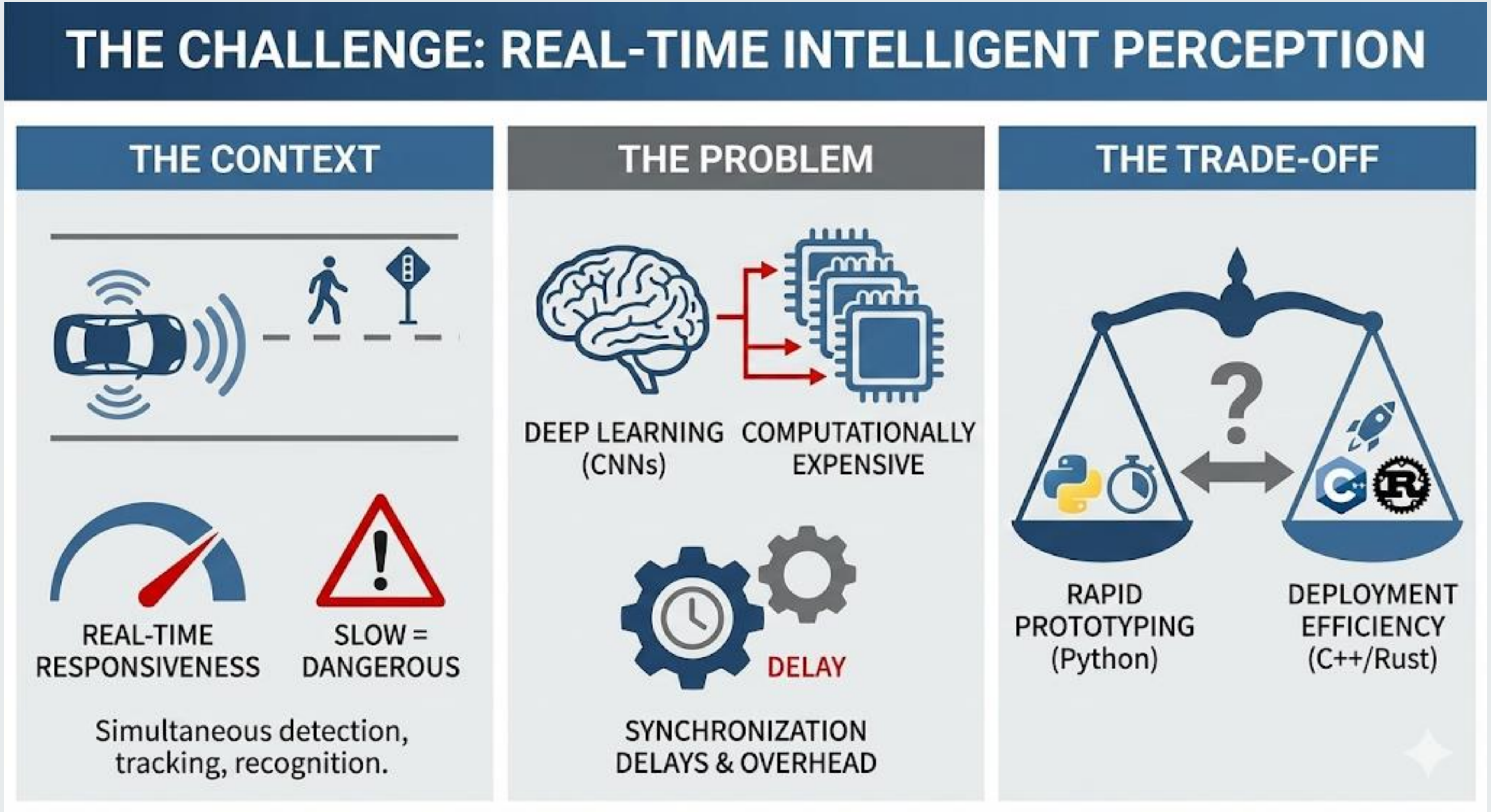
AutoVision

XLR8 PROJECT – ITSG 2025-2026

PRESENTERS: DAVID SZILAGYI, RAZVAN FILEA, ARMIN TOROK


THE CHALLENGE: REAL-TIME INTELLIGENT PERCEPTION
THE CONTEXT
REAL-TIME RESPONSIVENESS
SLOW = DANGEROUS
Simultaneous detection, tracking, recognition.
THE PROBLEM
DEEP LEARNING (CNNs)
COMPUTATIONALLY EXPENSIVE
DELAY
SYNCHRONIZATION DELAYS & OVERHEAD
THE TRADE-OFF
?
RAPID PROTOTYPING (Python)
DEPLOYMENT EFFICIENCY (C++/Rust)

Initial ML Approach: Fine-Tuning on Real-World Data (BDD100K)

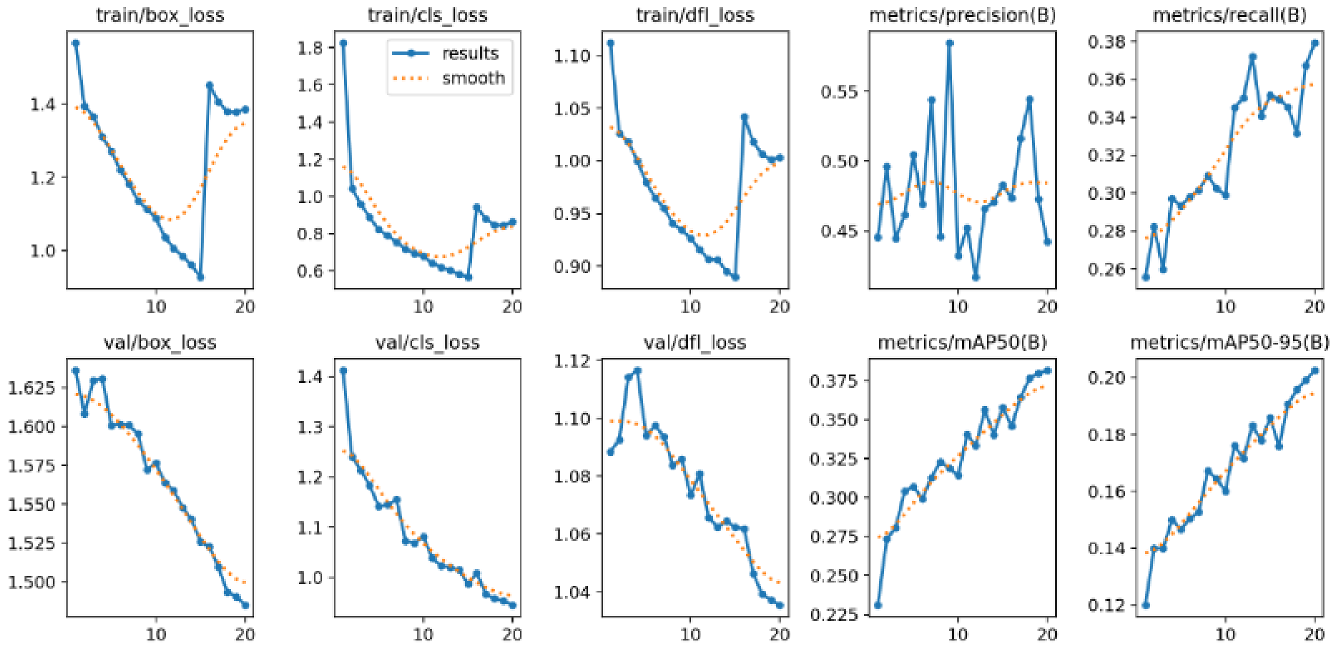


Figure 5.2: Training and validation loss evolution along with precision, recall, and mAP metrics across epochs. The late-epoch fluctuation corresponds to mosaic augmentation being disabled.

- Methodology:
  - Fine-tuned YOLO 11s (COCO-pretrained) on the Berkeley DeepDrive 100K dataset.
  - Training: 20 epochs, AdamW optimizer, 640×640 resolution, focused on traffic lights and signs.
- Theoretical Performance:
  - Achieved stable convergence with validation mAP@0.5 of 0.38 .
  - High precision for vehicles (0.90) and moderate precision for lights/signs in urban images. (0.53, 0.56)
- The "Sim-to-Real" Failure:
  - Critical Domain Gap: The model failed when deployed in the 1:10 scale in-office environment.
  - False Positives: Background objects (chairs, fire extinguishers) were frequently misclassified as traffic signals .

Expectation meets Reality: The Sim-to-Real Gap

Figure 5.6: Qualitative detection results on the BDD100K test set using the fine-tuned YOLO 11s model.

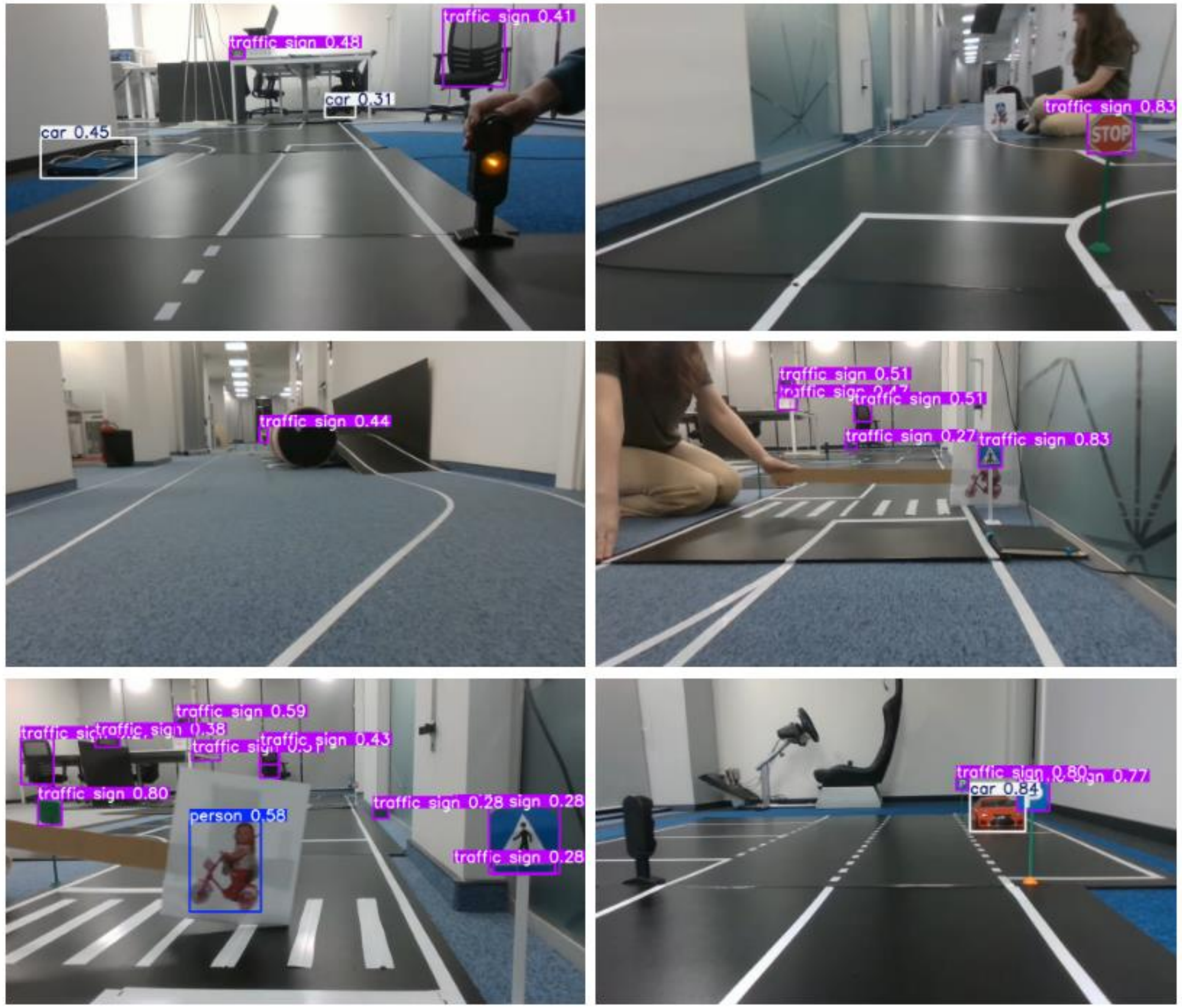


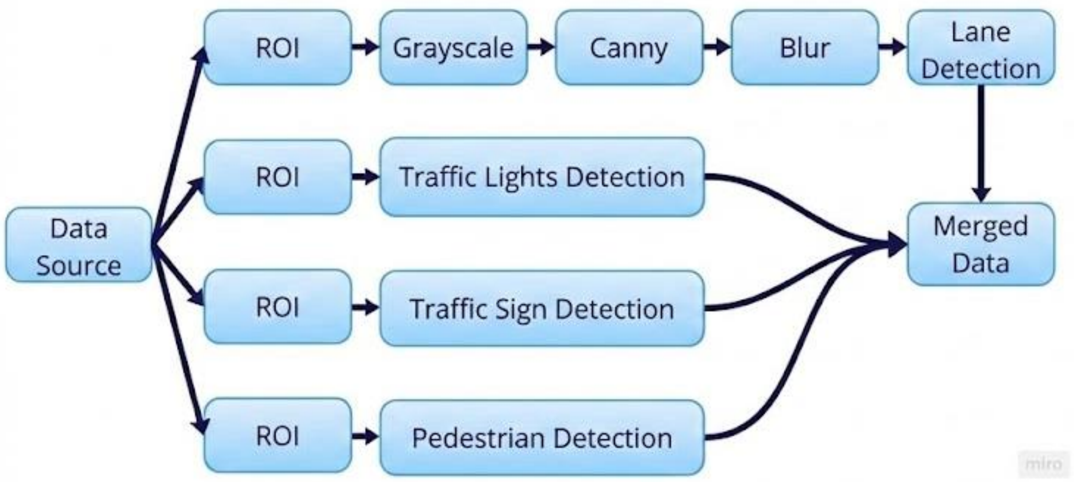
Figure 5.7: Detection performance in the real test environment.

The Pivot to Sensor-Native Data

- Data Strategy:
  - Abandoned large-scale external datasets in favor of a **Handcrafted Dataset** captured directly from the vehicle's onboard sensors.
  - **Data Collection:** Manual (RC-mode) driving sessions to capture exact lighting and camera angles.
- Data Pipeline:
  - Applied custom augmentations: Brightness jitter, Gaussian noise, and rotation to simulate sensor noise and diversity .

Model Specialization

- Architectural Change:
  - Replaced the monolithic model with **three specialized YOLO models** + a **geometric Lane Detector**
  - Why Specialization?
    - **Direct Classification:** Detects states (e.g., Red/Green) immediately, eliminating the need for a secondary classification stage
    - **No Compromises:** Avoids trade-offs in resolution and anchors by independently tuning for disparate objects
  - Key Result:
    - Significantly mitigated background false positives and achieved robust detection of miniature props in real-time.



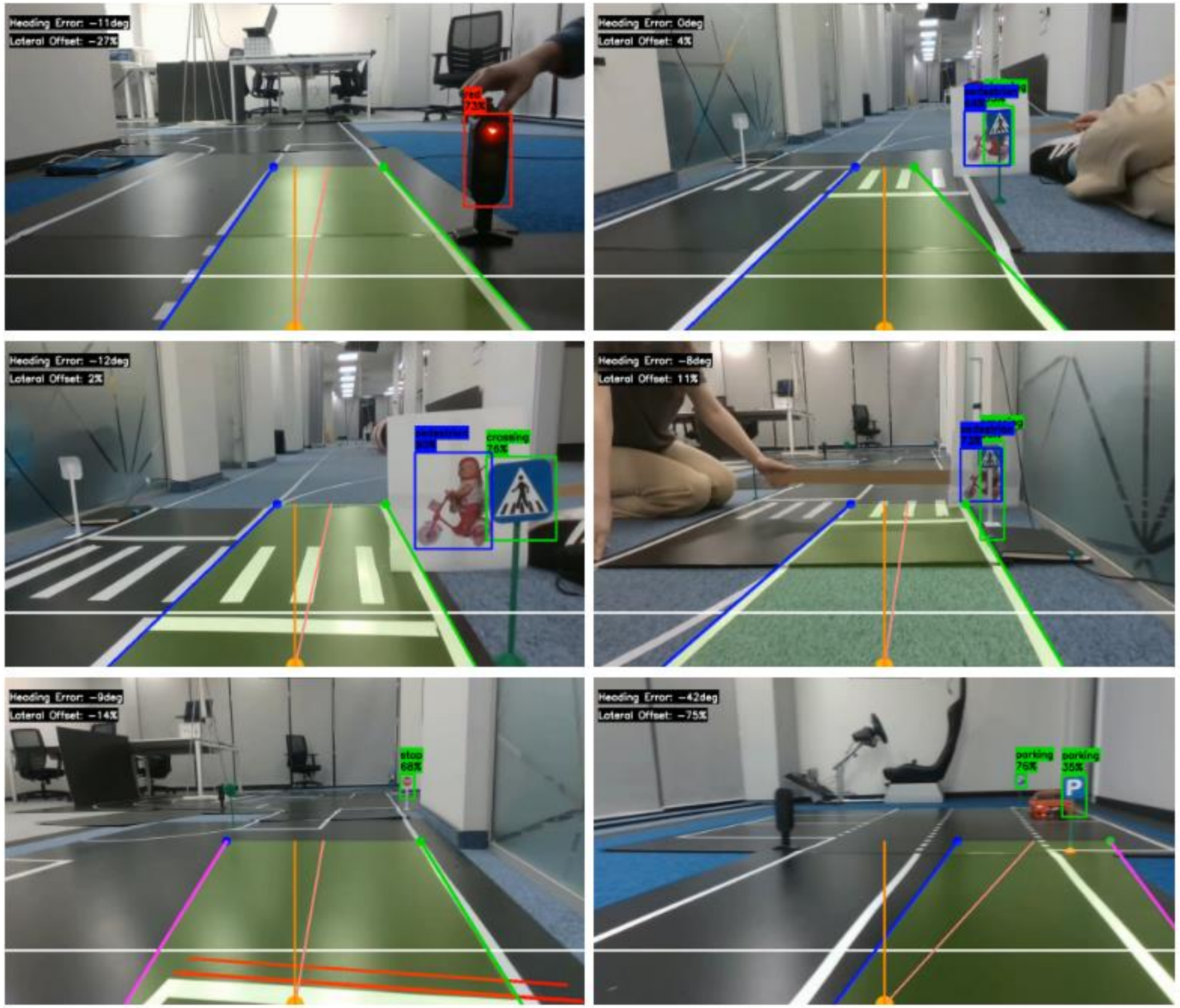


Figure 5.9: Qualitative detections in the scaled test environment after pretraining on the handcrafted dataset.

The New Bottleneck: Concurrency & Latency

The Consequence of Specialization:

We moved from 1 model to **4 concurrent perception branches,** forcing parallelism for performance

Python's Limitation:

Standard Python multiprocessing struggles with **high-throughput video streams .**

Leads to **high latency and memory overhead .**

The Timing Requirement:

Fast branches (ex Lane Detection, ~6ms) must **update continuously** for steering control .

Slow branches (ex Sign Recognition, ~28ms) must **not stall** the entire system .

The Shared-Message Specification

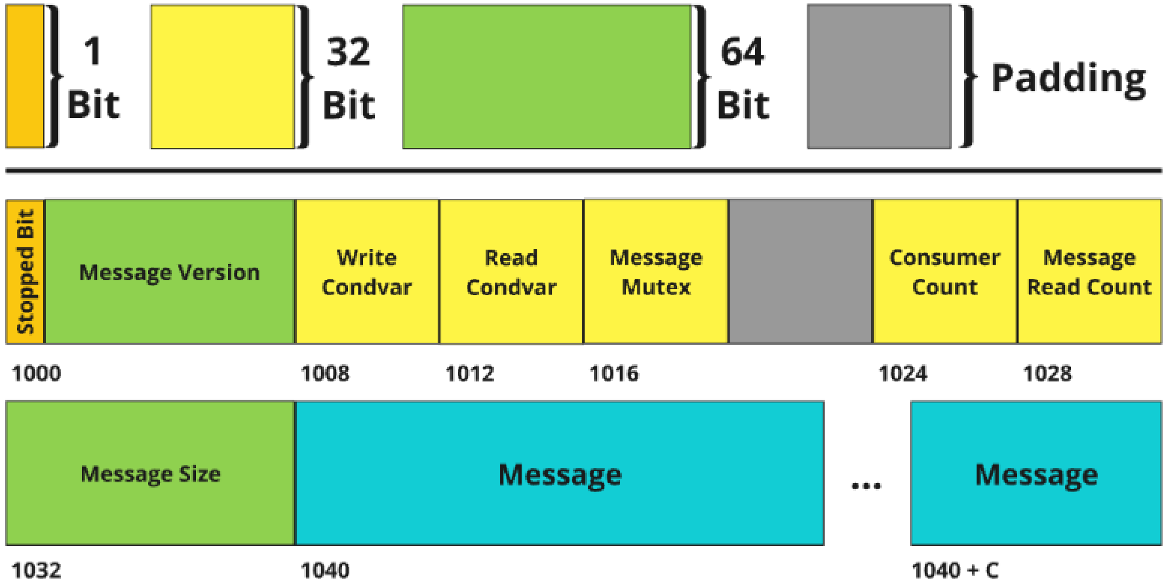


Figure 4.1: Memory layout of `SharedMessage`: fields are aligned to enable atomic access to *Message Version*, while the *Mutex* and condition variables reside in shared memory.

Table 1. Write-Read Average Transfer Time (ms) 50.000 iterations

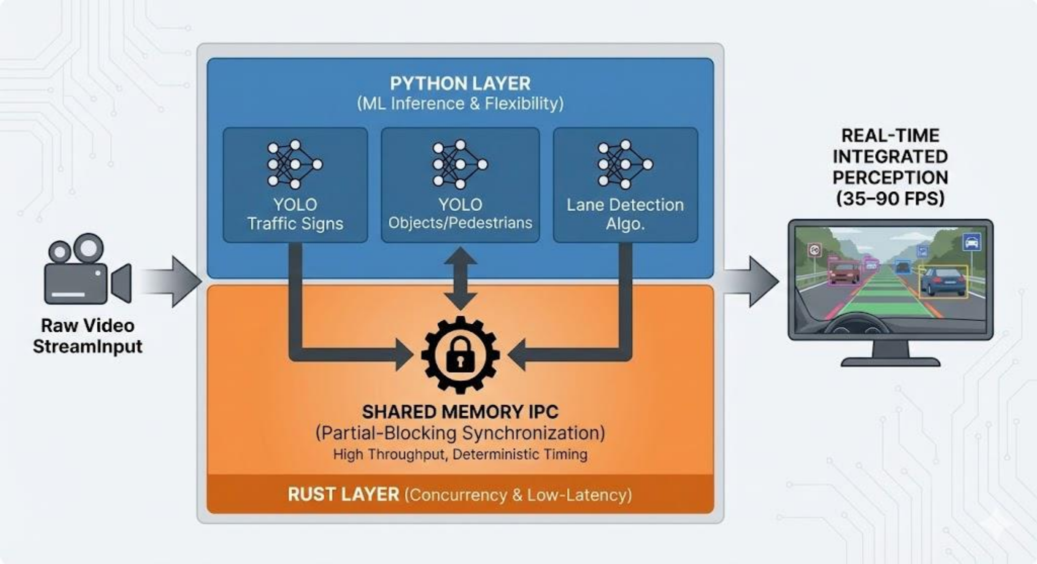
Resolution	Size	MP-Pipe	RS-IPC	SHM-LOCK	SHM-PTH
256x256	576 KB	0.247	0.2	0.226	0.233
512x512	2.25 MB	0.908	0.728	0.796	0.82
640x480	2.64 MB	1.019	0.898	0.982	0.994
1280x720	7.91 MB	5.542	4.529	4.762	4.833
1920x1080	17.80 MB	16.161	11.26	11.35	11.96

Table 2. Write-Read Total Transfer Time (s) 50.000 iterations

Resolution	Size	MP Pipe	RS-IPC	SHM-LOCK	SHM-PTH
256x256	576 KB	12.336	10.01	11.306	11.67
512x512	2.25 MB	45.411	36.408	39.789	40.986
640x480	2.64 MB	50.931	44.895	49.12	49.986
1280x720	7.91 MB	277.103	226.434	238.106	241.671
1920x1080	17.80 MB	808.053	562.98	567.252	598.005

Proposed Solution – Hybrid Architecture

- **Concept:** A "SharedMessage" infrastructure using Rust for low-level memory management
- **Design:**
  - **Python Layer:** Handles ML inference logging.
  - **Rust Layer:** Handles atomic synchronization, shared memory, and data flow and more
- **Benefit:** Enables high throughput without sacrificing the simplicity of Python development.



The Core Algorithm – Partial-Blocking IPC

The Communication Model.

- A Producer-Consumer strategy tailored for robotics.

The Innovation: "Partial-Blocking" policy.

- Producer (Camera): Advances to the next frame once at least one consumer started processing it

Why “Partial-Blocking”?

Non-Blocking: Risks Desynchronization (input mismatch)

Full-Blocking: Slowest model bottlenecks whole pipeline

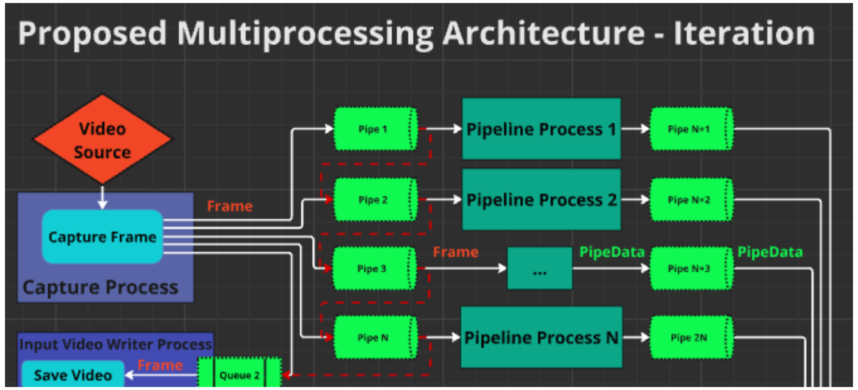
Partial-Blocking: Ensures frame consistency while allowing fast branches to run at maximum speed

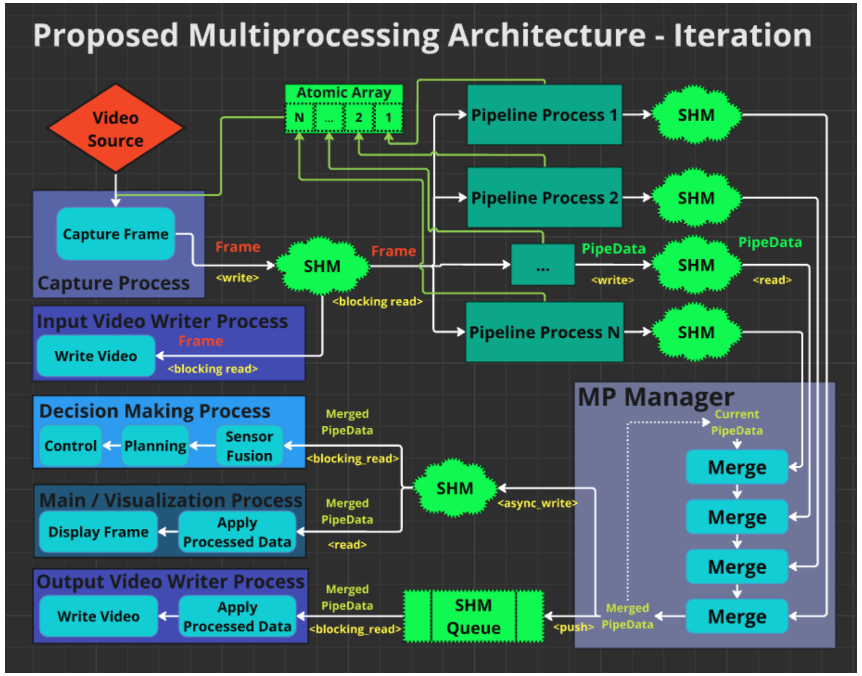
The Outcome.

- Fast branches (lane detection) update frequently; slow branches (sign recognition) catch up asynchronously without stalling the system.

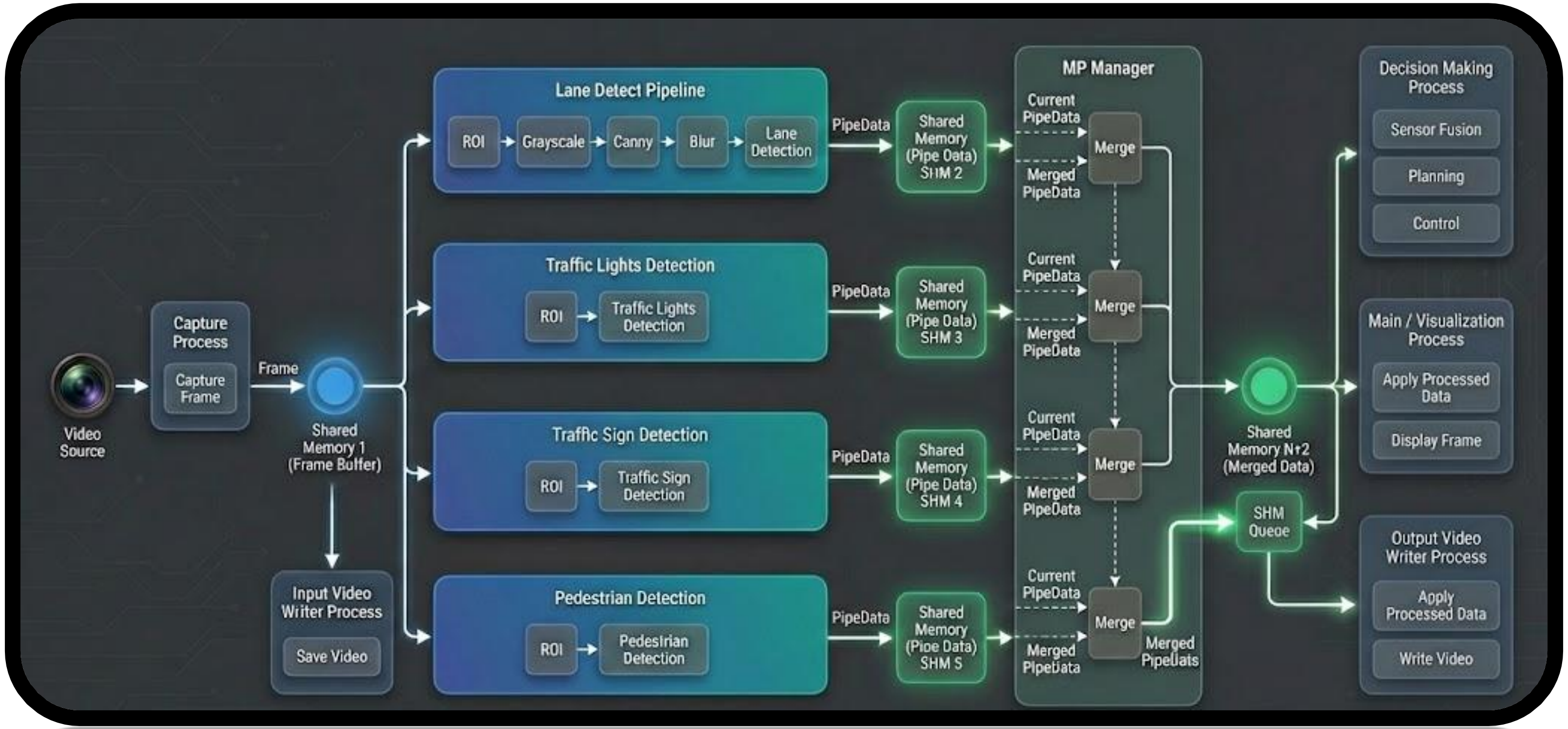
Hybrid Python–Rust Architecture

- Design Philosophy:
  - Python: Retained for ML development, model loading, and visualization.
  - Rust: Handles low-level memory management, atomic synchronization, and data flow.
- Mechanism:
  - Modules communicate via a Shared-Memory (IPC) layer.
  - Ensures zero-copy data access where possible to maximize throughput.
- Benefit:
  - Allows models to be retrained/swapped in Python without modifying the compiled Rust infrastructure .

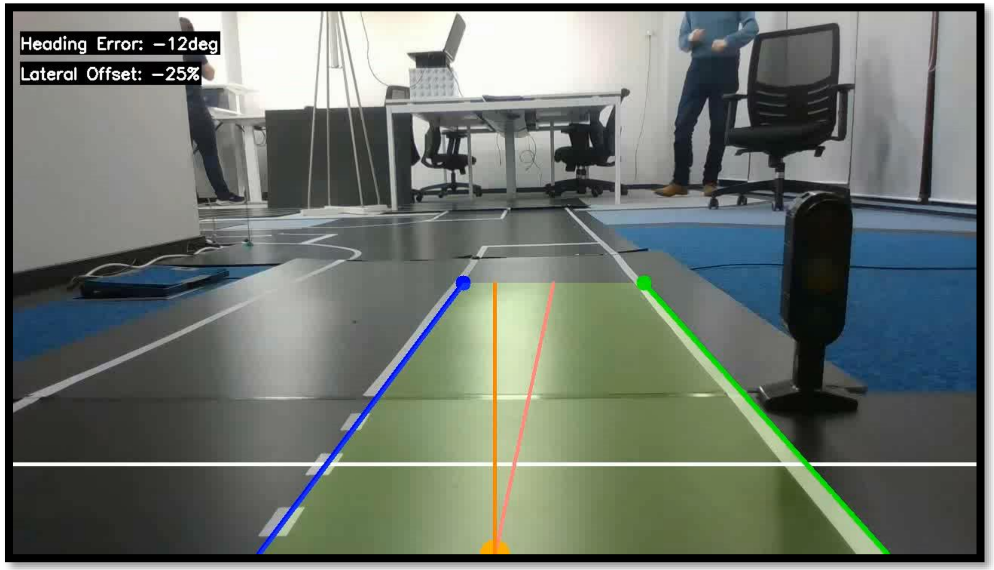




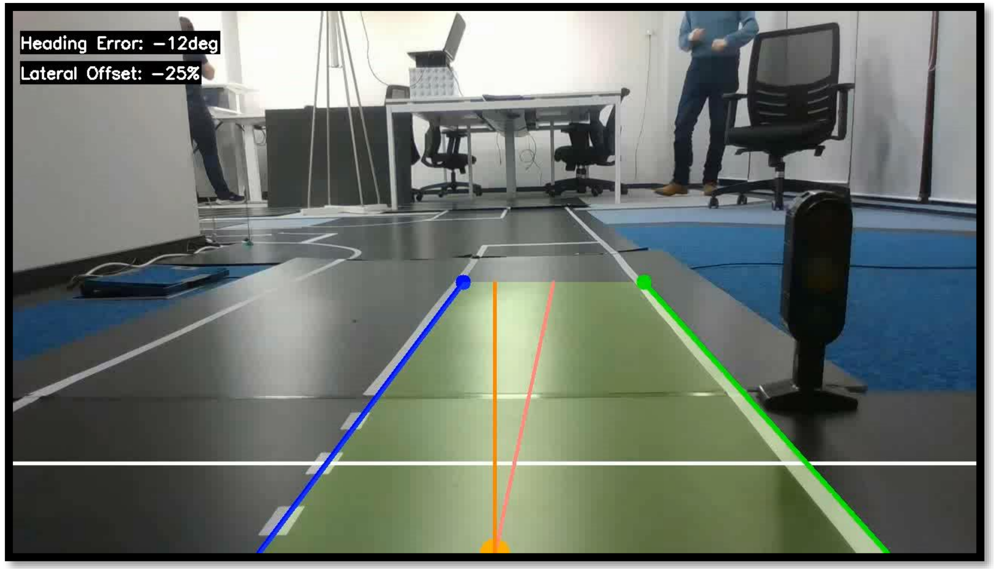
DEMO

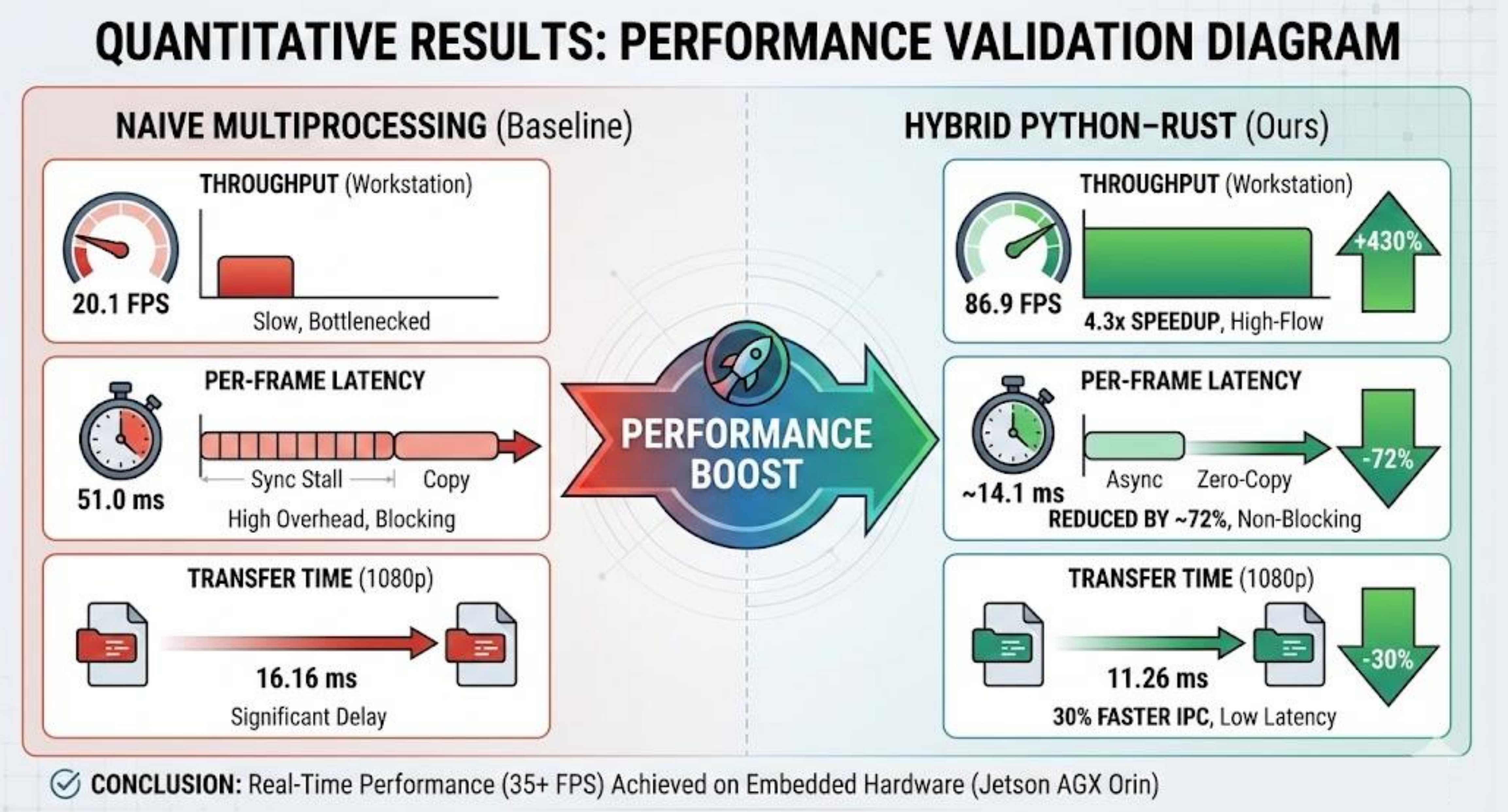

Video Source
Capture Process
Capture Frame
Frame
Shared Memory 1 (Frame Buffer)
Input Video Writer Process
Save Video
Lane Detect Pipeline
ROI
Grayscale
Canny
Blur
Lane Detection
Traffic Lights Detection
ROI
Traffic Lights Detection
Traffic Sign Detection
ROI
Traffic Sign Detection
Pedestrian Detection
ROI
Pedestrian Detection
PipeData
Shared Memory (Pipe Data) SHM 2
PipeData
Shared Memory (Pipe Data) SHM 3
PipeData
Shared Memory (Pipe Data) SHM 4
PipeData
Shared Memory (Pioe Data) SHM S
MP Manager
Current PipeData
Merge
Merged PipeData
Current PipeData
Merge
Merged PipeData
Current PipeData
Merge
Merged PipeData
Current PipeData
Merge
Merged PipeData
Merged PipeDats
Shared Memory N+2 (Merged Data)
SHM Queoe
Decision Making Process
Sensor Fusion
Planning
Control
Main / Visualization Process
Apply Processed Data
Display Frame
Output Video Writer Process
Apply Processed Data
Write Video

Full-blocking Approach
With standard IPC
(~20FPS)

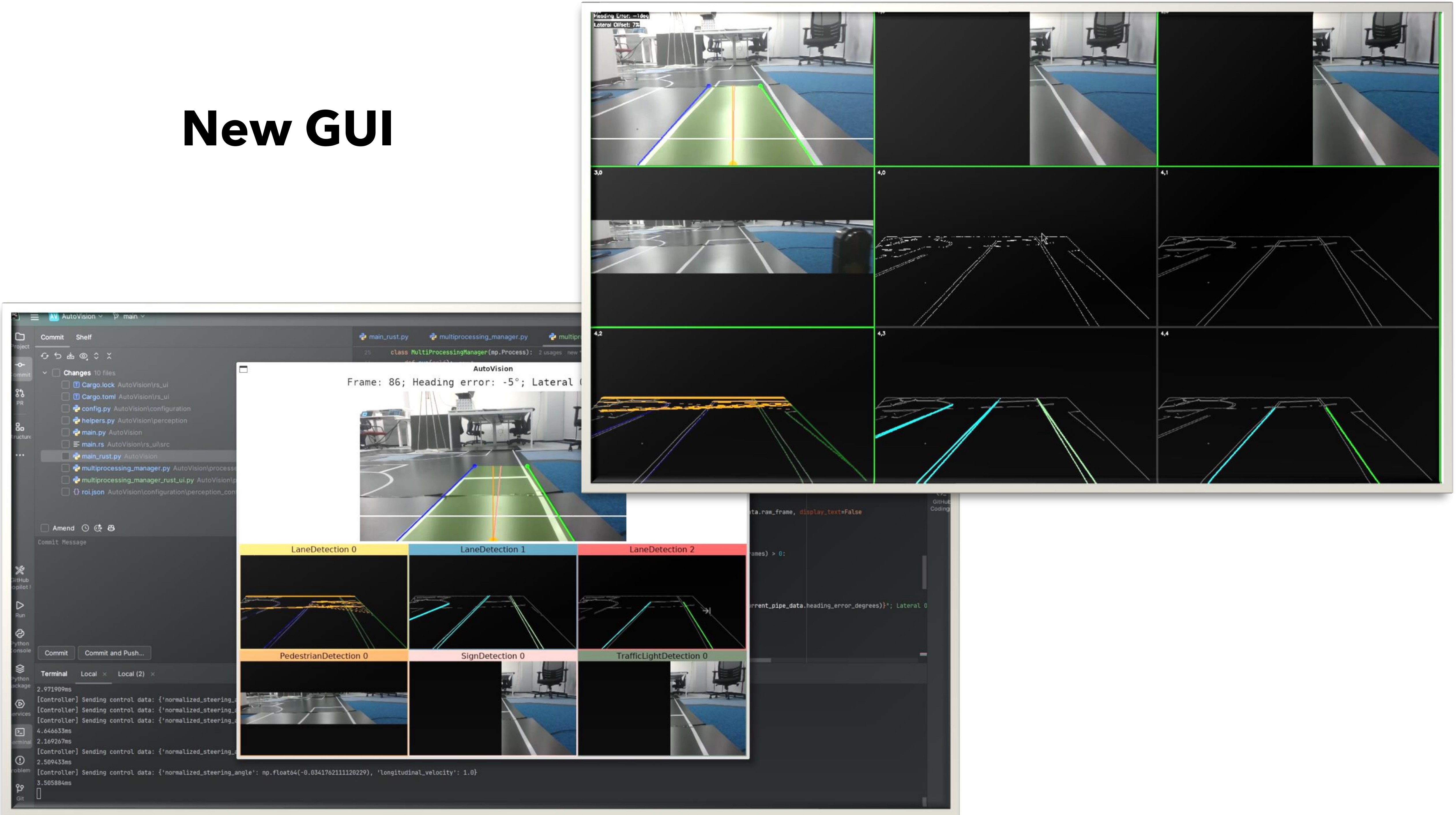


Proposed Shared-Memory
Partial-Blocking Architecture
(~90FPS)




QUANTITATIVE RESULTS: PERFORMANCE VALIDATION DIAGRAM
NAIVE MULTIPROCESSING (Baseline)
THROUGHPUT (Workstation)
20.1 FPS
Slow, Bottlenecked
PER-FRAME LATENCY
51.0 ms
Sync Stall
Copy
High Overhead, Blocking
TRANSFER TIME (1080p)
16.16 ms
Significant Delay
PERFORMANCE BOOST
HYBRID PYTHON–RUST (Ours)
THROUGHPUT (Workstation)
86.9 FPS
4.3x SPEEDUP, High-Flow
+430%
PER-FRAME LATENCY
~14.1 ms
Async
Zero-Copy
REDUCED BY ~72%, Non-Blocking
-72%
TRANSFER TIME (1080p)
11.26 ms
30% FASTER IPC, Low Latency
-30%
CONCLUSION: Real-Time Performance (35+ FPS) Achieved on Embedded Hardware (Jetson AGX Orin)

New GUI


AutoVision
Frame: 86; Heading error: -5°; Lateral
LaneDetection 0
LaneDetection 1
LaneDetection 2
PedestrianDetection 0
SignDetection 0
TrafficLightDetection 0
Commit
Shelf
Changes 10 files
Cargo.lock
Cargo.toml
config.py
helpers.py
main.py
main.rs
main_rust.py
multiprocessing_manager.py
multiprocessing_manager_rust_ui.py
roi.json
Amend
Commit Message
Commit and Push...
Terminal
Local
Local (2)
[Controller] Sending control data: {'normalized_steering_angle': np.float64(-0.0341762111120229), 'longitudinal_velocity': 1.0}
3.505884ms

SWOT ANALYSIS: HYBRID PERCEPTION ARCHITECTURE

STRENGTHS

- Modular pipeline
- High efficiency (Rust)
- Robust domain-specific detection

WEAKNESSES

- Manual annotation is labor-intensive
- Limited generalization outside the lab

OPPORTUNITIES

- Synthetic data generation
- Integration of Vision-Language Models (VLMs)

THREATS

- Rapid evolution of VLM architectures making YOLO obsolete

Conclusion & Future Work

- Summary:
  - We introduced a modular architecture combining **Python's flexibility** with **Rust's safety and efficiency**.
  - The **Partial-Blocking** strategy successfully balances throughput and consistency.
- Impact:
  - **Validated** real-time performance on embedded devices, providing a scalable **foundation** for **robotics research**
  - Companion Paper published in 2025 RRIA, Romanian Journal of Information Technology and Automatic Control
- Future Work:
  - **Next-Gen Perception:** Future iterations will explore integrating **Vision-Language Models (VLMs)** for semantic reasoning and adapting the runtime for the complex control stacks of **Humanoid Robots**.

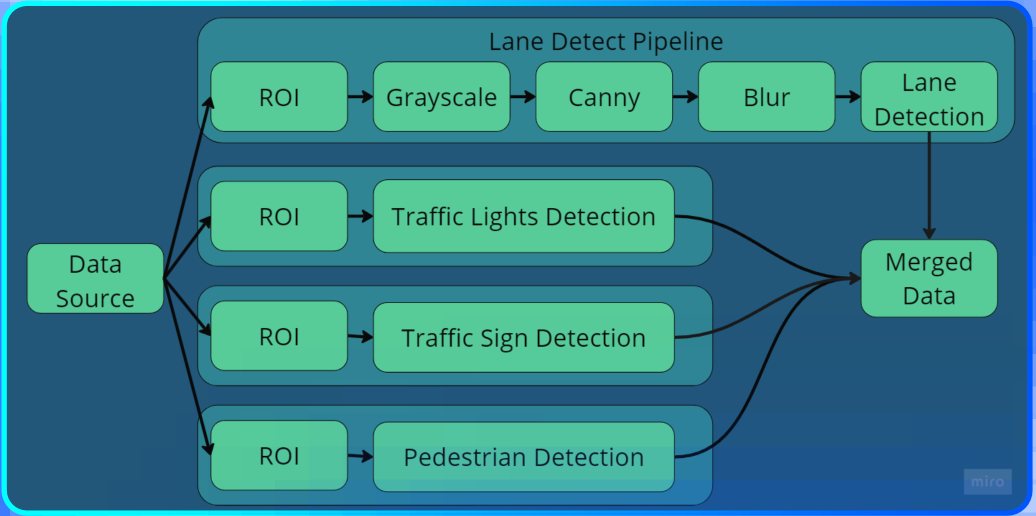
THANK YOU

Questions? | XLR8 Project

Perception System Multi-Processing Pipeline

```
{
    "roi": {
        "roi_type": "signs",
        "visualize": false
    },
    "signs_detect": {
        "model": "best_signs_close.pt",
        "visualize": true
    }
},
{
    "roi": {
        "roi_type": "traffic_lights",
        "visualize": false
    },
    "traffic_light_detect": {
        "model": "best_lights.pt",
        "visualize": true
    }
},
{
    "roi": {
        "roi_type": "pedestrians",
        "visualize": false
    },
    "pedestrian_detect": {
        "model": "best_pedestrian.pt",
        "visualize": true
    }
```

```
    {
        "grayscale": {
            "visualize": false
        },
        "canny_edge": {
            "low_threshold": 150,
            "high_threshold": 250,
            "visualize": false
        },
        "blur": {
            "kernel_size": 7,
            "sigmaX": 1,
            "visualize": false
        },
        "dilation": {
            "kernel_size": 1,
            "iterations": 1,
            "visualize": false
        },
        "roi": {
            "roi_type": "lines",
            "visualize": true
        },
        "lane_detect": {
            "visualize": false
        },
        "heading_error": {
            "visualize": false
        }
    }
```

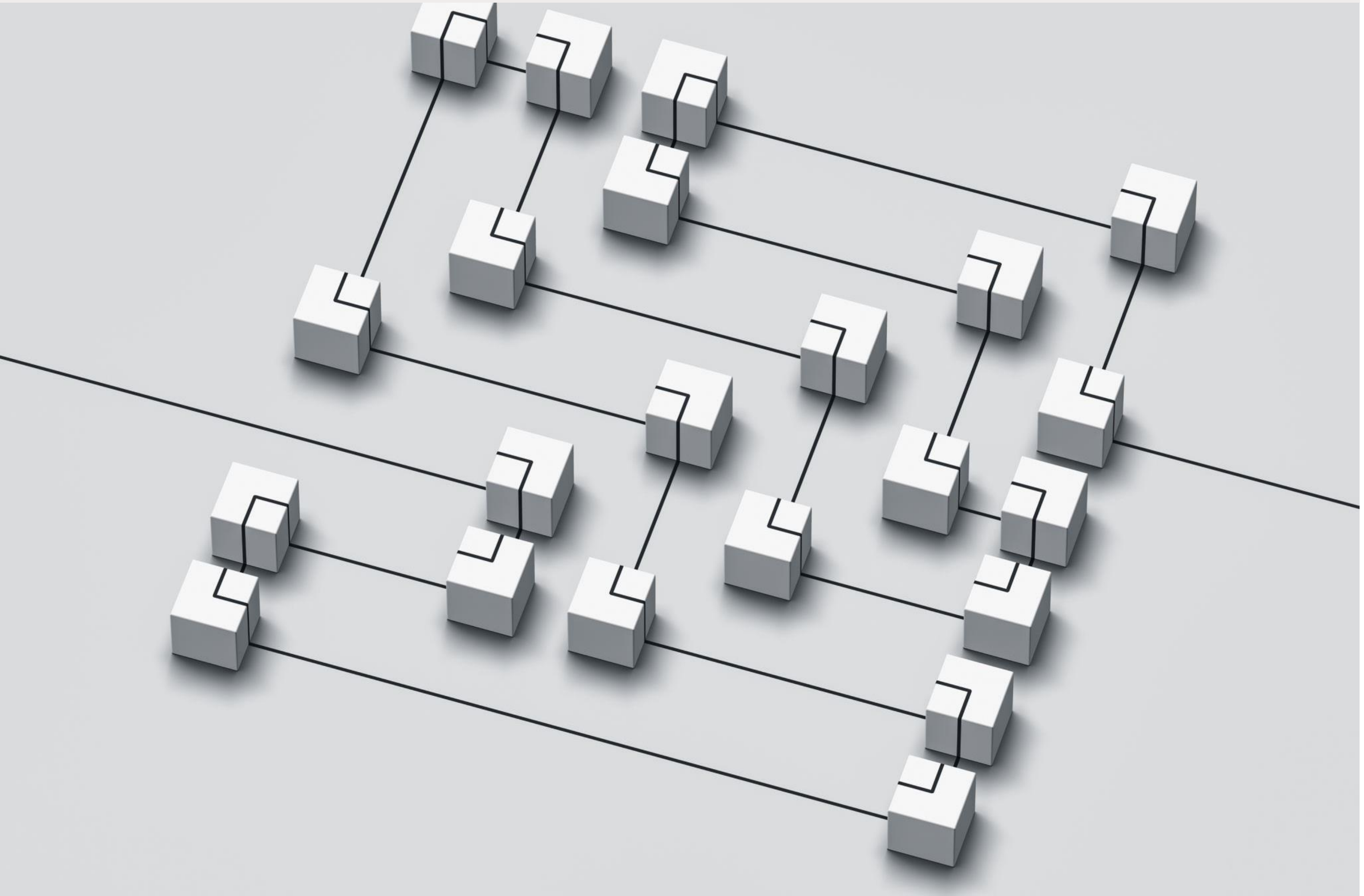


Modular & Scalable Data Processing Pipeline

JSON Based Configuration of Filters and their Positioning

Ideal for Parallelized Workflows such as Autonomous Driving Perception Systems

Performance Validation



- Throughput:
  - The Shared-Memory architecture achieved 86.9 FPS on a standard workstation, a 4.3x speedup over the naive approach (20.1 FPS).
  - Achieved real-time performance (35+ FPS) even on embedded hardware (Jetson AGX Orin).
- Latency Reduction:
  - Total per-frame iteration time reduced by ~72% by eliminating synchronization bottlenecks
  - Raw IPC transfer time for 1080p frames improved by 30% (16.16ms → 11.26ms).

The Challenge: Real-Time Intelligent Perception

- **The Context:**
  - Autonomous driving requires simultaneous lane detection, object tracking, sign recognition and more.
  - Safety depends on real-time responsiveness; a slow accurate detector is dangerous.
- **The Problem:**
  - Deep Learning (CNNs) is computationally expensive.
  - Executing multiple ML models concurrently introduces synchronization delays and overhead.
- **The Trade-off:**
  - Researchers must choose between **Rapid Prototyping** (Python) and **Deployment Efficiency** (C++/Rust) .