Digital Divide Data

Collaborative Perception & V2X for ADAS/AV

The Next Data Problem (multi-agent datasets, labeling complexity, and fusion-ready ground truth)

White Paper

February

2026

Authors

Udit Khanna
(Director- AI Solutions)

Kevin Sahotsky
(Head of Strategic Partnerships and GTM)

A practical whitepaper for:

Robotaxi AV stacks • L2/L2+ passenger ADAS • Commercial trucking

Presented by

www.digitaldividedata.com

TABLE OF CONTENTS

Collaborative Perception & V2X for ADAS/AV:

The Next Data Problem (multi-agent datasets, labeling complexity, and fusion-ready ground truth)

A practical whitepaper for: Robotaxi AV stacks • L2/L2+ passenger ADAS • Commercial trucking

1. Introduction

Collaborative perception promises something that single-vehicle systems struggle with: seeing what isn't directly visible. A delivery van blocks a pedestrian at a crosswalk. A vehicle cuts in from two lanes over, still partially occluded. A stalled car sits just beyond the crest of a hill in light rain.

In theory, V2V and V2I sharing should soften those edges. In practice, many programs discover, often a little too late. that the real friction isn't in model architecture. It's in the data plumbing underneath.

Teams rarely stall because their fusion network lacks capacity. They stall because clocks drift by 12 milliseconds, and no one notices until duplicate objects start appearing. Because infrastructure calibration quietly shifts after a maintenance event. Because track IDs don't survive occlusion consistently across agents. Or because the message arrived... just slightly too late to matter.

It's uncomfortable to admit, but collaborative perception appears to be less of an algorithm problem and more of a systems discipline problem. And systems discipline is, at its core, a data discipline.

Where Collaboration Actually Breaks

In multi-agent systems, small upstream inconsistencies don't stay small. They compound.

A timestamp that is technically monotonic, but off by 8–15 ms, turns into spatial misalignment at highway speeds. Calibration drift in an infrastructure LiDAR shows up downstream as phantom obstacles. Frame mismatches quietly generate duplicates that look like perception regressions. And bandwidth or network jitter transforms correct information into what operators sometimes call the "right answer too late" failure mode.

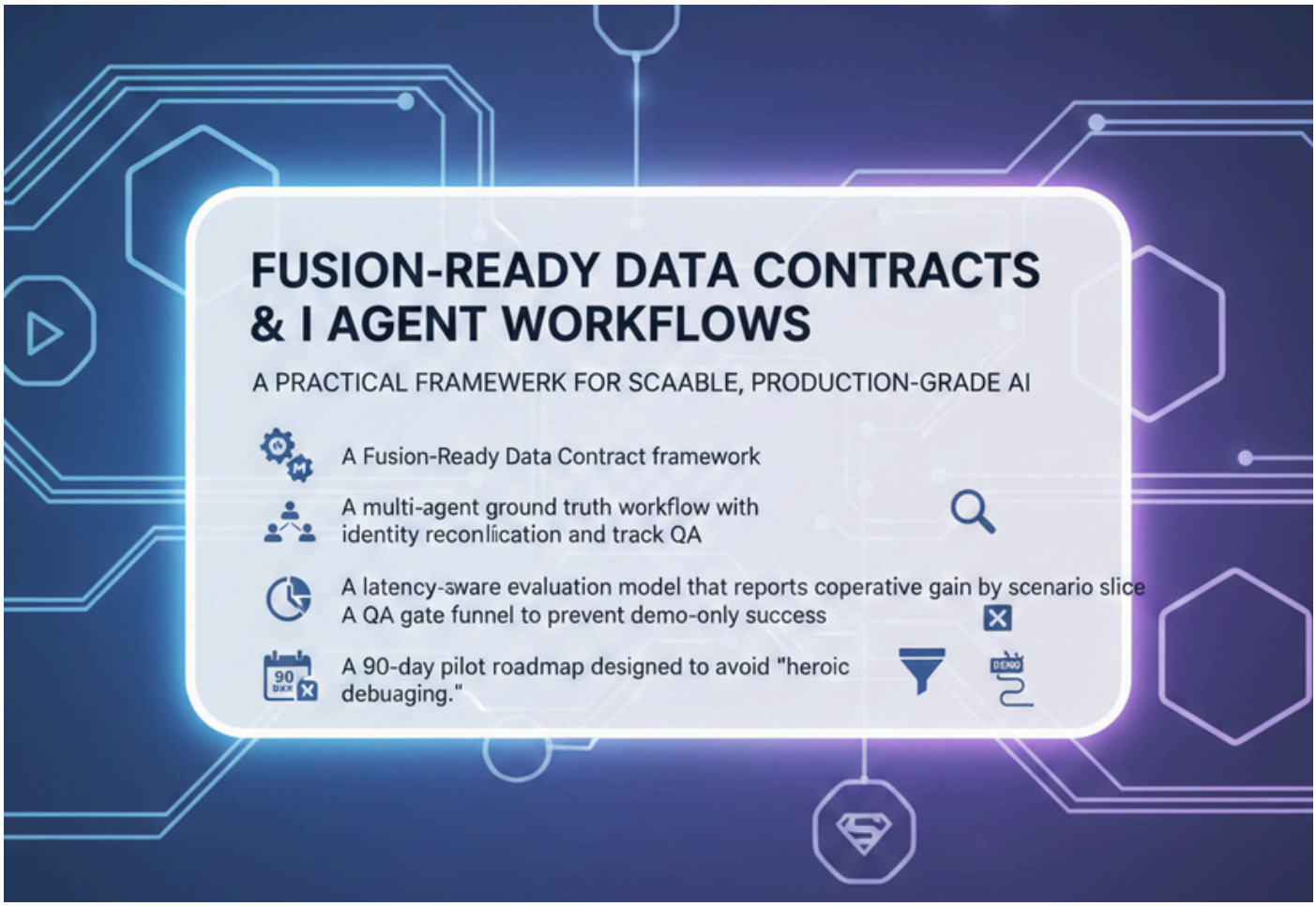
Individually, each issue seems manageable. Together, they erode trust in cooperative gain.
I've seen pilot dashboards where cooperative perception improved recall by 7–10% in occlusion slices. And yet the program still hesitated to ship, because ID switches doubled, or late-message rates spiked during peak traffic hours. The signal was there. The operational confidence wasn't.

Collaboration Requires Alignment, Consistency, and Latency Discipline

If we reduce it down to fundamentals, collaborative perception is three things:

- Alignment (time, frames, transforms, calibration)
- Consistency (schema, identity lifecycle, uncertainty semantics)
- Latency discipline (bounded, measured, enforced)

Miss any one of these, and fusion becomes unstable. Not catastrophically, just subtly enough to create hard-to-debug regressions. That's why this paper takes a position that may sound procedural, even unglamorous: Before scaling collaborative perception, you need a Fusion-Ready Data Contract.



This whitepaper lays out a practical framework that scales across:

- A Fusion-Ready Data Contract framework (minimum viable vs. production-grade)
- A multi-agent ground truth workflow with identity reconciliation and track QA
- A latency-aware evaluation model that reports cooperative gain by scenario slice
- A QA gate funnel to prevent demo-only success

A 90-day pilot roadmap designed to avoid "heroic debugging."

2. Collaboration Modes & Design Choices

Collaborative perception isn't a single architectural choice. It's a series of trade-offs, who shares, what gets shared, and how fast it needs to arrive.

Teams sometimes speak about "adding V2X" as if it were a feature flag. In reality, the design space is wider and unforgiving.. A decision about whether to share object tracks versus raw features can quietly triple your annotation burden. Choosing hybrid V2V + V2I may expand coverage, but it also multiplies calibration surfaces and failure modes. So before thinking about fusion models, it helps to step back and ask a more basic question:

What kind of collaboration are we actually building?

2.1 Who Shares: V2V, V2I, or Hybrid?

V2V (Vehicle-to-Vehicle)

Picture a highway merge where a vehicle two cars ahead detects a cut-in early. Sharing that track even 300 milliseconds sooner can meaningfully change downstream planning. In theory, it's elegant.

In practice, V2V must tolerate heterogeneous sensor stacks. One car runs a camera + radar. Another adds LiDAR. Pose quality differs. Calibration maintenance schedules differ. Even confidence semantics may differ.

It's tempting to assume "an object is an object." But object definitions aren't universal. A sedan classified with 0.82 confidence from one stack may not mean the same thing as 0.82 from another. And that subtle mismatch can bias fusion weighting.

V2V also raises a less glamorous issue: identity collision. Two vehicles tracking the same pedestrian from different vantage points need consistent association logic, or duplicates start accumulating downstream.

V2I (Vehicle-to-Infrastructure)

Infrastructure-mounted sensors don't blink, don't get occluded by their own A-pillars, and can observe VRUs from elevated vantage points. In dense urban environments, that perspective matters. But V2I assumes something that sounds simple and rarely is: stable infrastructure frames.

A pole-mounted LiDAR that shifts a few millimeters after maintenance may seem harmless. Yet over distance, that transform error compounds. Vehicles fusing that feed now absorb the misalignment.

And then there's association. Infrastructure sees a cyclist. The vehicle sees the same cyclist from a different angle. Whose track ID persists? How do you reconcile lifecycles across agents?

V2I can reduce occlusion risk significantly. It also centralizes the burden of calibration governance.

Hybrid (V2V + V2I)

Hybrid collaboration offers the most theoretical coverage. And perhaps the most operational complexity.

More agents mean more clocks, more transforms, and more identity policies to harmonize. The combinatorics grow quickly. What feels like incremental capability can become an exponential integration effort.

Still, for robotaxi deployments in dense cities, hybrid models may be hard to avoid. The question isn't whether hybrid works. The question is whether the underlying data contract can scale with it.

2.2 What Is Shared: Features, Objects, Tracks, or Occupancy?

The second axis is arguably more consequential than the first.

What you share defines not just bandwidth requirements, but labeling complexity, schema governance, and evaluation burden.

Features

Sharing intermediate features, BEV embeddings, voxelized tensors, and learned representations offers high theoretical upside. You preserve more information before decisions are compressed into discrete objects.

But this path demands high bandwidth and tight synchronization. It also assumes interoperability across model architectures, which is rarely trivial. From a data standpoint, feature sharing complicates validation. How do you audit a feature tensor? What does schema validation even mean at that layer? Debugging becomes less intuitive.

Feature-level collaboration may be powerful. It is also, at least today, harder to operationalize across heterogeneous fleets.

Objects

Bounding boxes, classifications, and velocities are human-interpretable. They're easier to log, audit, and annotate. Most early pilots land here.

Yet object sharing quietly requires consistent schema semantics.

What does confidence represent? Detection probability? Post-tracking probability? Does velocity reflect ego-relative or global frame? If two senders encode heading differently, fusion weightings can drift.

Tracks

Instead of sharing isolated detections, agents share temporally linked objects with lifecycle semantics. This reduces association burden downstream and can stabilize fusion.

However, track sharing forces you to define identity policy explicitly. When does an ID persist across occlusion? When does a split occur? How long before a reappearance counts as a new entity? Without a written policy, annotators improvise. Improvisation, in multi-agent systems, becomes instability.

Occupancy / Free-Space

Occupancy grids and free-space maps are particularly useful for planning stacks. They abstract away object identity and focus on navigable space. That abstraction simplifies some things and complicates others.

Frame semantics must be unambiguous. Latency bounds must be tight. An occupancy update that lags by 200 ms at 25 m/s changes meaning rapidly. Occupancy sharing may reduce identity headaches. It increases sensitivity to time discipline.

3. The Missing Piece: The Fusion-Ready Data Contract

Most collaborative perception pilots start with models.
Production programs eventually discover they need a contract.

When collaboration breaks, it rarely breaks because the fusion architecture is incapable. It breaks because assumptions were never written down. Someone assumed timestamps were aligned within 5 ms. Someone else assumed transforms were static. Another team assumed ID lifecycles were “handled in tracking.”

A Fusion-Ready Data Contract is essentially the API specification for perception sharing plus the operational guarantees that sit behind it. It defines what must be true about time, frames, calibration, identity, uncertainty, and traceability before cooperative gain is considered real.

What the Contract Actually Covers

At a minimum, a production-grade collaborative system needs agreement and measurable guarantees around:

- **Time** (timestamp semantics, monotonicity, bounded sync error)
- **Coordinate frames & transforms**
- **Calibration versioning and drift detection**
- **Identity lifecycle rules**
- **Uncertainty representation**
- **Schema validation**
- **Traceability from sensor to fused output**
- **Regression gates tied to scenario slices**

That may sound like governance overhead. It's not. It's fusion stability insurance.

The uncomfortable reality is this: fusion is exquisitely sensitive to upstream inconsistencies. If two agents disagree subtly about frame alignment or uncertainty scaling, fusion weights shift.

Track continuity degrades. Duplicate suppression becomes probabilistic guesswork.

3.1 Minimum Viable vs. Production-Grade

Not every program needs a production-grade contract on day one. But the difference between minimum viable and production-ready should be explicit, not aspirational.

Minimum Viable Contract (Pilot Phase)

At a minimum, you need:

- Timestamps with declared semantics and monotonic guarantees
- Clearly defined coordinate frames and transforms
- Calibration version identifiers
- A consistent object/track schema
- Sender identity metadata
- A small, audited gold set

This is enough to demonstrate cooperative gain in controlled slices. It's not enough to absorb real-world drift. In early pilots, teams often rely on manual spot checks. Someone inspects misalignments visually. Someone else cross-checks a handful of scenes. That can work for weeks.

Production-Grade Contract

Production demands measurable bounds and automated enforcement.

A production contract includes:

- **Bounded sync error**, continuously measured and monitored
- **Pose uncertainty models** are explicitly used in fusion weighting
- **Automated calibration drift detection,** with quarantine logic
- **Strict schema validation** and message sanity checks
- **A written identity lifecycle policy,** enforced in labeling and evaluation
- **End-to-end traceability**, from raw message to fused decision
- **Regression gates by scenario slice,** not just global metrics

The difference isn't philosophical. It's operational.

In one trucking program I observed, cooperative perception improved long-range detection recall by nearly 9% in controlled highway slices. But production rollout was paused. The reason? Late-message rates climbed under network congestion, and no alerting existed. The improvement was real, but fragile.

A production contract would have caught the latency envelope breach before it reached evaluation dashboards.

The Fusion Readiness Score (FRS)

To make the contract measurable, we propose a **Fusion Readiness Score (FRS)**, a weighted rubric that surfaces weak links early.

Categories may include:

- Time sync bounds + monitoring
- Frames and transform governance
- Calibration versioning + drift detection
- Ego pose and uncertainty modeling
- Schema validation and sanity checks
- Identity lifecycle policy
- Traceability and auditability
- Regression gates and slice coverage

Each category is scored 0–5. Weighted. Sums to a 0–100 readiness index.
Is a single number reductive? Possibly. But it forces conversations that otherwise stay implicit.
A program with strong model metrics but a readiness score of 58 probably shouldn't scale. A score of 85 doesn't guarantee success, but it suggests the foundation is less brittle. The FRS isn't about compliance theater. It's about making invisible fragility visible.

4. Fusion-Ready Ground Truth: Multi-Agent Labeling That Works

Collaborative perception doesn't just need more labels. It needs different labels.

One of the most common shortcuts in early pilots is to reuse single-agent ground truth and assume fusion will "figure out the rest." Sometimes it works, at least in clean slices. But once you introduce occlusion, cross-agent disagreement, or partial visibility, single-agent labels start to show their limits. Fusion becomes unstable, not because the math is wrong, but because the supervision was never designed for multi-agent consistency.

Ground truth, in collaborative systems, has to be multi-agent native. That means identity consistency across agents, explicit duplicate handling, and track continuity that survives occlusion and reappearance. Without those, the evaluation may suggest cooperative gain while hiding instability underneath.

Identity Policy Comes First

Before a single bounding box is reconciled across agents, there needs to be an identity policy. Not an implicit understanding, an actual written rulebook.

What counts as the "same object"?
How long does an ID persist through occlusion?
When two partial observations diverge slightly, do we merge or split?
When do we admit ambiguity instead of forcing a decision?

If these rules are undefined, annotators improvise. And improvisation leads to disagreement. Disagreement leads to inconsistent supervision. In multi-agent fusion, inconsistent supervision shows up as duplicate tracks, ID switches, or fragmentation spikes that are hard to diagnose.

An identity policy typically specifies:

- Occlusion persistence thresholds (for example, a pedestrian ID may persist for 2 seconds without observation; a vehicle for 4)
- Reassociation thresholds based on position, velocity, and heading
- Split/merge criteria (e.g., diverging trajectories for more than N frames trigger a split)
- Duplicate definitions (overlap within a spatial tolerance for a defined duration)
- Ambiguity rules (tag uncertain identity and route to adjudication rather than forcing a merge)

The last point is often overlooked. Forcing annotators to decide in ambiguous cases can introduce artificial certainty. That artificial certainty later destabilizes fusion weighting. Sometimes it's better to admit uncertainty and escalate.

Why Single-Agent Labels Aren't Enough

In single-agent datasets, identity continuity is already challenging. In multi-agent contexts, it's amplified.

Consider a pedestrian partially occluded by a parked truck. Infrastructure sees the full body from above. The ego vehicle sees only a partial silhouette. Each agent may assign a different bounding box extent and slightly different center position. Individually, both are defensible. When fused, they can either merge cleanly, or create duplicate artifacts depending on identity reconciliation rules. If ground truth doesn't explicitly encode cross-agent identity relationships, evaluation becomes misleading. The fusion system may appear to produce duplicates, when in fact the labels never defined how those identities should align. Multi-agent ground truth must reflect the reality that objects are observed differently by different agents. The goal isn't geometric perfection. Its identity consistency under viewpoint variation.

A Scalable Multi-Agent Annotation Workflow

First comes single-agent labeling. Each vehicle or infrastructure stream is annotated independently, following class definitions and spatial guidelines. This stage should already enforce temporal continuity within each stream.

Next comes cross-agent identity reconciliation. Objects that represent the same real-world entity are linked across sources according to the identity policy. This step is where many programs underinvest. It requires tools that allow annotators to visualize multi-view alignment and make decisions with context.

After reconciliation, tracking QA becomes critical. Continuity is reviewed. Fragmentation is measured. Occlusion handling is stress-tested in dense or ambiguous scenes. Track-level metrics, ID switch rate, and fragmentation frequency should be computed even before fusion models are trained.

A gold set, carefully curated across priority scenario slices, anchors the process. When annotators disagree or when guideline gaps surface, an adjudication loop resolves ambiguity and updates the policy. Over time, the guideline evolves. Without that feedback loop, inconsistencies accumulate quietly.

Audit sampling each release helps prevent drift in labeling quality. Annotation programs, like calibration systems, can drift if not monitored.

Track QA Is Not Optional

In single-agent datasets, identity continuity is already challenging. In multi-agent contexts, it's amplified.

If ID switches double in occlusion-heavy intersections, fusion may still improve recall, but planning stability can degrade. A vehicle that alternates between two identities every few frames creates downstream uncertainty.

Track QA should measure:

- ID switch rate
- Fragmentation frequency
- Duplicate rate across agents
- Occlusion persistence behavior
- Reassociation accuracy

These aren't secondary metrics. They are early indicators of fusion health.

I've seen systems where detection AP improved modestly with cooperation, yet fragmentation rose sharply in dense VRU slices. The overall dashboard looked positive. Operators, however, noticed planning hesitation in those same scenes. The ground truth had not enforced a consistent identity persistence rule across agents. The model was reacting to label ambiguity.

5. Evaluation That Matches Reality: Latency-Aware KPIs + Cooperative Gain

Evaluating collaborative perception the same way you evaluate a single-agent detector is... tempting. You already have mAP dashboards. You already track recall, precision, and maybe some tracking metrics. Why not just add a "+V2X" column and compare? Because collaboration changes the failure surface.

A single-agent detector can be late and still look accurate on frame-level metrics. A multi-agent system cannot. A shared object that arrives 250 milliseconds too late at highway speed isn't neutral; it's misleading. And a fused track that switches IDs twice in three seconds might technically count as detected, yet behave unpredictably downstream.

Stability Is a First-Class Metric

In collaborative systems, stability isn't a nice-to-have. It's foundational.

ID switch rate, duplicate object rate, and track fragmentation are not peripheral metrics. They are signals of whether multi-agent alignment is working under stress.

Imagine an urban intersection sliced with dense VRUs. Cooperative perception increases pedestrian recall by 6%. On paper, that's progress. But if ID switches increase in the same slice, planners may see oscillating object histories. The net system behavior could feel less confident, not more.

Track continuity, in particular, deserves careful attention. It's not enough to know that an object was detected. You need to know whether it remained coherent across occlusion, across agent boundaries, across brief latency spikes.

Latency Changes Meaning

Latency in collaborative perception isn't just a systems metric. It changes semantic truth.

A message that is technically correct but arrives outside a usable time window may distort fused output. For example, a vehicle entering from a blind spot may already have an altered trajectory by the time an infrastructure message arrives. The fused track may appear to "jump" or overshoot.

That's why evaluation needs latency-aware KPIs such as:

- Late message rate (percentage of messages outside acceptable time bounds)
- Stale observation rate (observations fused after relevance window)
- Latency-to-impact (time from event onset to fused confidence threshold)

Latency-to-impact, in particular, reframes evaluation. It asks: how long does it take for cooperation to meaningfully influence the fused output? If cooperation improves recall but doesn't reduce latency-to-impact in high-risk slices, say, early cut-ins at 80 km/h, its practical value may be limited.

Dashboards Should Reflect Operational Reality

A useful collaborative perception dashboard doesn't just show precision and recall. It shows:

- Cooperative Gain by slice
- Late/out-of-sync trend lines
- ID switch and duplicate rates
- Track continuity distributions
- Optional false-positive cost curves

When those panels are viewed together, patterns emerge. A spike in duplicates might correlate with a sync drift anomaly. A drop in latency-to-impact may coincide with improved calibration governance. Without this integrated view, teams risk chasing model tweaks for what are actually upstream contract violations.

6. QA Gates & the Release Funnel

At some point, every collaborative perception team faces the same uncomfortable question:
Are we debugging, or are we operating?

In early pilots, debugging dominates. Engineers inspect logs manually. Someone exports a few problematic scenes. A sync issue is discovered after a late-night Slack thread. It's messy, but manageable, because the scope is small.

Scale changes that dynamic. Once vehicles or infrastructure nodes are deployed across geographies, manual vigilance no longer holds. What used to be a "we'll catch it" assumption turns into regression drift.

That's where QA gates matter. Not as a ceremony, but as protection.

From Debugging to Guardrails

Collaborative perception is especially prone to upstream leakage. A calibration shift here, a message schema change there, and suddenly fusion metrics move in ways that look like model regressions.

Without automated gates, teams spend cycles tuning models to compensate for what are essentially contract violations. A release funnel forces discipline. It catches upstream anomalies before they masquerade as perception instability. And importantly, it distributes responsibility. The burden doesn't sit solely on the model team.

The Release Gate Funnel

A production-grade release process should move through clearly defined stages:

1. Data Intake Validation
Before anything touches fusion, incoming data is validated for structural integrity. Are required fields present? Are timestamps monotonic? Are coordinate frames declared correctly?

2. Sensor Health Checks
This includes sync drift monitoring, transform sanity validation, and calibration drift detection. If calibration shifts beyond tolerance, quarantine logic should trigger automatically. No quiet degradation.

3. Message Sanity Checks
Schema validation becomes strict, not advisory. Field ranges are enforced. Out-of-order or duplicate message rates are tracked. Latency envelopes are monitored continuously.

4. Label QA
Gold set comparisons and audit sampling validate that annotation quality hasn't drifted. Identity policies are checked for consistent application.

5. Fusion Evaluation
Stability metrics and latency-aware KPIs are computed by scenario slice. Cooperative Gain is reported relative to baseline.

5. Fusion Evaluation
Stability metrics and latency-aware KPIs are computed by scenario slice. Cooperative Gain is reported relative to baseline.

6. Regression Gates
Explicit ship/no-ship thresholds are applied. Not just on global metrics, but on critical slices.

This funnel isn't meant to slow progress. It's meant to prevent downstream instability from creeping into production releases.

The First Eight Checks to Automate

If automation bandwidth is limited, start here:

- Sync drift score per sender
- Calibration drift or transform sanity violations
- Message schema validation and missing-field detection
- Out-of-order and duplicate message rate
- Late or stale observation rate
- Duplicate object rate post-fusion
- ID switch rate and fragmentation trend
- Scenario slice coverage change

Slice-Based Ship/No-Ship Criteria

One subtle but important shift in collaborative perception is tying release decisions to scenario slices rather than aggregate performance.

A robotaxi stack might tolerate slight regression in low-density highway slices, but not in dense pedestrian intersections. A commercial trucking program might prioritize long-range cut-in stability above all else.

So release gates should reflect those priorities.

“Don’t ship if ID switches rise in critical occlusion slices.”
“Don’t ship if the late-message rate exceeds the threshold in highway merges.”
“Don’t ship if the duplicate rate increases under adverse weather.”

This makes release governance context-aware rather than metric-obsessed.

QA as a Cultural Signal

There’s also a softer dimension here. QA gates signal that collaborative perception is not experimental plumbing, it’s production infrastructure.

When drift detection automatically quarantines a miscalibrated sender, teams internalize that contract compliance matters. When regression gates block release due to slice instability, it reinforces that cooperative gain alone is insufficient.

In my experience, the presence of automated gates changes behavior upstream. Teams think more carefully about schema updates. They version identity policies more deliberately. They measure sync error more rigorously.

It shifts collaborative perception from hopeful experimentation to disciplined systems engineering.
Collaborative perception doesn’t scale because the model improved by 2%.
It scales because upstream inconsistencies are caught early, enforced systematically, and prevented from cascading.

7. Segment Playbooks: Same Framework, Different Priorities

It's tempting to think collaborative perception is a universal upgrade to V2X, improve safety, and move on. But once you look closely, the operational reality of a robotaxi fleet is nothing like an L2 passenger vehicle program. And neither resembles long-haul trucking in winter spray at night.

The framework we've outlined, data contract, multi-agent ground truth, latency-aware KPIs, and QA gates can be applied across all three. The priorities cannot.

That distinction matters. If you apply the wrong slice emphasis or KPI weighting to the wrong segment, you may optimize for a problem that barely exists while ignoring one that defines your risk profile.

Robotaxi AV Stacks

Pedestrians stepping out from behind delivery vans. Cyclists weaving between lanes. Infrastructure-assisted perception often shows its strongest gains here, especially with elevated viewpoints.

So for robotaxi programs, cooperative gain under occlusion-heavy slices is often the primary KPI. Not global recall, occlusion-specific improvement. And stability matters deeply. Duplicate pedestrians or ID switches in dense VRU zones aren't cosmetic. They affect planner confidence and rider comfort.

A reasonable "don't ship" condition in this segment might read:

- Do not ship if duplicate or ID switch rates increase in high-VRU intersection slices.
- Do not ship if occlusion persistence degrades under cooperative fusion.

Contract focus in this segment leans heavily toward frame governance, identity lifecycle discipline, and traceability. When something goes wrong in a city center, you need to trace it precisely.

Passenger ADAS (L2/L2+)

Passenger ADAS operates under a different constraint: driver trust. On highways, cooperative perception might help with early cut-ins or stopped vehicles beyond line-of-sight. That's valuable. But if cooperation increases false-positive braking events, even slightly, the user experience degrades quickly.

For this segment, false-positive cost becomes a central KPI. Cooperative gain must not come at the expense of stability across vehicle variants and sensor trims.

Slice priorities often include:

- Highway cut-ins
- Stopped vehicles at long range
- Merge scenarios
- False braking conditions

Here, release gates might look like:

- Do not ship if the false-positive cost increases beyond the threshold.
- Do not ship if trim-specific regressions exceed defined bounds.

The contract focus shifts slightly. Variant metadata, schema sanity validation, and message integrity checks are critical. ADAS fleets may have wide hardware diversity; consistency across variants matters as much as raw cooperative gain.

Same Framework, Different Emphasis

What's interesting is that the underlying mechanics, data contract, identity policy, and QA gates don't fundamentally change across segments. What changes is the weighting.

A robotaxi team might accept slightly higher latency if occlusion recall improves dramatically. A trucking team likely will not. An ADAS team may reject a release over a modest false-positive regression that a robotaxi team would consider tolerable. The framework must be flexible enough to reflect those realities.

And perhaps that's the key takeaway here: collaborative perception isn't a monolithic upgrade. It's a context-dependent system enhancement. Applying a single global KPI across segments is likely to obscure more than it reveals.

If you tune slices, thresholds, and contract focus deliberately, the same disciplined foundation can support very different operational goals.

8. 90-Day Pilot Roadmap + Partner Checklist

Most collaborative perception pilots fail for a predictable reason: they optimize for the demo before stabilizing the foundation.

The first live test looks promising. Cooperative recall jumps in a few curated scenes. A slide deck circulates. But somewhere between week six and week twelve, reality intrudes, sync drift appears, identity inconsistencies surface, latency envelopes fluctuate, and no one is quite sure whether the gain is durable.

A 90-day pilot, done well, should prove two things:

1. Cooperative gain exists in priority slices.
2. The operational path to production is viable.

If it only proves the first, you've built demo debt.

Weeks 0–2: Define Before You Build

Resist the urge to start collecting everything immediately.

The first two weeks should focus on narrowing the scope:

- Define collaboration mode (V2V, V2I, hybrid).
- Select 5–10 priority scenario slices.
- Draft the minimum viable Fusion-Ready Data Contract.
- Instrument telemetry for sync error, latency, and calibration versioning.

It's tempting to postpone telemetry, assuming it can be added later. It can, but then you'll lack baselines when anomalies appear.

Weeks 3–5: Build Ground Truth That Matches the System

Collect scenes that stress your priority slices. Build a small, carefully curated gold set. Draft and formalize the identity policy. Stand up an adjudication loop early, even if the volume feels manageable.

During this period:

- Label single-agent streams.
- Perform cross-agent identity reconciliation.
- Run track QA metrics (ID switches, fragmentation).
- Update guidelines based on disagreement patterns.

If annotators disagree frequently, treat that as a signal, not friction. It may suggest the identity policy is underspecified. By the end of week five, you should have a stable gold set and an identity governance document that feels precise, not aspirational.

Weeks 6–8: Evaluate Honestly

This is where many pilots jump prematurely to model iteration. Instead, establish a clean baseline vs. cooperative comparison. Compute Cooperative Gain by slice. Introduce stability metrics and latency-aware KPIs. Build your first integrated dashboard.

Don’t just look at recall deltas. Look at:

- Late/stale observation rate trends.
- Duplicate and ID switch behavior in dense slices.
- Latency-to-impact shifts in high-speed scenarios.

If cooperative gain appears in some slices but not others, that’s normal. What matters is whether the slices you prioritized show measurable improvement without destabilization.
If instability appears, resist the instinct to tune the model immediately. Check contract enforcement and identity policy first.

Weeks 9–10: Automate the Guardrails

Up to this point, much of the pilot may have relied on manual review. That’s fine. But now automation has become mandatory.
Implement the first set of QA gates:

- Sync drift scoring.
- Calibration drift detection.
- Schema validation checks.
- Out-of-order and duplicate message tracking.
- Stability threshold alerts.

Define preliminary ship/no-ship thresholds for critical slices, even if they’re conservative. Expand the gold set slightly. Increase audit cadence. If metrics fluctuate, investigate upstream before attributing to fusion. This is the phase where the pilot either matures or reveals fragility.

Weeks 11–12: Production Readiness Assessment

The final stretch isn’t about squeezing out one more percentage point of gain.

It’s about asking harder questions:

- Are latency envelopes measurable and enforced?
- Is the identity policy versioned and traceable?
- Do regression gates prevent silent degradation?
- Does the Fusion Readiness Score reflect operational maturity?

Upgrade the minimum viable contract toward production-grade where feasible. Document gaps explicitly. Decide whether to scale, extend the pilot, or pause. A “go” decision should mean more than “the model improved.” It should mean the system behaves predictably under defined constraints.

Partner / Vendor Readiness Checklist

Collaborative perception doesn't scale in isolation. If you rely on vendors or external data providers, alignment must extend beyond your internal stack.

Before committing, ask:

Data Contract Discipline

- Can you provide timestamps with declared semantics and measurable sync bounds?
- Are coordinate frames versioned and transforms auditable?
- Is latency envelope performance observable in real time?

Ground Truth Governance

- How do you handle cross-agent identity reconciliation?
- Is there a written identity lifecycle policy?
- What adjudication loop exists for ambiguous cases?

Evaluation Maturity

- Do you report Cooperative Gain by scenario slice?
- Are stability and latency-aware metrics part of the release criteria?
- Are regression gates automated or manual?

Operational Health

- How is calibration drift detected and quarantined?
- How are message integrity issues surfaced in the field?
- What happens when telemetry degrades?

Weeks 11–12: Production Readiness Assessment

The final stretch isn't about squeezing out one more percentage point of gain.

It's about asking harder questions:

- Are latency envelopes measurable and enforced?
- Is the identity policy versioned and traceable?
- Do regression gates prevent silent degradation?
- Does the Fusion Readiness Score reflect operational maturity?

Upgrade the minimum viable contract toward production-grade where feasible. Document gaps explicitly. Decide whether to scale, extend the pilot, or pause. A "go" decision should mean more than "the model improved." It should mean the system behaves predictably under defined constraints.

Appendix A: Copy-Ready Templates

Fusion Readiness Score (FRS), a slice-based evaluation report, and an identity policy quick-start. These are meant to be edited, argued over, versioned, and revisited. If they feel slightly uncomfortable filling out, that's usually a good sign.

A1) Fusion Readiness Score (FRS)

The FRS is a weighted rubric designed to surface fragility early. Each category is scored from 0 to 5, with 0 meaning "undefined" and 5 meaning "measured, enforced, and monitored."
It's not meant to be gamed. In fact, a score that drops after a candid review is healthier than one that remains artificially high.

Scoring Guidance (0–5 per category)

- 0 = Not defined
- 1 = Defined informally
- 2 = Documented but not enforced
- 3 = Partially enforced/monitored
- 4 = Fully enforced with telemetry
- 5 = Enforced, monitored, versioned, and tied to regression gates

Fusion Readiness Score Table

Time Sync Bounds + Monitoring (Weight: 20)
Are sync error bounds explicitly defined?
Is drift measured per sender?
Is alerting automated?

Score: ___
Evidence: ___

Coordinate Frames + Transforms (Weight: 15)
Are frames versioned?
Are transform updates traceable?
Are sanity checks automated?

Score: ___
Evidence: ___

Calibration Versioning + Drift Detection (Weight: 15)
Is the calibration version logged per message?
Is drift detectable automatically?
Is quarantine logic implemented?

Score: ___
Evidence: ___

Ego Pose + Uncertainty Model (Weight: 10)
Is pose uncertainty modeled explicitly?
Is it used in fusion weighting?

Score: ____
Evidence: ____

Message Schema + Sanity Validation (Weight: 10)
Are required fields enforced?
Are range checks and ordering checks automated?

Score: ____
Evidence: ____

Identity Policy + Track Lifecycle (Weight: 15)
Is identity persistence documented?
Are split/merge rules versioned?
Is track QA measured continuously?

Score: ____
Evidence: ____

Traceability + Audit Trails (Weight: 10)
Can a fused decision be traced back to source messages?
Are logs complete and queryable?

Score: ____
Evidence: ____

Regression Gates + Slice Coverage (Weight: 5)
Are slice-based thresholds defined?
Is coverage drift monitored?

Score: ____
Evidence: ____

Final Fusion Readiness Score (0–100):
A score below 60 likely indicates pilot fragility.
A score between 60–80 suggests partial production readiness.
Above 80 may indicate operational maturity, assuming scoring is honest.

A2) Slice-Based Evaluation Report

This template is meant to prevent selective storytelling.

Each release should have a short, candid summary that compares baseline vs. cooperative performance by slice—not just globally.

Release ID: ___
Baseline vs. Cooperative Version: ___
Evaluation Date: ___

Robotaxi AV Stacks

For each slice:

- Slice Name: ___
- Cooperative Gain (Δ): ___
- Late Message Rate: ___
- Duplicate Rate: ___
- ID Switch Rate: ___
- Latency-to-Impact: ___
- Pass / Fail: ___

If a slice fails, document why. Don't bury it in appendix charts.

Overall Summary

- Overall Cooperative Gain (Δ): ___
- Stability Trend: Improving / Flat / Regressing
- Latency Envelope Status: Within Bounds / Exceeded

Recommendation:
Ship / Do Not Ship / Ship with Restrictions

Notes:
(Brief explanation of any regressions, trade-offs, or risk areas.)

The point of this report is clarity. If the recommendation feels ambiguous, that ambiguity likely reflects unresolved contract or stability issues.

A3) Identity Policy Quick-Start

Identity governance is often treated as secondary until fragmentation and duplicate rates spike.

This quick-start template forces teams to define identity semantics before large-scale labeling begins.

Occlusion Persistence Threshold (Per Class)

- Pedestrian: ____ seconds
- Cyclist: ____ seconds
- Passenger Vehicle: ____ seconds
- Commercial Vehicle: ____ seconds

(Define based on typical motion patterns and slice priorities.)

Reassociation Thresholds

When re-linking objects after occlusion:

- Position tolerance: ____ meters
- Velocity tolerance: ____ m/s
- Heading tolerance: ____ degrees

If multiple candidates satisfy thresholds, define tie-break rules explicitly.

Split / Merge Rule

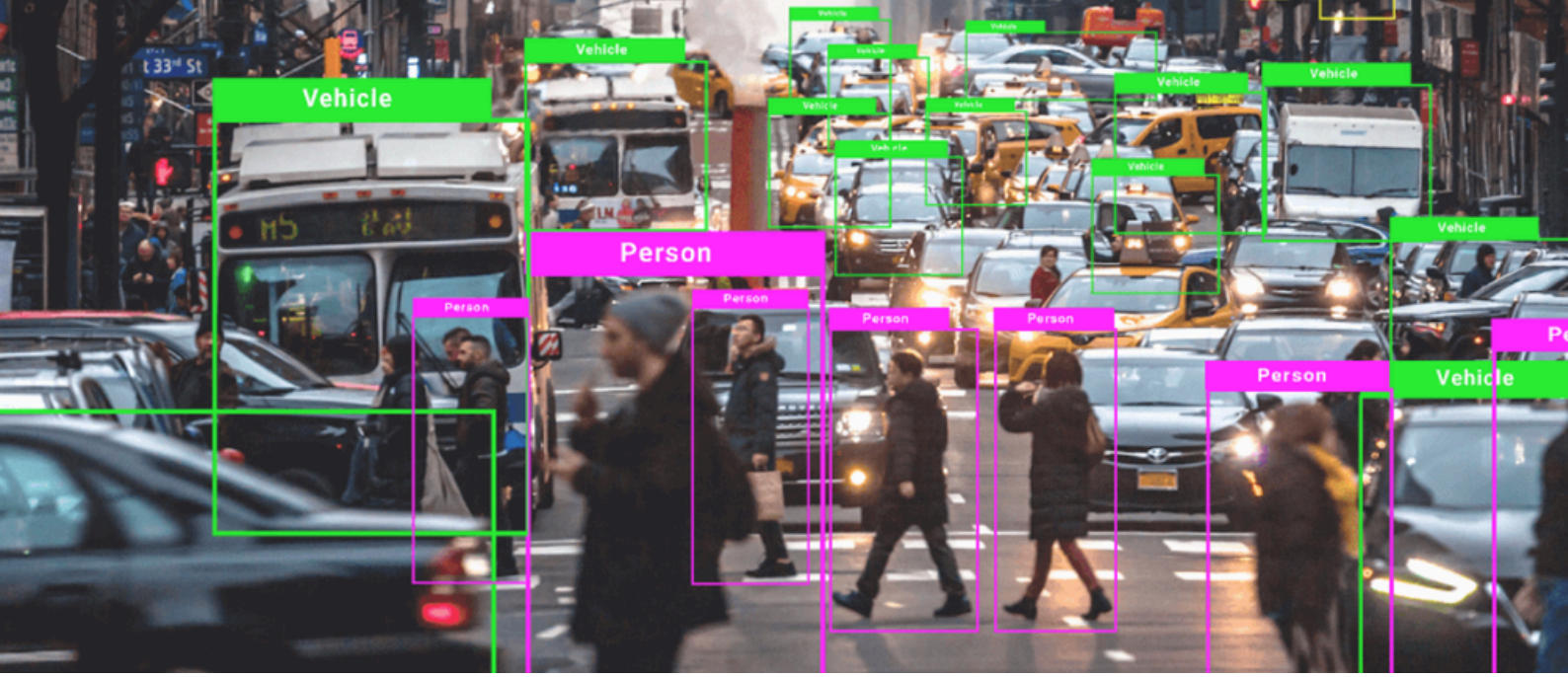
If two trajectories diverge for more than ____ frames, create a split.
If two tracks converge within ____ meters for more than ____ frames, evaluate merge.

Be explicit. Vague rules create annotator drift.

Duplicate Rule

If overlap persists within ____ meters for more than ____ frames, mark as duplicate.

Clarify whether duplicates are suppressed immediately or flagged for adjudication.



Ambiguity Rule

If identity is uncertain, tag as ambiguous and route to adjudication.
Do not force-merge.

This last rule often prevents artificial stability in labels that later destabilize fusion.

A Closing Note on Templates

If your FRS score changes unexpectedly after a minor system update, that's a signal. If slice reports repeatedly show instability in one scenario type, that's a signal. If annotators struggle to apply identity rules consistently, that's a signal.

Collaborative perception succeeds when those signals are visible early, before they show up as unexplained planner behavior in the field.

Why Choose Digital Divide Data (DDD)

Collaborative perception doesn't usually fail because of a missing model. It fails in the seams, between agents, between schemas, between identity rules, between what was assumed and what was specified. And those seams are exactly where promising cooperative gains tend to erode.

We don't build your fusion model. We make sure the data underneath it behaves.

Over the years, we've worked with AV, ADAS, and AI teams who discovered, sometimes mid-program, that multi-agent systems demand a different level of operational discipline. Single-agent annotation pipelines don't naturally extend to cross-agent identity reconciliation. Standard QA workflows don't automatically catch sync drift. Traditional evaluation dashboards often show improved recall without revealing latency-to-impact regressions.

At first, the gaps seem small. A duplicate rate rises gradually in occlusion slices. An identity persistence rule is interpreted slightly differently across labeling teams. A calibration shift goes unnoticed for weeks. The dashboard still looks acceptable. The cooperative gains are still visible. Until they aren't.

Where DDD Fits in the Stack

We operate at the data layer, the layer most collaborative programs underestimate early on.

Fusion-Ready Data Contracts

We formalize the semantics of timestamps, governance, calibration versioning, schema validation, and traceability into testable, measurable specifications. Because cooperative gains should not be accidental, they should be bounded and defensible.

Multi-Agent Ground Truth Workflows

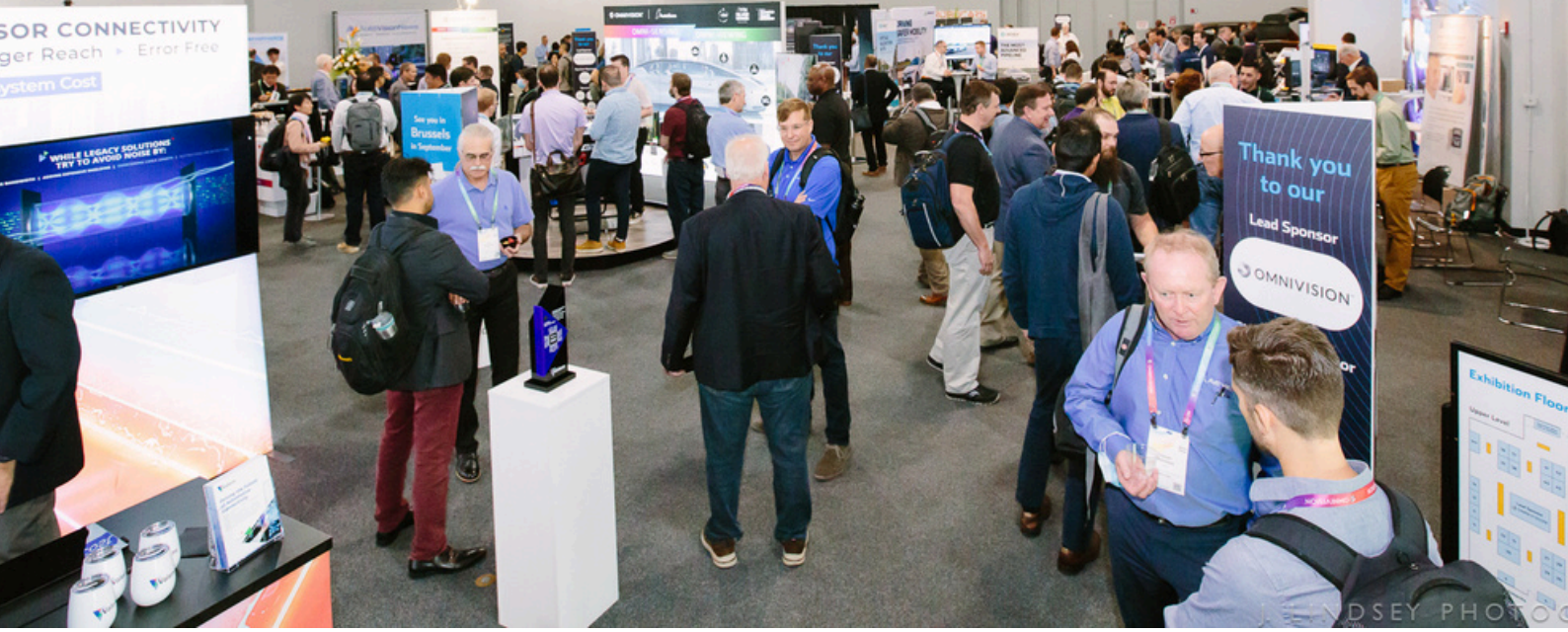
Cross-agent identity reconciliation isn't just a labeling task; it's a governance problem. We design identity policies, adjudication loops, and track QA pipelines that reduce fragmentation and preserve stability under occlusion and reappearance.

Slice-Based Evaluation Infrastructure

We operationalize cooperative gain reporting by scenario slice, pairing it with stability and latency-aware metrics. Improvements are only meaningful if they hold under the slices that matter operationally, not just in aggregate dashboards.

QA Automation & Drift Detection

We help teams implement early guardrails, sync drift scoring, calibration sanity checks, duplicate monitoring, ID switch trend analysis, so regressions are caught upstream before they cascade into perceived model instability.



Why This Matters Now

Collaborative perception is moving from research curiosity to deployment reality. Infrastructure-assisted sensing is expanding. V2X ecosystems are maturing. Multi-agent datasets are growing.

At the same time, tolerance for instability is shrinking.

At fleet scale, even a small rise in duplicate objects translates to thousands of affected scenes per day. A subtle latency envelope breach may not show up immediately in recall metrics, but it can alter planner confidence in edge cases. Sustaining cooperative gains at scale becomes less about squeezing out another percentage point of recall and more about ensuring alignment, consistency, and latency discipline are continuously enforced.

That's a data operations challenge as much as a modeling challenge.

DDD's experience in large-scale data annotation, QA governance, and production data workflows positions us directly at that intersection. We don't just help teams demonstrate improvement. We help them make it measurable, auditable, and resilient enough to survive production.

References:

Bayartsengel, M., Soleimaniamiri, S., Huang, Z., Wang, Q., & Racha, S. (2024, October). Enhancing vulnerable road user safety at signalized intersections through cooperative perception and driving automation: Final report (Report No. FHWA-HRT-24-171). Federal Highway Administration. https://doi.org/10.21949/1521615

European Telecommunications Standards Institute. (2024, November). ETSI TS 103 926 V2.1.1: Intelligent Transport Systems (ITS); Testing; Collective Perception Service (CPS); Interoperability tests specification; Release 2. https://www.etsi.org/deliver/etsi_ts/103900_103999/103926/02.01.01_60/ts_103926v020101p.pdf

Federal Communications Commission. (2024, December 13). Use of the 5.850–5.925 GHz band (Final rule; Federal Register). https://www.federalregister.gov/documents/2024/12/13/2024-28980/use-of-the-5850-5925-ghz-band

5GAA Automotive Association. (2024, May). Creating trust in connected and automated vehicles (White paper). https://5gaa.org/content/uploads/2024/05/5gaa-trust4auto-white-paper-2024.pdf

5GAA Automotive Association. (2025, January). C-V2X roadmap white paper (White paper). https://5gaa.org/content/uploads/2025/01/5gaa-wi-cv2xrm-iii-roadmap-white-paper.pdf

National Highway Traffic Safety Administration. (2024, February 15). Research on connected vehicle technology: Report to Congress. https://www.nhtsa.gov/sites/nhtsa.gov/files/2024-02/research-connected-vehicle-technology-report-to-Congress-021524.pdf

U.S. Department of Transportation. (2024, August 16). USDOT releases national deployment plan for vehicle-to-everything (V2X) technologies to reduce death and serious injuries on America's roadways (Press release). https://www.transportation.gov/briefing-room/usdot-releases-national-deployment-plan-vehicle-everything-v2x-technologies-reduce

U.S. Department of Transportation, Intelligent Transportation Systems Joint Program Office. (2024). Executive briefing: Vehicle-to-everything (V2X) technology. https://www.itskrs.its.dot.gov/sites/default/files/2025-01/executive-briefing/Executive%20Briefing%202024_Vehicle-to-Everything%20%28V2X%29%20Technology_FINAL508_0_0_0.pdf

Contact us

Ready to move collaborative perception from promising pilot to production-ready system? Let's make your data contract fusion-ready. Talk to an Expert!

www.digitaldividedata.com

Digital Divide Data