_{\theta'}$ is evaluated against:

- a held-out validation set (if available),
- a curated test set representing difficult cases (distant/occluded obstacles),
- and operational constraints (latency, false alarm tolerance).

We adopt a gating rule:

$$\text{Deploy } f_{\theta'} \text{ only if } \Delta\text{Recall} \geq \epsilon_r \ \wedge \ \Delta\text{FPR} \leq \epsilon_f \ \wedge \ \Delta\text{Latency} \leq \epsilon_\ell$$

If these conditions are not met, the update is rejected or retained for further tuning. Versioning enables rollback to f_θ if post-deployment monitoring detects performance regressions.

5.8. Relation to Reinforcement Learning and "Judging-as-Reward."

Although adaptation is implemented via supervised and distillation losses, the orchestration-controlled learning loop can be interpreted through a reinforcement learning lens. The orchestrator

 doi:10.20944/preprints202602.0030.v1

acts as a judging agent that determines whether an outcome is acceptable for learning. In this interpretation:

- pseudo-label acceptance corresponds to positive reinforcement signals,
- pseudo-label rejection corresponds to negative feedback (no learning update),
- deployment gating corresponds to policy constraints and safe rollout.

This framing clarifies how the proposed approach differs from uncontrolled online RL: learning is always gated, validated, and rollback enabled.

Taken together, the mechanisms introduced in this section describe a controlled adaptation process in which high-confidence stationary detections serve as teacher signals for the ego detector, while association confidence, uncertainty estimates, and latency constraints regulate pseudo-label selection, training triggers, and deployment. This design enables incremental improvement without compromising runtime safety. The next section details the concrete implementation of the learning loop, association mechanisms, and runtime orchestration pipeline.

6. Implementation

This section describes the implementation of the dual-camera multi-agent pipeline, including agent modules, orchestration runtime, cross-view association, pseudo-label logging, and the controlled adaptation workflow. The implementation is designed as an offline/nearline proof-of-concept that reflects practical constraints of infrastructure-assisted perception, including variable latency, imperfect association, and the need for safe model update gating.

6.1. System Components and Runtime Pipeline

The pipeline is implemented in Python as a modular multi-agent system, where each agent exposes a consistent interface:

- predict(input) → output
- metadata() → confidence/latency statistics
- trace() → evidence and internal signals used for auditing

Figure 4 summarizes the testing (inference) pipeline, highlighting the real-time ego detection flow and the integration of delayed stationary evidence through the probabilistic orchestrator.

The orchestrator consumes agent outputs and produces:

$$\mathcal{D}^t = (\hat{y}^t, p_o(y \mid \mathcal{E}^t), \hat{b}^t, \hat{u}^t, \tau^t)$$

as defined in Section 4.2, where τ^t is a decision trace.

Runtime timing model.

- Ego inference runs synchronously, frame-by-frame, at time index t.
- Stationary camera predictions arrive asynchronously with delay Δ (and optional jitter).
- The orchestrator performs event-driven fusion whenever new stationary evidence becomes available within the fusion window $[t - T_{\max}, t]$.

This execution model ensures that ego decisions can be produced without waiting for delayed evidence, while still benefiting from stationary corrections when they arrive in time.

 doi:10.20944/preprints202602.0030.v1

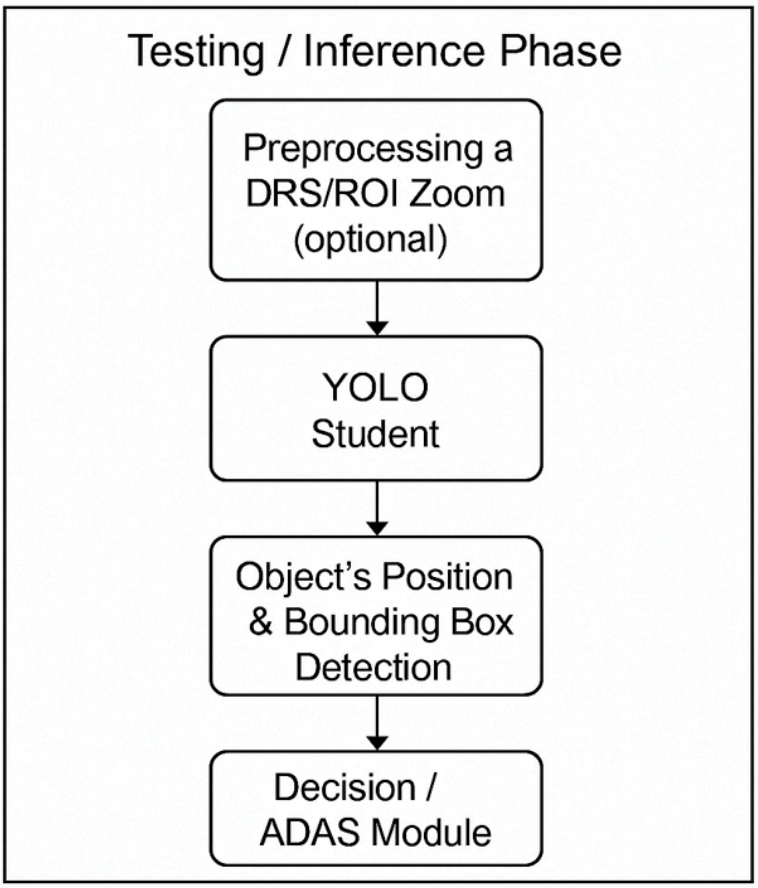

Figure 4. Inference pipeline (runtime).

6.2. Ego Detector (Student Model)

Model. The ego detector (Agent A1) is implemented using a YOLO-family object detector (e.g., YOLOv5/YOLOv8) configured for obstacle detection. The model is initialized from a publicly available pre-trained checkpoint and adapted to the target domain through fine-tuning using pseudo-labels curated by the orchestrator.

Input preprocessing.

- Resize to $[W \times H]$(e.g., 640×640).
- Normalize pixel values to $[0, 1]$.
- Optional augmentation for robustness (color jitter, blur, random crop).

Output. Per frame t, the ego detector outputs:
bounding boxes b_e^t,
class distribution $p_e(y \mid x_e^t)$,
objectness score $p(\text{obj})$.

Latency constraints. The model is optimized for low-latency inference on the ego compute platform (GPU/edge). In the proof-of-concept, ego inference runs at $[X]$ FPS on $[GPU/CPUspecification]$.

6.3. Stationary Detector (Teacher Agent A3)

The stationary detector uses the same model family but is treated as a higher-confidence teacher due to:

- stable viewpoint,
- reduced motion blur,
- improved long-range visibility.

It outputs:

 doi:10.20944/preprints202602.0030.v1

- bounding boxes $b_s^{t-\Delta}$,
- class distribution $p_s(y \mid x_s^{t-\Delta})$,
- objectness scores.

Communication payload. Stationary predictions are transmitted along with metadata:

- timestamp $t - \Delta$,
- estimated delay Δ,
- calibration parameters or confidence summary statistics.

In the proof-of-concept, stationary delay is modeled either as:

- fixed delay, or
- stochastic delay with jitter: $\Delta \sim \mathcal{N}(\mu, \sigma^2)$, truncated to positive values.

6.4. Confidence Calibration (Required for Fusion)

Because fusion assumes comparable probabilities, we calibrate ego and stationary outputs:

$$p_e'(y) = \text{Calibrate}(p_e(y)), p_s'(y) = \text{Calibrate}(p_s(y))$$

Calibration is performed using temperature scaling on a validation set (Section 4.6). The calibration module is implemented as a lightweight post-processing step applied to agent logits before orchestration.

6.5. Geometric Localization (Agent A2) and Shared Coordinate Frame

To enable cross-view association, detections are mapped into a shared representation using planar homography projection.

Ego projection:

$$\ell_e^t = \Pi_e(b_e^t; H_e)$$

Stationary projection:

$$\ell_s^{t-\Delta} = \Pi_s(b_s^{t-\Delta}; H_s)$$

where H_e and H_s are homography matrices computed from camera calibration. Projection is implemented using OpenCV routines. Because planar homography is approximate (non-planar road surfaces, calibration drift), the projected localization is used as a coarse alignment signal, not as a deterministic match.

6.6. Cross-View Association (Agent A4)

The association agent links ego and stationary detections that correspond to the same physical object. Candidate pairs are generated by spatial proximity in the shared coordinate frame:

$$d(\ell_e^t, \ell_s^{t-\Delta}) \leq r$$

For each candidate pair, association confidence is computed:

$$p_m^t = p(\text{match} \mid d, \Delta, \text{class similarity})$$

In practice, we compute a score from three factors:

1. Spatial score: Gaussian penalty over distance d.
2. Temporal score: exponential penalty in delay Δ.
3. Semantic score: class agreement (e.g., cosine similarity between class distributions or KL divergence).

The final match probability p_m^t is obtained via a logistic transform or normalized scoring function and passed to the orchestrator for fusion and learning gating.

6.7. Orchestrator Runtime and Logging

The orchestrator implements the fusion policy described in Section 4.4. For each time step t, it:

1. collects ego predictions,
2. checks for matched stationary evidence within window constraints,
3. computes $\alpha^t = p_m^t \exp(-\lambda\Delta)$,

 doi:10.20944/preprints202602.0030.v1

4. outputs fused belief $p_o(y)$, label $\hat{y}^t$, and uncertainty $\hat{u}^t$.

Decision trace logging. For each decision, we log:

1. agent contributions (ego-only vs fused),
2. calibrated distributions $p_e^{'}$, $p_s^{'}$,
3. match confidence p_m^t,
4. delay Δ,
5. fusion weight α^t,
6. uncertainty $\hat{u}^t$,
7. threshold values and gating decisions.

This trace supports explainability, debugging, and auditing of both inference and learning.

6.8. Pseudo-Label Logging and Dataset Construction

Pseudo-label candidates are logged whenever teacher evidence is available. The orchestrator applies the acceptance gates from Section 5.2:

$$p_m^t \geq \theta_m, \max_y p_s'(y) \geq \theta_s, \Delta \leq T_{\text{learn}}, \hat{u}^t \leq \theta_u$$

Accepted pseudo-labels are stored as structured entries:

$$\left(x_e^t,\ \tilde{y}^t,\ \tilde{b}^t,\ p_s'^t,\ p_m^t,\ \Delta^t\right)$$

These samples are exported into a detector-compatible format (YOLO/COCO-style) with extended metadata fields. This enables reproducible training and later ablation analysis (e.g., sensitivity to $\theta_m$or T_{learn}).

6.9. Student Fine-Tuning and Distillation Workflow

Adaptation is triggered when a minimum dataset size $N_{\min}$is reached or after a fixed collection window W. Fine-tuning runs offline/nearline for a bounded number of epochs and produces a new student checkpoint $f_{\theta'}$.

The student is trained using:

$$\mathcal{L} = \mathcal{L}_{det} + \lambda_{KD}\mathcal{L}_{KD}$$

as defined in Section 5.3–5.5. The training workflow includes:

- checkpoint versioning,
- validation-based early stopping,
- and a deployment gate requiring recall improvement while bounding false positive increase and latency regression (Section 5.7).

6.10. Configuration and Reproducibility

All runtime and learning parameters are stored in a single configuration file, including:

- model checkpoint paths,
- calibration temperature values,
- thresholds $\theta_m, \theta_s, \theta_u$,
- timing windows $T_{\max}, T_{\text{learn}}$,
- latency decay λ,
- training triggers $N_{\min}, W$,
- distillation parameters T, λ_{KD}.

This configuration-driven design enables reproducibility of experiments and clear reporting of all orchestration and learning settings.

Taken together, the components described in this section constitute a modular dual-camera, multi-agent perception pipeline with asynchronous infrastructure evidence, probabilistic association, latency-aware orchestration, and gated teacher–student adaptation. Decision traces and pseudo-labels are logged to support auditing, rollback, and controlled model updates. This implementation serves as the basis for the experimental evaluation presented in the next section, which compares the

 doi:10.20944/preprints202602.0030.v1

proposed approach against ego-only and deterministic orchestration baselines under varying distance, occlusion, and latency conditions.

7. Evaluation

This section evaluates the proposed probabilistic orchestrator and the feedback-driven adaptation loop on the dual-camera obstacle recognition task. We compare against ego-only perception and deterministic orchestration baselines and analyze robustness under long-range detection, occlusion, and variable latency. We further quantify the benefit of the teacher–student adaptation loop through ablation experiments.

7.1. Evaluation Goals

We design the evaluation to answer four questions:

Q1. Does probabilistic orchestration improve decision quality over ego-only detection under long-range and occlusion conditions?

Q2. Does probabilistic orchestration outperform deterministic fusion/state-machine baselines under delayed stationary evidence?

Q3. Does the teacher–student learning loop improve ego detector performance over time without increasing false alarms?

Q4. How sensitive is performance to latency and association uncertainty?

7.2. Dataset and Experimental Setup

We evaluate on a dataset of synchronized ego and stationary camera sequences containing obstacle events under diverse conditions (distance variation, illumination changes, partial occlusions). The dataset is split into:

Training set: used for initial ego fine-tuning (if applicable) and for collecting pseudo-labels.

Validation set: used for confidence calibration, threshold tuning, and gating parameter selection.

Test set: held out for final evaluation and latency sensitivity analysis.

Where ground truth labels are unavailable for all sequences, we rely on partial annotation for key events and use stationary detections as weak supervision for evaluation of long-range improvements (clearly stated as a limitation).

7.3. Baselines

We compare against the following baselines:

B1: Ego-only detector

The ego detector (A1) without stationary assistance.

This baseline reflects common camera-only real-time perception.

B2: Deterministic orchestrator (rule-based / state machine)

A deterministic fusion policy that selects stationary predictions when available, otherwise uses ego predictions, using hard thresholds and fixed precedence rules. This represents common system engineering practice.

B3: Probabilistic orchestrator (ours, no learning)

The probabilistic orchestrator described in Section 4, with calibrated confidence, association-aware weighting, and latency-aware mixing, but without the adaptation loop.

B4: Probabilistic orchestrator + teacher–student learning (ours full)

B3 plus the gated pseudo-labeling and knowledge distillation adaptation loop described in Section 5.

 doi:10.20944/preprints202602.0030.v1

7.4. Experimental Protocol

7.4.1. Calibration and Threshold Selection

Confidence calibration (e.g., temperature scaling) is performed on the validation split. Gating thresholds $\theta_m, \theta_s, \theta_u$ and timing windows $T_{\max}, T_{\text{learn}}$ are tuned to balance recall and false positives while ensuring stable behavior.

7.4.2. Latency Modelling

Stationary evidence delay Δ is evaluated under:

- fixed delay conditions (e.g., 100 ms, 300 ms, 500 ms),
- jittered delay conditions (e.g., $\Delta \sim \mathcal{N}(\mu, \sigma^2)$).

This allows analysis of robustness under realistic communication uncertainty.

7.4.3. Association Reliability

To evaluate sensitivity to association errors, we vary the association threshold θ_m and optionally inject association noise (false matches) at controlled rates.

7.4.4. Adaptation Schedule

For B4, pseudo-labels are collected in windows and used to fine-tune the student model periodically. Each model update is validated before being compared on the held-out test set. We report both:

- performance of the initial student model (before adaptation),
- performance of the adapted student model (after one or more adaptation cycles).

7.5. Results

7.5.1. Overall Performance Comparison

We first compare overall performance across baselines B1–B4 on the full test set.

Expected outcome (qualitative):

- B2 improves recall over B1 but increases false positives under association errors and delayed evidence.
- B3 improves recall over B1 while controlling false positives due to match-weighted, latency-aware fusion.
- B4 yields additional gains in long-range recall by improving the ego detector itself.

This graph (Figure 5) demonstrates new Orchestrator Teacher-Student based on distillation technology compared to standard deterministic Orchestrator based on state machine. The results were obtained for training process described above. New advanced Orchestrator demonstrates advantages by all categories: Correct detection, Under/Over fitting and Missed detection.

 doi:10.20944/preprints202602.0030.v1

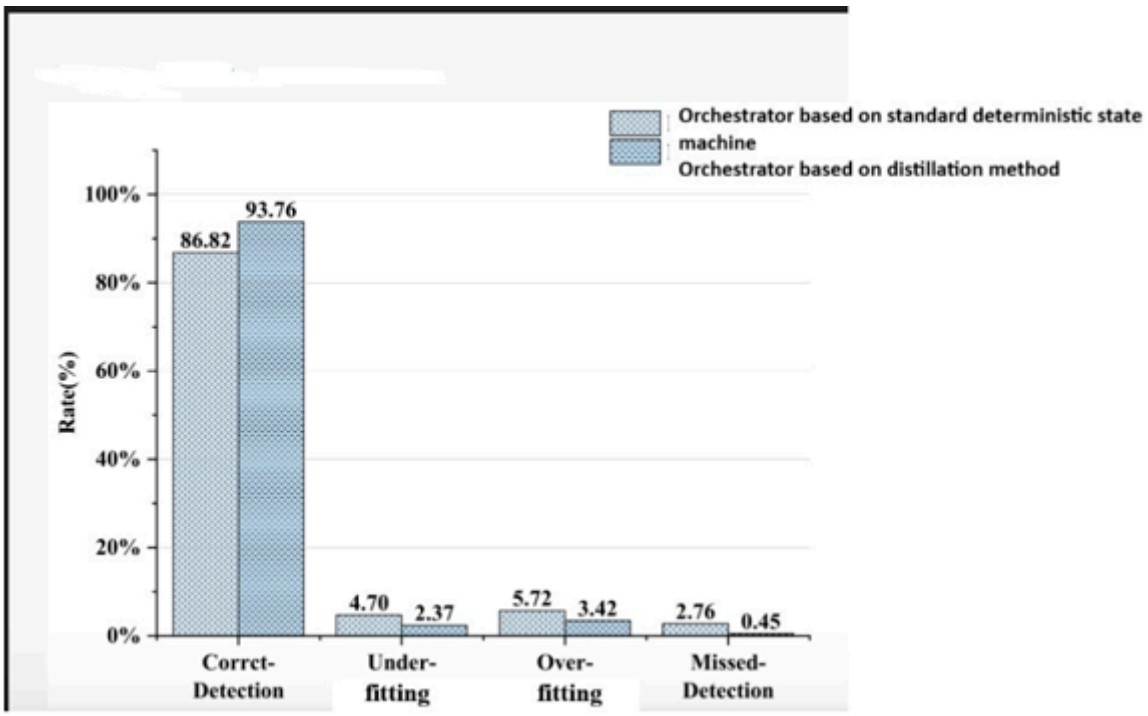

Figure 5. Comparative Performance.

7.5.2. Effect of Latency and Jitter

We evaluate how performance changes as stationary delay Δ increases. Deterministic selection policies often degrade sharply when delayed evidence arrives too late and causes decision instability. The probabilistic orchestrator mitigates this via exponential latency decay in the mixing weight.

Expected outcome:

- B2 may show oscillations and increased decision churn at moderate delays.
- B3 remains stable and degrades gracefully as Δincreases.
- B4 remains comparable to B3 at inference time, and also reduces reliance on stationary evidence through improved ego performance.

7.5.3. Effect of Association Gating

We evaluate sensitivity to association threshold θ_m. Low thresholds increase fusion coverage but risk incorporating incorrect stationary evidence; high thresholds reduce coverage and may reduce gains.

Expected outcome:

- B3 exhibits a favourable trade-off where moderate $\theta_m$maximizes benefit.
- B2 is more sensitive to $\theta_m$due to hard switching.

7.5.4. Ablation: Benefit of Distillation and Pseudo-Label Gating

We isolate learning contributions through ablations:

- Ablation A: B4 without KD (fine-tuning only).
- Ablation B: B4 without pseudo-label gating (accept all).
- Ablation C: B4 without latency constraints in pseudo-label selection.
- Ablation D: B4 without association thresholding.

Expected outcome:

KD provides smoother improvement and reduces overfitting to pseudo-label noise.

Gating is critical: removing it increases false positives and may degrade performance.

Latency constraints reduce label mismatch and stabilize adaptation.

 doi:10.20944/preprints202602.0030.v1

7.6. Discussion of Findings

The results support three conclusions:

Probabilistic orchestration improves robustness compared to ego-only and deterministic fusion, especially for long-range and occluded obstacles.

Latency-aware mixing stabilizes decisions and avoids abrupt behavior under delayed evidence.

Gated teacher–student adaptation improves ego performance over time while controlling false alarms, reducing reliance on infrastructure signals.

7.7. Limitations

We highlight limitations of the current study:

The proof-of-concept assumes calibrated cameras and approximate planar road geometry.

Evaluation depends on stationary evidence availability; infrastructure coverage may be limited in real deployments.

Pseudo-labelling can still introduce noise in rare failure cases; further safety mechanisms (e.g., human-in-the-loop validation) could be integrated.

Online adaptation is not explored; we focus on offline/nearline updates for safety.

Taken together with the experimental results presented in this section, these limitations delineate the current scope of applicability of the proposed approach. Within this scope, the evaluation demonstrates that probabilistic orchestration improves decision quality under uncertainty and latency constraints, and that gated teacher–student adaptation further strengthens ego perception. The next section discusses broader implications of this work and considers how probabilistic orchestration may generalize to other multi-agent settings, including multi-agent generative AI workflows.

8. Discussion

This work argues that multi-agent perception systems should be designed as uncertainty-aware orchestration problems rather than deterministic routing problems. The dual-camera obstacle recognition use case highlights a recurring pattern in real-world agent ecosystems: some agents are fast but unreliable in edge cases (ego detector), while others are slow but more accurate (stationary camera). Effective systems must therefore (i) decide under incomplete evidence and (ii) improve future decision-making by learning from high-quality feedback. In this section, we discuss how the proposed probabilistic orchestrator addresses these requirements, how it generalizes beyond the case study, and what limitations remain.

8.1. Why Deterministic Orchestration Fails in Heterogeneous Agent Systems

Our results reinforce that deterministic orchestration—hard thresholds, rule-based switching, or state-machine decision logic—tends to be brittle in heterogeneous multi-agent pipelines. The failure modes are structural:

1. Conflict without uncertainty: deterministic policies require a single "winner," even when evidence is ambiguous, producing unstable or overconfident decisions.
2. Confidence mismatch: deep models often produce miscalibrated probabilities, so using raw confidence as a switching criterion can amplify errors.
3. Association fragility: cross-view association is inherently uncertain; deterministic fusion treats association as binary, leading to incorrect overrides and false alarms.
4. Latency blindness: delayed evidence may arrive after action is taken; deterministic policies may oscillate when late signals are applied without timing-aware logic.

These issues are not unique to perception; they occur whenever multi-agent outputs conflict under time pressure. The key design takeaway is that orchestration must represent uncertainty explicitly and incorporate both evidence reliability and timing.

 doi:10.20944/preprints202602.0030.v1

8.2. The Contribution of Probabilistic Orchestration

The proposed orchestrator contributes a practical approach to uncertainty-aware decision coordination. Rather than replacing engineering design with a black-box policy, the orchestrator introduces a structured probabilistic decision state that fuses:

- calibrated confidence from heterogeneous agents,
- association reliability as a match probability, and
- latency-aware weighting to limit the influence of delayed evidence.

The result is a soft gating mechanism that degrades gracefully when evidence is delayed, uncertain, or mismatched. Importantly, the orchestrator preserves uncertainty, enabling downstream safety logic and operator intervention rather than forcing premature commitment. From a systems perspective, this is a central advantage: uncertainty is not an error—it is an operational signal that prevents unsafe automation.

8.3. Orchestration as a Learning Manager: Safe Feedback vs Error Amplification

A second contribution of this work is the explicit integration of learning into orchestration. Many real-world pipelines separate inference (runtime decisions) from learning (offline training), resulting in slow adaptation and brittle behavior under drift. In contrast, our orchestrator produces decision traces and gating signals that enable a controlled teacher–student adaptation loop.

This design is critical because naive feedback learning can amplify errors. If a teacher model is wrong or an association is incorrect, pseudo-labels may reinforce failure modes and degrade performance over time. Our gating approach—conditioning pseudo-label acceptance on association confidence, teacher confidence, latency constraints, and uncertainty thresholds—reduces this risk. In effect, the orchestrator functions as a judging agent that determines whether evidence is suitable for learning. This parallels "controlled reinforcement learning" in safety-critical systems: improvements are allowed only when feedback quality is high and when deployment validation criteria are met.

8.4. Generalization Beyond the Dual-Camera Use Case

Although evaluated on dual-camera obstacle recognition, the proposed orchestration pattern generalizes to a broad class of systems that combine:

- fast/low-cost agents (real-time edge models),
- slow/high-quality agents (infrastructure sensors, cloud services, heavy models),
- association/verification agents (matching, checking consistency),
- and an orchestrator that must decide under uncertainty and timing constraints.

Potential applications include:

- cooperative perception and V2X sensor networks,
- robotics perception with multi-view cameras and remote supervision,
- anomaly detection pipelines where slow forensic analysis informs fast detection,
- multi-sensor medical imaging triage pipelines.

The essential principle is unchanged: probabilistic orchestration enables robust decisions and safe adaptation when agents differ in reliability and latency.

8.5. Relevance to GenAI Multi-Agent Orchestration

The same orchestration challenges now emerge in agentic generative AI systems. Modern GenAI workflows often involve multiple agents and tools, such as:

- a fast, low-cost LLM for drafting responses,
- a slower, high-quality LLM for refinement,
- retrieval agents that fetch evidence,
- verifiers that validate claims,
- policy agents that enforce safety and compliance.

 doi:10.20944/preprints202602.0030.v1

In these systems, deterministic orchestration (e.g., fixed tool order or always trusting one model) is brittle under hallucinations, retrieval failures, and distribution shift. The analogy to our perception system is direct:

- Teacher agent: a higher-quality LLM or verified tool output,
- Student agent: a smaller or faster model operating under latency/cost constraints,
- Association/verification: evidence linking, citation checks, tool validity checks,
- Orchestrator: a probabilistic policy that fuses model outputs and triggers refinement or verification based on uncertainty.

Crucially, our "gated learning" principle maps to GenAI settings where systems must learn from feedback without reinforcing hallucinations. For example, only high-confidence verified outputs should be used as training or adaptation signals; low-confidence or unverified outputs should be excluded. This suggests a unified design pattern for agentic systems across modalities: probabilistic orchestration + verified feedback + controlled deployment.

8.6. Safety, Accountability, and Explainability

Safety-critical systems require not only correctness but also accountability. By producing a decision trace and uncertainty estimates, the orchestrator enables:

- auditability: tracking which agent influenced each decision,
- explainability: communicating the evidence used (confidence, match score, latency),
- intervention hooks: enabling downstream policies such as "do not act when uncertainty is high."

This is aligned with Responsible AI principles applied to agentic systems: safe behaviour requires traceability and the ability to veto or roll back decisions and model updates. Our controlled deployment criteria (Section 5.7) provide a minimal framework for safety gating, though additional safeguards can be integrated (e.g., human approval for high-risk updates, adversarial testing before deployment).

8.7. Limitations and Future Work

Several limitations remain:

1. Dependence on calibration and association: The orchestrator assumes that confidence can be calibrated and association reliability can be estimated. Severe calibration failures or systematic association errors may still degrade performance.
2. Planar-road assumption: Homography-based projection is approximate and may fail in non-planar environments or under calibration drift.
3. Infrastructure availability: Stationary cameras may not always be available; while adaptation reduces reliance over time, deployment depends on infrastructure coverage.
4. Offline adaptation: We focus on offline/nearline learning; future work could explore safe online adaptation with stronger safeguards.

Future directions include:

- learned or adaptive gating policies that adjust thresholds under drift,
- integration with additional sensors (lidar/radar) and temporal tracking,
- richer uncertainty models (ensembles or evidential learning),
- broader evaluation across environments and weather conditions,
- extension to full multi-agent GenAI orchestration pipelines with verification and policy gating.

9. Conclusion

This paper introduced a probabilistic orchestrator for multi-agent perception pipelines that must operate under uncertainty, conflicting predictions, and variable latency. Motivated by dual-camera road obstacle recognition, we showed how deterministic orchestration mechanisms—such as hard thresholds and rule-based state machines—are brittle when combining fast ego-camera predictions with delayed but higher-confidence stationary-camera evidence. Our orchestrator addresses these

 doi:10.20944/preprints202602.0030.v1

limitations by fusing agent outputs using calibrated confidence, cross-view association reliability, and latency-aware policies, producing uncertainty-aware decisions and decision traces that support auditability.

Beyond inference-time fusion, we integrated a controlled teacher–student adaptation loop within the orchestration layer. High-confidence stationary detections serve as teacher signals for fine-tuning and knowledge distillation of the ego detector, while orchestrator gating (association confidence, teacher confidence, latency constraints, and uncertainty thresholds) mitigates label noise and reduces the risk of error amplification. Evaluation across distance, occlusion, and latency conditions demonstrates that probabilistic orchestration improves robustness and stability compared to ego-only and deterministic baselines, and that the gated learning loop further strengthens ego performance over time while controlling false alarms.

The proposed framework generalizes beyond the autonomous driving case study to broader multi-agent systems where evidence sources differ in accuracy and timing, including infrastructure-assisted perception, robotics, and multi-agent generative AI workflows. Future work will explore richer uncertainty models, learned gating strategies, online adaptation with stronger safeguards, and expanded evaluation across diverse environments and sensor configurations.

10. Broader Impact / Ethics & Safety (Addendum)

Multi-agent systems that coordinate perception and learning can improve safety and reliability by reducing missed detections and enabling adaptation to changing environments; however, they also introduce risks associated with incorrect evidence fusion and unsafe learning updates. The approach in this paper explicitly addresses these risks through uncertainty-aware decision making, traceable orchestration, and gated learning, ensuring that high-impact updates are triggered only under reliable association and confidence conditions and are validated prior to deployment with rollback capability. Nevertheless, real-world deployment must consider additional safeguards, including robust calibration under distribution shift, monitoring for drift and failure modes, privacy-preserving handling of sensor data, and human oversight for high-risk scenarios. Infrastructure-assisted perception also raises governance questions (e.g., sensor placement, access control, and accountability for failures). We view probabilistic orchestration with controlled learning as a step toward safer agentic systems, but emphasize that operational deployment should include formal testing, continuous monitoring, and clear responsibility boundaries among system operators and stakeholders.

Appendix A

Ground-Plane Homography (Fast & Robust Algo)

For road users we care about contact points on the road plane (nearly planar). Use a homography Hper camera to map image pixels to ground plane (world XY).

If the ground plane has normal $n$and distance d, the homography from image $i$to ground is:

$$H_i = K_i(R_i - \frac{t_i\, n^\top}{d})K_i^{-1}$$

In practice you can:

- Solve $H_i$once using 4–8 surveyed road points visible in the image (OpenCV findHomography with RANSAC).
- For every detection, take the bottom-centre of the bounding box (contact point) $(u, v, 1)$, map with H_i^{-1}(or H_i, depending on convention) to ground coordinates, normalize by scale, and you get (X, Y)in meters.

Result: both the stationary and ego detections become 2D points on the road, in the same metric frame.

 doi:10.20944/preprints202602.0030.v1

References

1. A. Kojukhov, I. Levin, and A. Bovshover, "From Subsumption to Semantic Mediation: A Generative Orchestration Architecture for Autonomous Systems," Algorithms, vol. 18, no. 12, art. no. 773, 2025.
2. M. Wooldridge, An Introduction to MultiAgent Systems, 2nd ed. Wiley, 2009.
3. Y. Shoham and K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, 2008.
4. L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," Journal of Artificial Intelligence Research, vol. 4, pp. 237–285, 1996.
5. S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Pearson, 2020.
6. S. Rajbhandari et al., "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models," SC, 2020.
7. M. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," NeurIPS, 2020.
8. R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Journal of Basic Engineering, 1960.
9. Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," ICML, 2016.
10. A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" NeurIPS, 2017.
11. C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," ICML, 2017.
12. B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," NeurIPS, 2017.
13. M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential Deep Learning to Quantify Classification Uncertainty," NeurIPS, 2018.
14. H. Wang et al., "Multi-View 3D Object Detection and Tracking: A Survey," IEEE T-ITS, 2021.
15. G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," NIPS DL Workshop, 2015.
16. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.
17. D.-H. Lee, "Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks," Workshop on Challenges in Representation Learning, ICML, 2013.
18. X. Chen et al., "Self-Training for Few-Shot Transfer Across Extreme Domain Shift," ICLR, 2020.
19. K. Sohn et al., "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence," NeurIPS, 2020.
20. A. Tarvainen and H. Valpola, "Mean Teachers are Better Role Models: Weight-Averaged Consistency Targets Improve Semi-Supervised Deep Learning Results," NeurIPS, 2017.
21. A. Geiger, P. Lenz, and R. Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," CVPR, 2012.
22. N. Caesar, J. Uijlings, and V. Ferrari, "COCO-Stuff: Thing and Stuff Classes in Context," CVPR, 2018.
23. J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv:1804.02767, 2018.
24. L. Chen et al., "Cooperative Safety Intelligence in V2X-Enabled Transportation: A Survey," arXiv:2512.00490, 2025.
25. H. Zhang et al., "VICooper: A Practical Vehicle–Infrastructure Cooperative Perception Framework," in Computer Vision and Image Processing, Springer, 2024.
26. L. Chen et al., "Cooperative Perception Datasets and Benchmarks: A Survey," IEEE Communications Surveys & Tutorials, 2023.
27. H. Liu et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," arXiv:2302.04761, 2023.
28. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," NeurIPS, 2022.

 doi:10.20944/preprints202602.0030.v1

29. T. Brown et al., "Language Models are Few-Shot Learners," NeurIPS, 2020.
30. S. Welleck et al., "Self-Refine: Iterative Refinement with Self-Feedback," arXiv:2303.17651, 2023.