2026 IEEE 4th International Conference on Mobility, Operations, Services and Technologies (MOST) | 979-8-3315-6182-6/26/$31.00 ©2026 IEEE | DOI: 10.1109/MOST69733.2026.00015

LightAD: A Lightweight Vision-Based Autonomous Driving System

Yunge Li*, Zhaodong Zhou†, Shaibal Saha*, Ali Irshayyid†, Keer Chen†, Lanyu Xu*, Jun Chen†
*Department of Computer Science & Engineering
†Department of Electrical & Computer Engineering
Oakland University, Rochester Hills, USA
{yungeli, zhaodongzhou, shaibalsaha, aliirshayyid, keerchen, lxu, junchen}@oakland.edu

***Abstract*—Autonomous driving systems deployed in the real world suffer from excessive hardware redundancy and prohibitive deployment barriers. To address these challenges, *LightAD* is proposed as a lightweight, vision-based autonomous driving system framework tailored for simple, low-speed environments such as confined work zones and low-speed logistics. Diverging from industrial-grade solutions that rely on costly proprietary sensors and high-performance computing, *LightAD* implements system-level simplification. By leveraging consumer-grade hardware and lightweight algorithms for perception, planning, and control, *LightAD* significantly reduces costs and integration complexity while satisfying necessary autonomous driving functionality. Field trials at the Intelligent Ground Vehicle Competition on a physical vehicle confirmed the feasibility of *LightAD* and revealed practical implementation lessons. These results indicate that by combining low-cost general-purpose hardware with lightweight algorithms, *LightAD* effectively supports fundamental autonomous driving functionality in simple, well-defined, and low-speed environments, offering a feasible engineering pathway for cost-effective deployment of autonomous driving.**

***Index Terms*—autonomous driving system, lightweight, low-cost.**

I. Introduction

In recent years, significant progress has been made in autonomous driving technology, which is widely considered a core component of future intelligent transportation systems. Current research and development efforts are focused on addressing the challenges of complex urban open-road scenarios. To ensure safety in these dynamic, intricate environments, existing autonomous driving systems typically use high-performance hardware. These systems, such as Autoware [1] and Baidu Apollo [2], commonly incorporate expensive multi-beam LiDAR, high-precision RTK-GNSS positioning equipment, and high-performance automotive-grade computing platforms. While such expensive hardware is essential for navigating complex urban traffic, it often leads to inefficient resource utilization in many simple application scenarios, such as commuting within enclosed campuses, low-speed logistics inside factories, and last-mile delivery in university grounds. In these relatively well-defined environments with simple traffic flow, the direct application of industrial-grade systems not only imposes substantial hardware costs but also increases system maintenance complexity and energy consumption. This constitutes a significant barrier to the widespread deployment of autonomous driving technology in such specific domains.

Although reducing the cost of autonomous driving systems has emerged as an important research direction, there remains a lack of practical solutions tailored to simple scenarios. Most existing low-cost autonomous driving platforms are small model vehicles developed for educational purposes [3]–[7] or to validate some algorithms [8], [9]. These systems are often designed for idealized indoor environments, assuming stable lighting conditions and flat road surfaces. However, real-world applications, such as logistics or campus operations, often operate in outdoor environments and face challenges such as varying lighting conditions, shadows, and unstructured road surfaces. Existing educational-grade platforms lack the robustness required to handle such practical disturbances. Consequently, a significant research gap persists. Existing solutions are either expensive industrial-grade systems or functionally limited educational platforms. A low-cost system framework that maintains reliable performance in simple, well-defined, and low-speed outdoor scenarios is still missing.

To address this gap, this paper introduces *LightAD*, a lightweight vision-based autonomous driving system framework specifically designed for simple scenarios such as enclosed campuses and low-speed roads. The core objective is to optimize hardware configurations and algorithms based on scenario requirements. Specifically, cost-intensive hardware components such as LiDAR and high-precision positioning modules are removed because they are considered unnecessary for simple scenarios. Instead, consumer-grade depth cameras are employed as the primary environmental perception sensors, and general-purpose laptops or edge devices serve as the computing platform. In terms of software algorithms, computationally intensive complex models are avoided. Instead, it is demonstrated that in low-speed scenarios, efficient geometric control algorithms, such as the Pure Pursuit controller [10] and Stanley controller [11], combined with lightweight perception networks [12]–[15], are sufficient to meet basic autonomous driving demands. This design approach ensures that autonomous driving system functionality is maintained while significantly reducing hardware costs and computational resource requirements.

The main contributions of this paper are summarized as follows:

979-8-3315-6182-6/26/$31.00 ©2026 IEEE
DOI 10.1109/MOST69733.2026.00015

- We propose *LightAD*, a vision-based system framework tailored for simple, well-defined, and low-speed autonomous driving environments. By substituting expensive hardware with commodity components, we significantly reduce the overall system cost while retaining essential autonomous driving functionalities.
- We employ lightweight algorithms to replace computationally intensive models in the software stack. This strategic substitution reduces computational demand, enabling *LightAD* to support basic autonomous tasks with limited computing resources effectively.
- We implement and deploy *LightAD* on a physical vehicle and validate its feasibility through the Intelligent Ground Vehicle Competition (IGVC). Furthermore, we synthesize the lessons learned from this implementation to provide a practical reference for the future development of similar cost-effective autonomous driving platforms.

II. Related Work

A. Autonomous Driving System Architecture

In today's field of autonomous driving research and application, full-stack open-source frameworks represented by Autoware [1] and Baidu Apollo [2] dominate the landscape. Autoware, one of the earliest and most widely used open-source platforms based on ROS (Robot Operating System), offers a comprehensive set of software modules covering localization, detection, prediction, and path planning, aiming to provide general solutions for complex urban scenarios. At the same time, Baidu Apollo represents a more mature industrial-grade solution, supporting flexible hardware architectures and integrating high-definition maps, cloud-based simulation, and advanced deep learning-based perception models, capable of handling highly complex mixed traffic flows.

Although these full-stack systems are functionally robust and well-supported, they face significant challenges in practical deployment due to being "over-engineered". To support complex sensor fusion (such as multimodal fusion of LiDAR and cameras) and high-dimensional optimization planning algorithms, these frameworks typically impose significant hardware resource demands, often relying on expensive automotive-grade computing platforms (such as NVIDIA Orin) and costly sensor suites. For systems designed to perform simple tasks (such as patrols in closed campuses or low-speed last-mile delivery), directly adopting such architectures not only results in severe computational redundancy but also introduces prohibitive hardware costs and power consumption, making them difficult to deploy at scale on low-cost or edge computing devices.

To address the high cost and technical barriers, the academic community has developed a range of lightweight, miniature autonomous driving platforms focused on education and algorithm validation. Duckietown [3] is a representative project originating at MIT that builds a scaled-down urban environment where vehicles are equipped only with monocular cameras and embedded processors such as Raspberry Pi, primarily using classical computer vision algorithms for lane keeping and other tasks. F1TENTH [9] focuses more on high-speed racing scenarios, utilizing model cars equipped with 2D LiDAR to tackle reactive planning and dynamic control under extreme conditions. Additionally, Donkey Car [7], a community-driven DIY platform, is widely used for introductory research in end-to-end imitation learning, enabling developers to implement basic autonomous driving functions with minimal code and low-cost hardware. However, these lightweight systems exhibit apparent limitations in engineering practicality. As their design is primarily intended for teaching or competition, these platforms typically operate in highly well-defined and idealized environments (such as uniform artificial lighting, high-contrast lane markings, or smooth track surfaces). Once removed from these specific experimental conditions and placed in outdoor unstructured environments with drastic lighting changes, complex ground textures, or shadow interference, these systems, built on simplistic assumptions, often lack sufficient robustness to meet real-world application demands.

B. Vision-Based Perception

The selection of environmental perception algorithms directly determines the response speed and reliability of autonomous driving systems. In research focused on high detection accuracy, two-stage detectors, such as Faster R-CNN [16] and Mask R-CNN [17], have established important performance benchmarks in complex scenarios by introducing Region Proposal Networks. Additionally, with the development of deep learning, models based on the Transformer architecture, such as DINO [18], have incorporated self-attention mechanisms and demonstrated strong performance in handling occlusion and long-range dependencies. For lane segmentation, SCNN [19] introduced a slice convolution scheme that effectively captures the elongated spatial structure of lane lines. However, these high-precision models typically require significant computational resources and memory. For example, specific convolutional operations in SCNN and attention calculations in DINO can result in substantial inference latency, presenting significant challenges when deploying these models on edge computing devices with limited processing power, such as embedded development boards or standard laptops.

To achieve real-time perception with constrained hardware resources, lightweight architectures and multi-task learning have become mainstream choices for low-cost autonomous driving. For object detection, the YOLO series [12], [13] adopts a single-stage detection paradigm, significantly improving inference speed while maintaining good accuracy. In the field of semantic segmentation, ENet [20] employs an asymmetric encoder-decoder design specifically for latency-sensitive mobile tasks. To further enhance computational efficiency, end-to-end multi-task perception networks such as YOLOP [14] and HybridNets [21] have been proposed. These models typically follow a shared backbone and multiple output heads architecture, enabling the system to perform multiple tasks, such as object detection, lane segmentation, and drivable area segmentation, simultaneously after running feature

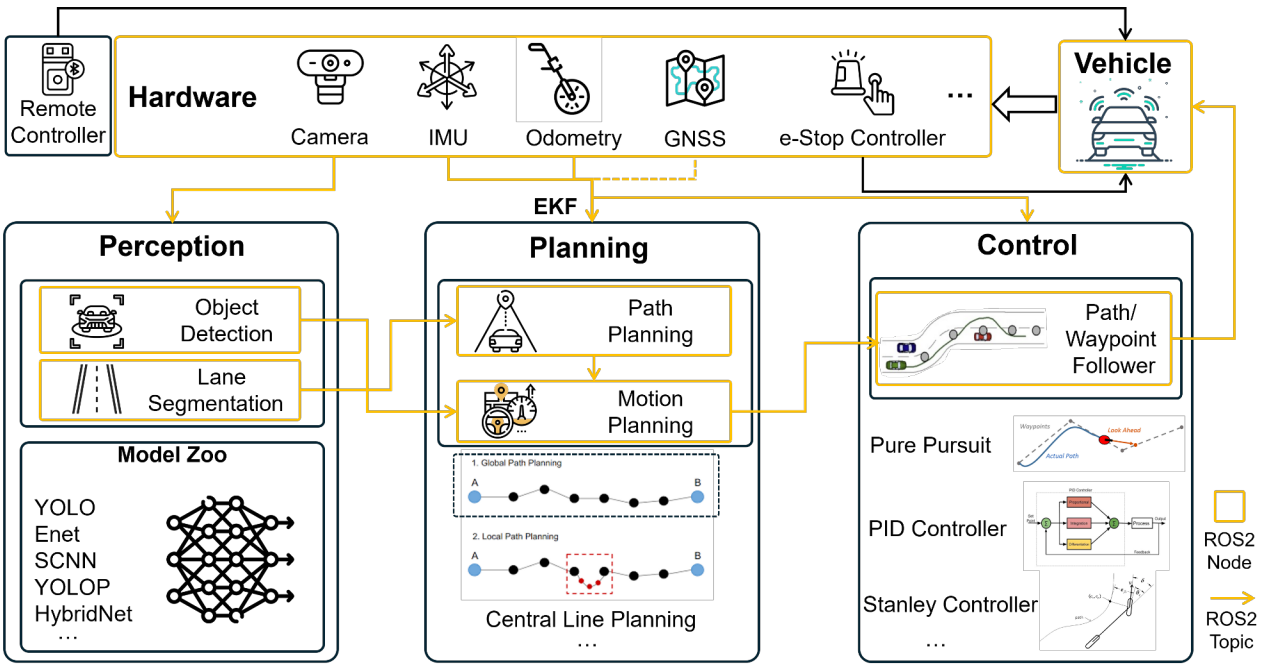


Fig. 1: The system design of *LightAD*.

extraction just once. This multi-task architecture effectively avoids the computational waste associated with repeated feature extraction, ensuring comprehensive perception capabilities while aligning well with this system's requirements for real-time performance and low computational cost.

C. Path Planning and Control

In the field of path planning, to handle complex dynamic obstacles, mainstream research often employs search-based Hybrid A* [22]–[24] or optimization-based Lattice Planner algorithms [2], [25]. While these methods can generate theoretically optimal solutions in high-dimensional spaces, they incur significant computational overhead and are often complex for simple lane-following tasks on structured roads. In contrast, the lightweight centerline planning strategy avoids costly spatial searches and directly generates reference trajectories based on perception results or spline interpolation. This geometric method ensures path smoothness while significantly reducing system latency, making it more suitable for efficient lightweight systems.

For trajectory tracking control, model-based algorithms such as MPC (Model Predictive Control) and LQR are commonly used. MPC [26]–[28] handles vehicle dynamics and multi-objective optimization by solving constrained optimization problems over a finite time horizon and is widely used in industrial frameworks like Apollo. LQR [29], on the other hand, maintains lateral stability in high-speed scenarios by minimizing a cost function. However, both methods rely heavily on an accurate vehicle dynamics model, and MPC requires solving complex real-time quadratic programming problems. The substantial computational demands make them challenging to deploy directly on low-cost embedded platforms.

In contrast, lightweight algorithms based on geometric relationships, with their low computational requirements and high robustness, are the preferred choice for some low-cost systems. The most basic PID controller [30] adopts a model-free strategy, achieving effective longitudinal and lateral control solely through error feedback. In geometric tracking, the Pure Pursuit algorithm [10] determines the steering angle by calculating the geometric arc from the vehicle's rear axle center to a look-ahead point on the target path. This mechanism inherently smooths path jitter and is commonly used in various mobile robots. Another classic method, the Stanley controller [11], originated from the DARPA Grand Challenge and performs nonlinear feedback control based on lateral and heading errors at the vehicle's front axle center, demonstrating excellent convergence and stability. These geometric algorithms involve only simple trigonometric and algebraic operations, eliminating the need for complex dynamic modeling. They can efficiently run on lightweight hardware such as the Raspberry Pi, aligning well with the system's core requirements for low cost and ease of deployment.

III. LightAD System Design

A. System Overview

LightAD employs minimal hardware and a streamlined data flow. The system framework is shown in Figure 1. The camera provides the sole environmental input, which the perception module processes to generate object summaries and lane or drivable-area geometry information. IMU (Inertial Measure-

ment Unit) and wheel speed or odometry data, supplemented with GNSS (Global Navigation Satellite System) when necessary, are fused on board to estimate the vehicle pose. The planning module operates in a local coordinate system, using the pose and lane geometry to compute a path and output a timestamped trajectory based on a constant cruise speed and a forward safe distance threshold. The control module follows the trajectory using a simple controller and implements basic speed control for deceleration and stopping. Commands are sent via a motor controller to actuate the steering and acceleration. The system includes an independent e-Stop mechanism for hardware-based power-cut braking. Bluetooth and a remote controller are provided for manual intervention and testing purposes.

B. Hardware and Vehicle-mounted Electronics

The system is designed for low-speed, well-defined scenarios, with hardware configured in a minimal viable setup entirely installed on the vehicle. The forward-facing camera provides the only external environmental input for extracting object and lane geometry information. The IMU and wheel speed sensor/odometer offer high-frequency observations to pose estimation, enabling stable planning and control in a local coordinate system. GNSS is optional and is integrated only as a low-frequency correction when routes are long or alignment with fixed stations or maps is required. An independent e-Stop controller directly cuts off the main power supply in the event of anomalies or upon manual activation, meeting basic functional and safety requirements while avoiding the cost and debugging complexity associated with multiple sensors.

The vehicle side also integrates the electronic and actuation components necessary for propulsion and power supply. A battery powers the entire vehicle, typically distributing power along two paths: the propulsion side supplies the motor and steering actuators, while the service side powers the computer and sensors via a DC/DC converter. This design reduces electromagnetic interference and facilitates fuse protection. The motor controller converts the high-level speed and steering commands into voltage and current control signals for the motor, thereby implementing acceleration and steering. An Arduino serves as a lightweight control unit. On one side, it receives `/vehicle/cmd` from the ROS system, locally generates PWM, direction, and enable signals, and transmits them to the motor controller. On the other hand, it collects chassis feedback, such as wheel speed and steering angle, via wheel odometry, and sends it back via `/vehicle/state` and `/wheel_odom`, serving as the real-time interface between the computer and the actuators. A Bluetooth module enables remote manual control; a handheld remote sends commands to the Arduino via Bluetooth. In remote-control mode, the system prioritizes manual commands and suppresses autonomous outputs, facilitating tasks such as onboard debugging, navigation through narrow sections, and emergency takeovers. The remote control also provides a fail-safe mechanism for the vehicle in emergency situations.

Overall, the hardware configuration centers on a camera, IMU, odometry, GNSS (optional), e-Stop, and a basic actuation and power supply chain. Without introducing unnecessary sensors or complex hardware, this setup meets the requirements for perception, state estimation, control actuation, and safe shutdown in simple autonomous driving scenarios.

C. Perception

The perception layer of this system is designed for low-speed, controlled, and simple autonomous driving scenarios. The only sensor input is a camera (ROS2 topic `/camera/image_raw`, typically 30 Hz). The camera-to-vehicle extrinsic parameters are fixed and published in `/tf_static`, and time synchronization is achieved via NTP/PTP or GNSS-PPS alignment with the system clock.

Upon receiving an image, the inference node selects detection and segmentation networks flexibly from the model zoo based on task requirements and available computational resources (for example, YOLO series with lightweight lane/drivable area segmentation models, or multi-task networks like YOLOP/HybridNets). Regardless of the specific model used, the node employs a model-agnostic adaptation layer to standardize outputs into two types of planning-oriented summaries: one is a list of objects (class, 2D position/scale, confidence), and the other is road geometric elements (lane boundaries, drivable area, or derived local centerlines). To reduce end-to-end latency and bus bandwidth usage, raw images and intermediate features are consumed within the process and not broadcast at high frequency; only low-frequency thumbnails are published to `/debug/image` during debugging.

In the ROS2 dataflow, the perception node publishes only the minimum necessary information downstream: object summaries are published via `/perception/objects`, and geometric elements are published via `/perception/lanes`. The planning layer only needs to subscribe to these two topics to perform centerline generation and speed planning, without directly using raw images.

To maintain a sufficient and stable implementation in simple scenarios, the perception layer avoids heavy cross-frame data association and complex multi-sensor fusion. It applies only Non-Maximum Suppression (NMS) to detection results to improve temporal consistency, and performs lightweight morphological denoising and polynomial fitting on segmentation results to obtain continuous geometric representations. The inference side can optionally employ FP16/INT8 quantization (e.g., TensorRT/ONNX Runtime) and leverage ROS2 composable deployment with intra-process communication to reduce copying and serialization overhead. This enables a low-latency, reproducible, and model-replaceable perception front-end even on entry-level computing hardware.

D. Planning

The planning module is designed for simple autonomous driving scenarios and operates in the local coordinate system (`/odom`). It relies on only two types of inputs: the geometry of

lanes or drivable areas, the summary of detected objects from the perception node, and the vehicle's pose and speed provided by the localization module. Specifically, the module subscribes to `/perception/lanes` and `/perception/objects`, and also to `/localization/pose`, where the pose is estimated through an Extended Kalman Filter (EKF) within the process, fusing data from an IMU (`/imu`) and wheel odometry (`/wheel_odom`). GNSS is optional (`/fix` or `/navsat/odom`) and used only for low-frequency corrections in scenarios with long routes or when alignment with stations or maps is required; it is not required for short or simple routes.

For path generation, the module projects the left and right boundaries (or boundaries derived from the drivable area) into the vehicle coordinate system and directly calculates their equidistant centerline. A rolling centerline segment anchored to the current vehicle position is maintained for subsequent temporal parameterization. This segment can optionally be published to `/planning/path` for visualization.

Subsequently, in trajectory generation, the module timestamps the centerline and publishes the result to `/planning/motion`. The speed policy follows two simple rules: a maximum speed v_{set} is set, and the forward distance to the nearest detected object along the path is continuously monitored. When the distance exceeds the safety threshold D_{safe}, the vehicle proceeds at v_{set}. If the distance falls within the threshold, a fixed deceleration slope is applied to generate a stopping trajectory, ensuring the vehicle comes to a complete stop before reaching the safety distance.

The trajectory timestamps are aligned with the pose timestamps. The controller subscribes only to `/planning/motion` and does not directly consume raw image data or large segmentation outputs. Through this minimal implementation—comprising single-camera perception summaries, IMU/odometry-based pose estimation, centerline-following, and two speed rules—the planning module can stably output low-latency, reproducible timestamped trajectories even on low-power computing platforms, while retaining the flexibility to integrate GNSS corrections when needed.

E. Control

The control module is responsible for tracking the timestamped trajectory provided by the planning module and converting it into steering, driving, and braking commands for the vehicle. The node subscribes to `/planning/motion` and `/localization/pose` and publishes them to `/vehicle/cmd` via an Ackermann interface. Commands are forwarded through a safety gateway, which monitors emergency stop signals; in the event of a timeout or an emergency stop, the gateway triggers braking and power cutoff.

For lateral control, the lightweight Pure Pursuit is recommended: the look-ahead distance is adjusted adaptively based on vehicle speed, and the curvature to the target look-ahead point on the trajectory is calculated and mapped to a steering angle. To suppress oscillations, steering angle and steering rate are subject to amplitude limiting and first-order filtering. Additionally, command output is synchronized with the trajectory timestamp to avoid overshoot caused by phase lag. Alternatively, the Stanley Controller algorithm can be used. This algorithm constructs a nonlinear feedback control law based on the lateral and heading errors of the front wheel center, thereby achieving fast convergence and high-precision tracking of the reference path without pre-aiming.

In simple scenarios, the recommended cost-effective approach is to use Pure Pursuit or a Stanley Controller for lateral control and rely on the chassis's built-in speed closed-loop control or a fixed cruise mode for longitudinal control. The planning module provides a target cruise speed v_{set} and a stopping point based on forward distance. The control module then sends v_{set} as the chassis target or maintains constant throttle or PWM, enabling reliable operation on flat roads, at low speeds, and under stable load conditions.

If greater stability is required, for example, to handle voltage fluctuations, slight slopes, or the need to decelerate to a stop within a safe distance, a simple speed PID controller can be added for longitudinal control. This PID takes the reference speed from the trajectory as input, incorporates anti-windup and fixed acceleration and deceleration slope limits. It automatically decelerates when entering the safe distance zone, brings the vehicle to a stop before the threshold, and smoothly accelerates once conditions permit. This simplified strategy is sufficient because our system operates at low speeds in a well-defined environment. The trajectory is guaranteed to be smooth and rule-based by the planning stage, and vehicle dynamics approximate linear behavior within this low-speed regime. As a result, even without a complex controller, the required lateral accuracy and longitudinal comfort can be achieved.

IV. Algorithm Recommendations

In the *LightAD* system, lightweight algorithms and models are prioritized, provided they do not degrade performance in simple scenarios. The architecture of *LightAD* is flexible, allowing perception models and control algorithms to be readily swapped or replaced. This section evaluates the performance and efficiency of commonly used perception models and control algorithms in autonomous driving systems, and provides corresponding recommendations for suitable algorithms and models.

A. Perception Models

The *LightAD* framework does not mandate the use of specific perception models. Instead, it provides developers with various recommendations based on trade-offs between computational efficiency and perception accuracy. Table I presents a detailed comparison of mainstream models in terms of their performance and efficiency for object detection and lane line segmentation. For object detection, models such as YOLO5x and DINO have strong detection capabilities, but their high parameter counts or computational demands

TABLE I: Efficiency and accuracy comparison for single-task and multi-task models. * indicates training on the COCO dataset. † indicates training on the BDD100K dataset. Rec denotes the recommended status.

Model	Task	Efficiency		Accuracy		Rec
		Params	GFLOPs	mAP	IoU	
YOLO11x* [13]	OD	56.9M	**194.9**	**54.7**	–	✓
Mask R-CNN* [17]	OD	46.4M	199.4	37.1	–	✗
Faster R-CNN* [16]	OD	**40.0M**	207.0	21.9	–	✗
YOLO5x* [12]	OD	97.2M	246.4	53.2	–	✗
DINO* [18]	OD	47.0M	860	49.0	–	✗
ENet† [20]	LS	0.4M	4.1	–	14.6	✗
SCNN† [19]	LS	29.5M	–	–	15.8	✗
YOLO5s†	OD	9.1M	14.4	77.2	–	✓
HybridNets† [15]	OD+LS	12.8M	**7.8**	77.3	31.6	✓
Sparse U-PDP† [31]	OD+LS	12.0M	15.1	79.7	**32.4**	✓
YOLOP† [14]	OD+LS	**7.9M**	9.3	**76.5**	26.5	✓

of up to 860 G FLOPS make them unsuitable for the low-power, low-computation scenarios targeted by this framework. Notably, both the YOLOv5 and YOLOv11 series offer more lightweight variants. For example, compared to YOLO5x, its lighter model, YOLO5s, significantly reduces parameter count and FLOPs while maintaining strong object detection performance on the BDD100K dataset, making it more suitable for hardware platforms with limited resources. Additionally, YOLO11x is listed as a recommended option for higher-computation configurations due to its excellent detection accuracy (mAP = 54.7) and relatively reduced FLOPs. However, among single-task models for lane segmentation, none of the current mainstream solutions are recommended. The SCNN model suffers from parameter redundancy (29.5M), while the extremely lightweight ENet, despite its low computational cost, exhibits a severely insufficient segmentation IoU of only 14.6.

Based on the above analysis, to accommodate multiple perception tasks under limited computational resources, the *LightAD* framework strongly advocates for multi-task perception models. As shown in Table I, HybridNets and YOLOP both demonstrate excellent efficiency and are listed as recommended solutions. Notably, the YOLOP model achieves higher segmentation accuracy than the single-task ENet with only 7.9 M parameters and 9.3 G FLOPS. The system-level comparison in Figure 2 further confirms the effectiveness of this strategy: compared to combining single-task models (e.g., YOLOv5s, ENet-SAD [32], and GCNet [33]), adopting a unified multi-task architecture (YOLOP) not only reduces the total model size by approximately 4.5 times but also significantly lowers inference latency from 65.18 ms to 41.67 ms. These results indicate that while high-performance single-task models such as YOLO11x excel in specific metrics, multi-task models offer a more balanced and efficient engineering solution for cost-sensitive deployment in the well-defined scenarios targeted by *LightAD*.

B. Planning and Control Algorithm

Table 3b provides a detailed performance comparison of five different controllers under the simple path conditions shown

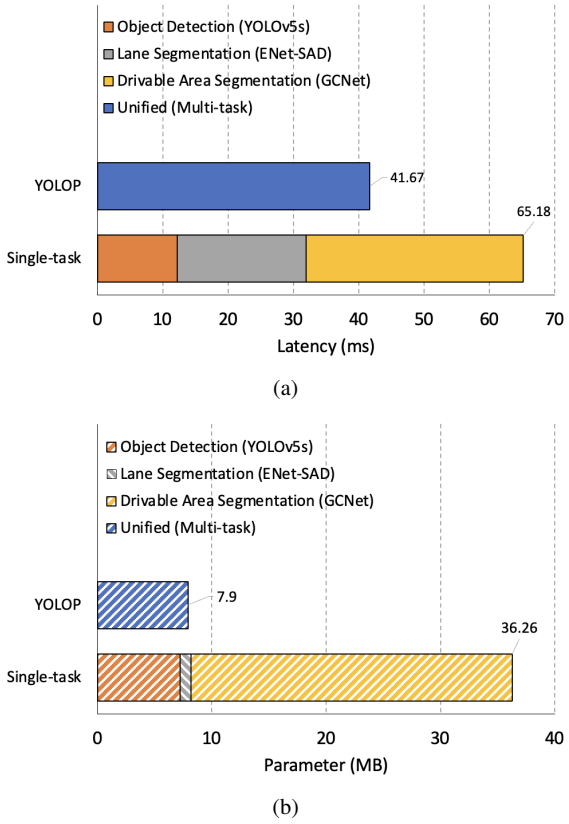
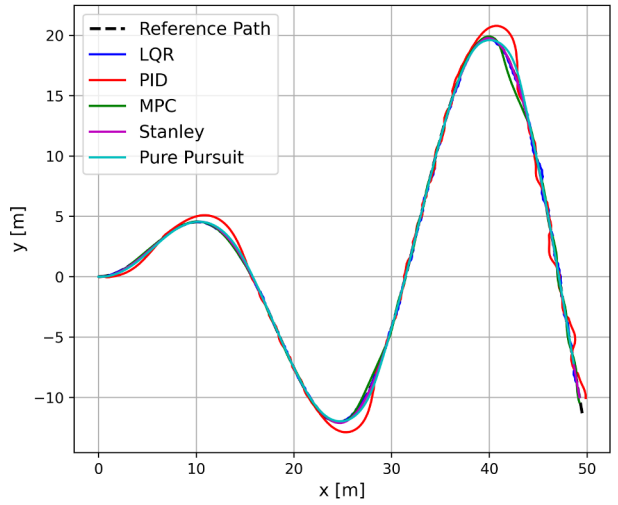


Fig. 2: **Comparison between single-task combination and multi-task models.** (a) Latency comparison. (b) Parameter size comparison.

in Figure 3a, covering maximum error, mean error, root mean square error (RMSE), and single-frame computation time. In terms of control accuracy, the optimization-based MPC controller demonstrated the expected advantage, with a mean error of only 0.34 m and an RMSE of 0.43 m. However, this high precision comes at the cost of significant computational overhead, with an average single-frame computation time of 21.09 ms. In contrast, the traditional PID controller, while having the fastest computation speed (0.03 ms), exhibited a mean error of 0.50 m and an RMSE of 0.57 m, making it the worst performer in terms of control quality.

Notably, the geometry-based Stanley and Pure Pursuit algorithms showed excellent overall performance in this experiment. Taking the Stanley algorithm as an example, its mean error (0.34 m) and RMSE (0.43 m) matched those of MPC exactly, while its computation time was only 0.07 ms. Similarly, Pure Pursuit achieved nearly identical mean error and RMSE, with a computation time of just 0.09 ms. This indicates that, in simple, structured road scenarios, lightweight geometric controllers can achieve the same level of trajectory-tracking accuracy as MPC, using approximately 1/300 the computation time. This strongly supports the conclusion that, on cost-constrained hardware platforms with limited computational



(a) Simulation of different control algorithms along a simple path.

Controller	Max Error (m)	Mean Error (m)	RMSE (m)	Time (ms)	Rec
MPC	1.15	0.34	0.43	21.09	✗
LQR	1.20	0.35	0.44	0.31	✗
PID	**1.29**	**0.50**	**0.57**	**0.03**	✓
Stanley	**1.21**	**0.34**	**0.43**	**0.07**	✓
Pure Pursuit	**1.18**	**0.35**	**0.43**	**0.09**	✓

(b) Performance and efficiency of different control algorithms under simple path conditions.

Fig. 3: Control algorithms: simulation (a) and quantitative comparison (b).

resources, geometric control algorithms are more efficient than complex optimization-based approaches.

V. Case Study at IGVC

This case study presents a complete engineering implementation that transforms a typical children's electric ride-on vehicle into an autonomous driving platform compliant with IGVC standards and deploy. IGVC is an outdoor event that evaluates low-speed intelligent ground vehicles on lane keeping, obstacle avoidance, and stop-sign handling under defined safety rules. The project involved not only in-depth electromechanical modifications of an electric vehicle but also the deployment of *LightAD* adapted for simple, structured, and rule-defined environments.

A. Hardware Implementation

1) ***Mechanical Platform Modifications****:* The experimental platform is a jeep-style children's electric vehicle (dimensions: 106 cm × 75 cm × 78 cm). Its compact size and plastic frame provide a solid foundation for modifications. To enable core autonomous control, the steering system has been fully converted to a steer-by-wire system. The original mechanical steering shaft was physically disconnected. Threads were re-cut using a 10mm die, and thrust bearings with lock nuts were introduced to achieve mechanical decoupling. The original open-loop motor was replaced with a high-speed DC motor with a built-in encoder. This motor is mounted to the chassis via a custom-designed 3D-printed bracket. Paired with a PID closed-loop control algorithm, it enables precise adjustment of the steering angle.

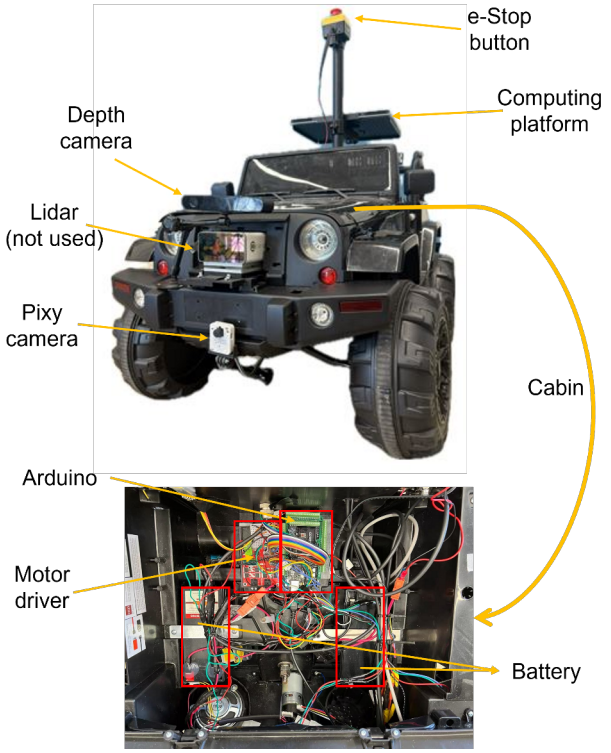


Fig. 4: Hardware system design of the experimental platform.

2) ***Sensor Integration****:* Sensors are arranged in a modular layout to meet environmental perception requirements. While a Cepton Vista-P60 LiDAR is mounted on the roof with a specially designed bracket featuring sliding rails for physical calibration in horizontal and vertical directions, the current implementation primarily relies on vision sensors. The core perception device is the ZED 2i stereo camera, mounted high on the front frame. This high-performance depth camera provides high-resolution RGB images based on binocular vision principles and generates high-precision depth maps using its built-in neural depth-sensing technology. Crucially, the ZED 2i integrates an IMU containing a barometer, magnetometer, and high-frequency gyroscope and accelerometer, providing essential pose data. A Pixy camera is mounted lower on the vehicle to recognize nearby ground lane markings and stop lines.

3) ***Electronic Architecture and Power Management****:* A key challenge for the electronic system was managing electromagnetic interference from high-motor loads while ensuring signal integrity. To address this, a dual-battery isolated power supply configuration was implemented. Two independent 12V

	LightAD		Standard Research-Grade Autonomous System	
Component	Description	Est. Cost	Description	Est. Cost
Mobile Platform	Modified Electric Ride-on Car	$250 - $400	Research-grade Chassis (e.g., AgileX Scout)	$4,500 - $8,000
Computing Platform	Low-cost edge device (e.g., Jetson Nano)	$400 - $1500	Powerful GPU (e.g., Jetson AGX Orin)	$1,500 - $2,000
Camera	Depth Camera (e.g., ZED 2i)	$300 - $600	Multi-camera setup	$1,000 - $1,500
LiDAR	Not used	$0	3D LiDAR (e.g., Velodyne VLP-16)	$1,000 - $2,000
Microcontroller	Arduino Mega 2560 / Uno Rev3	$30 - $50	High-performance Embedded (e.g., STM32F7)	Included
Motor Driver	High-power H-Bridge Driver (e.g., L298N)	$20 - $30	High-power H-Bridge Driver	Included
Power System	12V Lead-Acid or Li-Po Battery	$30 - $50	High-capacity Li-Ion Battery	$300 - $500
Safety & Misc.	E-Stop Button, pixy camera...	$100 - $150	Industrial E-Stop Tower, RTK-GNSS System...	$600 - $750
Total Estimated Cost		**$1,130 - $2,780**		**$8,900 - $20,600**

TABLE II: The comparison of the hardware costs of *LightAD* and the Standard Research-Grade Autonomous System.

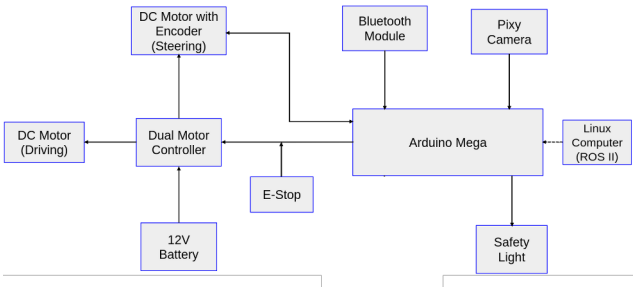


Fig. 5: Electrical subsystem block diagram.

9AH lead-acid batteries are used: one powers the high-power traction motor, steering motor, and the Arduino Mega 2560 low-level controller; the other powers the onboard PC and sensitive sensors. This physical isolation effectively prevents sensitive computing devices from resetting due to current surges when the motors start.

The Arduino Mega 2560 was selected as the low-level controller primarily due to its rich digital I/O interfaces (54 pins), which are well-suited for connecting to the L298N dual H-bridge driver and reading multiple sensors. For the communication interface design, a voltage divider was added to the circuit to protect the HC-05 Bluetooth module (used for remote debugging and emergency stops) from the Arduino's 5V logic level, since the module operates at 3.3V. Furthermore, the vehicle integrates a physical emergency stop button and a wireless remote emergency stop mechanism. When triggered, a relay directly grounds the motor drive signal, cutting off power output at the hardware circuit level to provide a fail-safe mechanism and ensure testing safety.

4) ***Computing Platform****:* The computing platform used in this project is an Alienware x16 R2 laptop. Its robust specifications enable it to handle complex deep learning tasks: it is equipped with an Intel Core Ultra 9 185H processor, 32GB of RAM, and an NVIDIA GeForce RTX 4070 GPU with 8GB of VRAM. This configuration ensures the system can run perception network inference and depth computation tasks from the ZED SDK, as well as a lightweight control algorithm.

B. Monetary Cost

To evaluate the monetary cost of the proposed framework, we conducted a comparative cost analysis between the *LightAD* hardware platform and a standard research-grade autonomous driving system typically deployed in campus environments. Table II details the Bill of Materials (BOM) for both configurations.

As shown in Table II, a standard baseline relies heavily on industrial-grade components to ensure robustness. The bulk of the price is driven by the high-precision mobile chassis (e.g., AgileX or Clearpath series, $5,000+) and the multi-modal sensor suite. Crucially, such systems depend on 3D LiDARs ($3,000+) for depth perception and mapping, often supplemented by a multi-camera suite and high-precision RTK-GNSS for localization. Consequently, the total hardware cost for a standard baseline typically exceeds $8,000, creating a significant barrier to entry for large-scale deployment or educational use.

In contrast, the proposed *LightAD* adopts a minimalist design philosophy centered on consumer-grade components. By replacing the industrial chassis with a modified electric ride-on vehicle, mobility costs are reduced by over 90%. More importantly, the proposed framework eliminates the financial burden of the perception stack by removing the expensive LiDAR and RTK-GNSS modules entirely. Instead, it relies on a single depth camera and a Pixy camera to achieve comparable functionality in structured environments. This sensor ablation strategy results in a drastic reduction in total system cost. The *LightAD* platform can be constructed for approximately $3,000, which represents a significant cost reduction compared to the standard research-grade autonomous driving system.

C. Software Implementation

1) ***Computing Environment****:* The software architecture is deployed on Ubuntu 22.04, with ROS2 Humble as the middleware. To fully leverage the hardware capabilities of the ZED 2i, the system deeply integrates the ZED SDK with the ROS2 wrapper. This architecture allows developers to call packaged low-level libraries directly to perform complex tasks, such as obtaining calibrated point cloud data, confidence maps, and high-frequency IMU pose data, without writing drivers from scratch. All connection interfaces have been upgraded to USB 3.0, ensuring low-latency transmission of the massive visual data and depth information.

2) ***Perception Module****:* To balance development efficiency, inference real-time performance, and hardware compatibility, the software system adopts a decoupled hybrid perception architecture. In this architecture, the tasks of obstacle detection

and lane segmentation are assigned to two deep learning models, each optimized for its specific purpose.

Object Detection via YOLO11: For object detection, the system selects the YOLO11s model. This decision is primarily based on the convenience of hardware integration and training efficiency. Since the SDK of the onboard ZED 2i camera provides native support for the YOLO series algorithms, adopting YOLO11 enables automatic alignment between 2D bounding boxes and 3D depth information at the underlying level. This eliminates the need for writing complex custom code to map depth information, as would be required with models like YOLOP. Additionally, a custom dataset containing 4,870 images was constructed specifically for the competition scenario, with the detailed data presented in Table III. Compared to the more complex multitask head structure of YOLOP, YOLO11's simpler architecture facilitates easier, more efficient fine-tuning on custom datasets.

TABLE III: Distribution of training, validation, and test samples in our generated traffic object detection dataset across various traffic object categories.

Type	Category	Train	Validation	Test
Stop sign	Stop-sign	456	131	65
	Stop-sign-fake			
	Stop-sign-vandalized			
Traffic barrel	Traffic-barrel	778	223	101
	Traffic-cone	558	165	86
Car tire	Car-tire	488	191	104
Pedestrian	Pedestrian	607	102	6

During model selection, while lightweight versions such as YOLO5s achieved mAP comparable to YOLOP on the BDD100K dataset, the latest YOLO11s was considered a candidate due to its highly efficient design. However, these lightweight models performed poorly in terms of accuracy on the custom dataset, as shown in Table IV. Therefore, larger models such as YOLO5x and YOLO11x were trained and evaluated. Although these models have increased parameters, their parameter size and Flops remain within the real-time processing capability of the computing platform. Ultimately, YOLO11x was selected for the object detection task in the system, as it achieves superior detection performance while maintaining a lighter structure compared to YOLO5x.

Lane Segmentation via YOLOP: While YOLO11 replaces the detection head of YOLOP, the system retains YOLOP's strengths in panoramic driving perception by specifically utilizing it for lane line segmentation. Compared to traditional computer vision methods, such as edge detection, and other networks like ENet and SCNN, YOLOP demonstrates greater robustness under varying lighting conditions and complex textures. In practice, only the segmentation branch of YOLOP is activated for inference, thereby avoiding the computational waste associated with redundant detection tasks.

TABLE IV: The performance of YOLO models on the custom dataset.

Model	mAP@0.5	mAP@0.5:0.95	Param (M)	FLOPs (G)
YOLO5s	0.517	0.244	9.1	24.0
YOLO11s	0.572	0.293	9.4	21.5
YOLO5x	0.630	0.313	97.2	246.4
YOLO11x	**0.887**	**0.692**	**56.9**	**194.9**

This dual-model parallel strategy fully leverages the advantages of different algorithms: YOLO11 provides high-precision obstacle coordinates with depth information, while YOLOP supplies global road geometry constraints. Both sets of data are ultimately transmitted to the ROS2 decision-making node.

*3) **Planning and Control**:* Given the hardware configuration and practical trade-offs, the navigation and obstacle avoidance logic of this system is entirely based on visual data; LiDAR point clouds are not used for closed-loop control. Lane keeping is achieved by subscribing to the lane line masks output by YOLOP. The algorithm dynamically calculates the midpoint of the left and right boundaries to generate a virtual center path. The vehicle's heading deviation is then calculated and fed into a PID controller to adjust steering.

For obstacle avoidance, the system uses real-time depth maps from the ZED 2i stereo camera as a substitute for LiDAR ranging. When the detection model detects an obstacle on the forward path at a distance below the safety threshold, the decision module issues a high-priority command to override the lane-keeping output and execute deceleration or avoidance maneuvers. Additionally, the Pixy camera serves as a dedicated ground feature sensor. The stop line signals recognized in real-time are fed into the system's state machine logic. Once the trigger conditions are met, the vehicle will ignore other control commands and execute a mandatory stop to comply with traffic rules.

VI. Discussion

Through the IGVC competition, the feasibility of *LightAD* was verified, while it still exhibits many shortcomings, from which we have also learned valuable lessons.

*A. **Lesson 1: Pre-trained models perform inconsistently in real-world scenarios, making scenario-specific fine-tuning a necessary condition.***

During the deployment of perception algorithms, our primary challenge was the failure of pre-trained models in specific scenarios. In essence, this reflects a significant visual discrepancy between what the model has "learned" and what it actually "sees." In simpler terms, although the model was trained on general datasets like COCO, the features it learned showed a substantial distributional difference from the actual objects in the competition environment. To illustrate this issue concretely, we used pedestrian detection as a case study: in the COCO dataset, the model learned to recognize humans with rich facial expressions, natural body movements, and varied clothing textures. However, in the IGVC competition, the detection targets were static, smooth-surfaced, and rigidly posed plastic mannequins. This fundamental difference in visual features caused models like YOLO and YOLOP, which excel in general scenarios, to struggle to extract discriminative features for these "fake humans," frequently resulting in missed detections.

To address this issue, this study adopted a fully supervised fine-tuning strategy. We augmented the initial competition dataset by collecting additional field data of target objects at the IGVC venue. This expanded dataset enabled the model to learn the specific feature distributions required for a real-world competition environment. As illustrated in Figure 6, the YOLO11x model fine-tuned on the expanded IGVC-specific dataset achieves improved detection accuracy across all classes compared to the model fine-tuned on our custom dataset presented in Table III. However, reflecting on this process, relying solely on large-scale data collection and fine-tuning is not the only or even the most efficient solution for such problems.

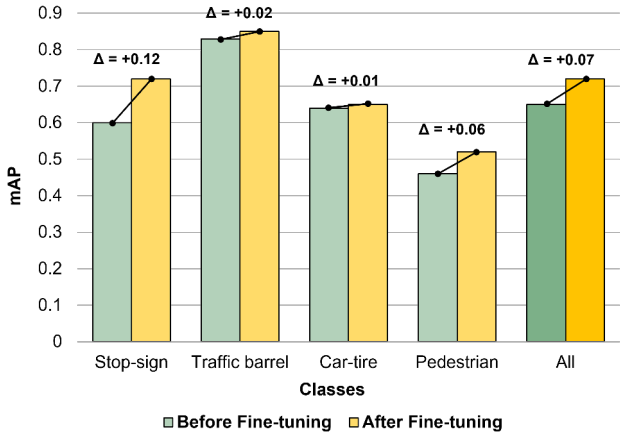


Fig. 6: Results of YOLO11x before and after finetuning on the expanded dataset.

In future research or in scenarios with more limited resources, we can explore more forward-looking alternatives. For example, leveraging Few-Shot Learning techniques [34] by providing the model with only a few core mannequin samples can quickly establish the concept of a new category through metric learning, significantly reducing data collection costs. Additionally, Synthetic Data Generation is a highly promising direction. Using simulation engines like Gazebo or Unity to batch-generate diverse virtual images containing mannequins, combined with Domain Randomization techniques [35] (such as randomly altering lighting and textures), can train robust models without any real-world data collection. Moreover, with the advancement of multimodal large models, Zero-Shot Learning using models like CLIP [36] to guide the detector via textual descriptions (such as "a plastic mannequin") to recognize targets directly also offers new possibilities for addressing such scenario-specific adaptation issues. Therefore, while fine-tuning remains the most robust engineering approach at present, in the long term, integrating synthetic data with few-shot or zero-shot learning paradigms will represent a more optimal solution for enhancing the environmental adaptability of autonomous driving systems.

B. Lesson 2: Impact of dynamic environmental illumination on imaging quality and perception stability.

In unstructured outdoor environments, the robustness of visual perception systems is challenged by rapid, substantial variations in natural lighting conditions. The dynamic range of scene illumination in the competition area often exceeds the linear response range of conventional cameras, primarily due to rapidly changing cloud cover and strong backlighting during specific time periods. Such abrupt changes in lighting conditions can directly cause instability in the camera's image signal processing (ISP) pipeline, leading to substantial oscillations and lag in automatic exposure and white balance adjustments. For example, under strong backlighting conditions, automatic exposure tends to darken the overall image to protect highlight details, resulting in lane-line textures in shadowed areas being completely lost. When clouds move quickly, white balance drift can cause confusion between yellow and white lane line hues.

(a) (b)

Fig. 7: **Indoor lane segmentation under different lighting conditions.** (a) The indoor lane markings are correctly segmented and output. (b) Output under drastic changes in lighting conditions.

We simulated image overexposure due to intense indoor lighting, as shown in Figure 7. Although it differs from natural outdoor light, we can observe that the intense lighting affected the input image, resulting in an incorrect output. These physical disturbances at the data input stage directly disrupt the color and texture features required by subsequent segmentation networks, leading to unpredictable fluctuations in segmentation results. This phenomenon suggests that relying solely on software-level data augmentation cannot fully compensate for physical imaging defects. Future system designs must prioritize front-end imaging quality. It is recommended to introduce hardware devices that support high dynamic range (HDR) or to develop ISP adjustment algorithms optimized for autonomous driving scenarios, ensuring accurate exposure and stable color reproduction even under extreme weather or lighting conditions.

C. Lesson 3: Noise sensitivity in semantic segmentation and necessity of geometric constraints.

Lane segmentation is highly susceptible to interference from environmental noise in complex texture backgrounds. Experiments show that non-target objects with similar color or texture features, such as curb edges, dry grass, ground cracks, or parking lines, are frequently misclassified by deep

(a) (b)

Fig. 8: **Outdoor lane segmentation results.** (a) The correctly segmented outdoor lane line. (b) The segmentation results are after being corrupted by noise from other lane lines.

learning models as lane boundaries, resulting in substantial false-positive noise in the segmentation mask. More critically, post-processing algorithms that rely solely on mathematical fitting, such as polynomial fitting, lack prior knowledge of the road's physical structure. When the input data contains outliers, these algorithms are prone to overfitting, generating sharply distorted paths that violate vehicle kinematic constraints. For example, Figure 8 shows an error we encountered during testing at the IGVC competition venue. The segmentation model incorrectly identified the parking lines as lane lines, resulting in severely distorted lane fits. This lesson highlights a key limitation: outputs of deep learning models often lack geometric consistency. To build a safe system, it is essential to introduce rule-based geometric verification or temporal consistency filtering during post-processing. By combining data-driven pixel-level predictions with traditional geometric model constraints, transient noise and outliers can be effectively eliminated, ensuring that the final generated navigation path is spatially smooth and temporally continuous.

*D. **Lesson 4: Cumulative error in consumer-grade IMUs and multi-sensor fusion strategies.***

In areas where GPS signals are unavailable or obstructed, relying solely on IMUs (Inertial Measurement Units) for localization presents significant physical limitations. The working principle of an IMU involves measuring acceleration and angular velocity to infer positional changes, which inherently involves relative positioning. Due to the inevitable presence of minor measurement noise in consumer-grade sensors, when these measurements are mathematically integrated to derive position, these minor errors accumulate continuously over time. This leads to a phenomenon known as drift. It is noteworthy that this drift does not manifest immediately; within short time windows, the accuracy of IMU-based estimates is usually sufficient to keep the vehicle within its designated lane, without causing traffic violations or collisions. However, as operating time increases, integration errors accumulate nonlinearly. The discrepancy between the vehicle's computed position and its physical ground truth gradually widens, eventually causing it to deviate from its intended route without external perception inputs to correct it.

To effectively suppress this long-term drift and build a robust localization system, relying on a single sensor is clearly insufficient. A multi-sensor heterogeneous sensor fusion strategy is essential. First, introducing absolute positioning constraints is the most direct approach, namely, integrating a GNSS/RTK module. Unlike relative estimates from IMUs, GNSS provides absolute positions in an Earth-fixed coordinate system. Its errors do not accumulate over time, allowing periodic resetting of the IMU's cumulative error using low-frequency, high-precision data. Second, making full use of chassis hardware resources for wheel odometry fusion is another critical step. While wheel speed data from motor encoders may be affected by wheel slip, their accuracy in low-speed and straight-line driving scenarios is significantly higher than that of IMUs, and they are unaffected by lighting conditions. By using an Extended Kalman Filter to incorporate wheel speed data as a velocity observation source, the bias drift of the IMU during stationary or low-speed vehicle states can be effectively constrained. This significantly extends the system's adequate operational time in GPS-denied environments.

VII. Conclusion

We introduce *LightAD*, a lightweight vision-based autonomous driving system to democratize autonomous mobility research by addressing the prohibitive costs of standard industrial systems. By adopting a comprehensive cost-reduction strategy, we demonstrated that a wide range of expensive specialized hardware, from the chassis to the sensor suite, can be effectively substituted by accessible consumer-grade alternatives for early testing in well-defined environments. Our experiments and the real-world deployment at the Intelligent Ground Vehicle Competition verified that, when the uncertainty of operating conditions is low, such a low-cost system coupled with lightweight algorithms is feasible and effective for fundamental self-driving tasks. Furthermore, the practical challenges encountered during the competition provided valuable lessons on system robustness and error handling, establishing a foundation for future improvements in *LightAD* system. In future work, we will prioritize data-efficient scenario adaptation, including few-shot and synthetic data, robust illumination imaging such as HDR and ISP, geometric–temporal constraints for lane estimation, and stronger localization via multi-sensor fusion, incorporating IMU, wheel odometry, and GNSS.

Acknowledgment

This work is supported in part by Michigan Economic Development Corporation through Advance Grant Proof-of Concept Fund, in part by National Science Foundation through Award #2432098, and in part by the School of Engineering and Computer Science at Oakland University.

References

[1] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 287–296.

[2] Baidu Apollo. (2017) Baidu apollo github repository. [Online]. Available: https://github.com/ApolloAuto/apollo

[3] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang *et al.*, "Duckietown: an open, inexpensive and flexible platform for autonomy education and research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1497–1504.

[4] X. Wu and A. Eskandarian, "An improved small-scale connected autonomous vehicle platform," in *Dynamic Systems and Control Conference*, vol. 59148. American Society of Mechanical Engineers, 2019, p. V001T04A003.

[5] S. Karaman, A. Anders, M. Boulet, J. Connor, K. Gregson, W. Guerra, O. Guldner, M. Mohamoud, B. Plancher, R. Shin *et al.*, "Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at mit," in *2017 IEEE integrated STEM education conference (ISEC)*. IEEE, 2017, pp. 195–203.

[6] Jetracer. (2019) Jetracer github repository. [Online]. Available: https://github.com/NVIDIA-AI-IOT/jetracer

[7] Donkeycar. (2018) Donkeycar github repository. [Online]. Available: https://github.com/autorope/donkeycar

[8] C. Rother, Z. Zhou, and J. Chen, "Development of a four-wheel steering scale vehicle for research and education on autonomous vehicle motion control," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 5015–5022, 2023.

[9] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," *Proceedings of Machine Learning Research*, vol. 123, 2020.

[10] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Tech. Rep., 1992.

[11] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *2007 American control conference*. IEEE, 2007, pp. 2296–2301.

[12] R. Khanam and M. Hussain, "What is yolov5: A deep look into the internal features of the popular object detector," *arXiv preprint arXiv:2407.20892*, 2024.

[13] ——, "Yolov11: An overview of the key architectural enhancements," *arXiv preprint arXiv:2410.17725*, 2024.

[14] D. Wu, M.-W. Liao, W.-T. Zhang, X.-G. Wang, X. Bai, W.-Q. Cheng, and W.-Y. Liu, "Yolop: You only look once for panoptic driving perception," *Machine Intelligence Research*, vol. 19, no. 6, pp. 550–562, 2022.

[15] D. Vu, B. Ngo, and H. Phan, "Hybridnets: End-to-end perception network," *arXiv preprint arXiv:2203.09035*, 2022.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[18] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum, "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," *arXiv preprint arXiv:2203.03605*, 2022.

[19] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[20] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[21] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang *et al.*, "Hybrid task cascade for instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4974–4983.

[22] J. Petereit, T. Emter, C. W. Frey, T. Kopfstedt, and A. Beutel, "Application of hybrid a* to an autonomous mobile robot for path planning in unstructured outdoor environments," in *ROBOTIK 2012; 7th German conference on Robotics*. VDE, 2012, pp. 1–6.

[23] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The international journal of robotics research*, vol. 29, no. 5, pp. 485–501, 2010.

[24] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2020.

[25] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 987–993.

[26] Z. Zhou, C. Rother, and J. Chen, "Event-triggered model predictive control for autonomous vehicle path tracking: Validation using carla simulator," *IEEE transactions on intelligent vehicles*, vol. 8, no. 6, pp. 3547–3555, 2023.

[27] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on control systems technology*, vol. 15, no. 3, pp. 566–580, 2007.

[28] Z. Zhou, C. Rother, and J. Chen, "Comparison of two-wheel and four-wheel steering using event-triggered predictive motion control and scale vehicles," *ASME Letters in Dynamic Systems and Control*, vol. 4, no. 3, p. 031002, 2024.

[29] J. M. Snider *et al.*, "Automatic steering methods for autonomous automobile path tracking," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[30] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[31] H. Wang, M. Qiu, Y. Cai, L. Chen, and Y. Li, "Sparse u-pdp: A unified multi-task framework for panoptic driving perception," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 10, pp. 11 308–11 320, 2023.

[32] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection cnns by self attention distillation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1013–1021.

[33] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, "Gcnet: Non-local networks meet squeeze-excitation networks and beyond," in *Proceedings of the IEEE/CVF international conference on computer vision workshops*, 2019, pp. 0–0.

[34] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 8420–8429.

[35] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[36] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.