Traffic Target Integrated Detection and Spatial Visualization System Based on Dual YOLOv11 Parallel Architecture

Siwei Yang
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: 2678827634@qq.com

Jiasi Wang
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, China
E-mail: 3270048122@qq.com

***Abstract*—To address the challenges of insufficient collaborative detection accuracy for traffic signs and vehicles, weak spatial visualization, and the imbalance between system integration and lightweight deployment in intelligent transportation scenarios, this paper proposes a traffic-target integrated detection and spatial visualization system based on a dual YOLOv11 parallel architecture. Following the technical route of independent training, parallel inference, and result fusion for dedicated traffic-sign and general-purpose vehicle models, the system incorporates frame-rate-based adaptive sampling, three-level spatial coordinate mapping, and digital-twin canvas rendering. It realizes a full-process closed loop encompassing multi-source input, dual-model parallel detection, lightweight video processing, spatial visualization, interactive control, data statistics and export, as well as user authentication and permission management. Experimental results demonstrate that the system achieves a traffic-sign detection mAP@0.5 of 92%, a comprehensive detection rate of 90%, an overall vehicle detection mAP of 92.7%, and a single-frame inference time of approximately 0.28 s, meeting real-time requirements. With the adaptive sampling strategy of one frame per second, CPU usage decreases by about 65% while processing efficiency increases threefold. Built with PyQt5 and SQLite, the system features convenient deployment, cross-platform compatibility, and strong scalability for applications in intelligent traffic management, driver assistance, traffic digital twins, flow monitoring, and teaching experiments.**

Keywords—Yolov11; Dual-Model Parallelism; Traffic Sign Detection; Vehicle Detection; Spatial Visualization; Digital Twin; Intelligent Transportation; Pyqt5

I. INTRODUCTION

With the rapid advancement of Intelligent Transportation Systems (ITS) and autonomous driving technologies, accurate and robust perception of road targets has become a fundamental prerequisite for improving traffic efficiency, ensuring driving safety, and enabling traffic digital twins. As urban road networks in China continue to expand and the number of motor vehicles increases year by year, traffic environments have grown increasingly complex and dynamic. Traditional traffic management approaches that depend heavily on manual inspection and rule-based decision-making are no longer sufficient to cope with the scale and complexity of modern urban road systems. These methods are not only labor-intensive and inefficient but also lack the real-time responsiveness required for dynamic traffic control. Machine vision-based object detection, owing to its advantages of non-contact sensing, flexible deployment, and relatively low hardware cost, has emerged as the predominant technical approach for road target perception.

In real-world traffic scenarios, traffic signs convey critical regulatory information including speed limits, prohibitions, directional guidance, and warnings, while vehicles represent the most dynamic and numerous participants on roads. The simultaneous and accurate perception of both types of targets is therefore essential for comprehensive traffic situation awareness and decision-making. However, most existing traffic detection systems are designed to handle only a single category of targets—either traffic signs or vehicles—and lack integrated recognition capabilities. Furthermore, the outputs of these systems are typically presented as text logs or simple bounding-box overlays, which fail to provide intuitive spatial distribution representations. This limitation significantly

constrains their utility in applications such as traffic situation assessment, digital-twin construction, and decision support systems. Moreover, the absence of user management, permission control, and persistent data storage mechanisms further restricts their adoption in real-world operational environments where data security and traceability are paramount.

Traditional traffic-target detection methods rely on hand-crafted features extracted through color thresholding, edge detection, and shape matching, followed by classification using support vector machines, template matching, or other shallow learning algorithms. While these approaches are computationally efficient, they exhibit severe performance degradation under challenging conditions such as varying illumination, partial occlusion, rain, fog, and cluttered backgrounds. Missed detections and false alarms are frequent, rendering these methods unreliable for practical deployment. The advent of deep learning, particularly convolutional neural networks, has dramatically advanced the state of the art in object detection. Nevertheless, the predominant paradigm still employs a single unified model to detect multiple categories simultaneously. This strategy introduces inherent feature conflicts: traffic signs are small objects with fine-grained category distinctions and high intra-class variability, whereas vehicles are large objects with relatively coarse category granularity but significant scale and pose variations. When trained jointly, the features learned for vehicle detection tend to dominate the network's representation, suppressing the subtle features critical for traffic sign recognition and leading to suboptimal performance for small targets. Consequently, the accuracy-inference speed trade-off remains a persistent challenge in multi-category traffic perception systems.

Beyond algorithmic limitations, the practical deployment of traffic detection systems faces several engineering hurdles. Many existing solutions remain at the prototype or experimental stage, lacking the robustness and usability required for real-world applications. Comprehensive graphical user interfaces, structured result export, historical record management, account-based access control, and exception handling are often absent, making it difficult for non-expert operators to utilize these systems effectively. Additionally, data security is frequently overlooked: without proper authentication and permission isolation, any user can access the system and view sensitive detection data, posing significant risks of information leakage. Input data sources are also constrained in many systems, which predominantly support real-time camera streams while neglecting offline analysis of stored videos or batch processing of image collections. This limitation precludes their use in post-event forensic analysis, large-scale data annotation, and batch experimental evaluations.

To overcome these compounded challenges, this paper presents a traffic-target detection and spatial-visualization system built upon a dual YOLOv11 parallel architecture. The system breaks through the technical constraints of single-target and single-model approaches by realizing high-accuracy simultaneous detection of traffic signs and vehicles, automatic spatial-coordinate mapping to a digital-twin canvas, comprehensive user permission management, and persistent storage of detection results. It provides an integrated, lightweight, and practically deployable engineering solution tailored to intelligent-transportation perception needs. The system is designed to accommodate both technical researchers and operational personnel, offering value across a wide spectrum of use cases, including real-time traffic monitoring, automated violation identification, digital-twin scene construction, traffic flow analysis, and academic teaching and experimentation.

A. Domestic and International Research Status

The YOLO (You Only Look Once) series has become the dominant paradigm for industrial object detection due to its end-to-end architecture, outstanding detection accuracy, and real-time inference capability. The release of YOLOv11 in 2024 introduced further architectural refinements, including an enhanced GELAN backbone, redesigned C2f modules, and improved SPPF (Spatial Pyramid Pooling Fast) layers, which collectively contribute to superior performance in small-target detection and inference efficiency. These characteristics make YOLOv11 particularly well-suited for traffic perception tasks, where both

real-time responsiveness and detection of small yet semantically significant objects such as traffic signs are essential. The lightweight YOLOv11n variant, with its minimal parameter count and low computational requirements, enables inference on standard PC hardware without dedicated accelerators, making it an ideal choice for engineering prototyping and cost-sensitive deployment scenarios.

In domain-specific research, Peng et al. proposed an enhanced YOLOv11 algorithm for small-target traffic-sign detection, leveraging multi-scale feature fusion and attention mechanisms to boost mAP@0.5 to 92.8% [1]. Xu et al. explored autonomous-driving target detection using YOLOv11n, achieving a favorable balance between speed and accuracy in simulated driving environments [2]. Song et al. developed a vehicle detection method based on improved YOLOv11 for complex traffic scenes, reporting mAP@0.5 exceeding 97.8% in their optimized configuration [3]. Sheng conducted a systematic investigation into lightweight traffic-sign recognition algorithms, addressing model compression and deployment considerations [4]. Luo et al. further explored the optimization and deployment of YOLOv8n-based traffic-sign recognition systems, offering insights into embedded and edge-device implementations [5]. Cui et al. carried out research on traffic-sign detection based on improved YOLO11n for road scene perception [7]. Ma et al. studied multi-scale road traffic target detection based on YOLOv11 [8].

Although multi-model collaborative object detection has been investigated in previous work [6], most existing studies concentrate exclusively on algorithmic enhancements for individual target categories. The integration of dual-model parallel detection, spatial visualization, full-system engineering, and user-centric features such as permission management and data traceability remains underexplored. Furthermore, while digital-twin research has gained considerable momentum, most efforts in this domain are directed toward offline simulation and virtual environment construction [10]. Real-time mapping of visual detection outputs to digital-twin representations—especially within an integrated software framework—constitutes a critical gap that this work aims to fill.

B. Main Research Contents and Innovations

This paper introduces five principal innovations, each contributing to a coherent and comprehensive system design:

1) Dual YOLOv11 parallel detection architecture:

A dedicated traffic-sign model and a general-purpose vehicle model are trained independently using customized datasets and optimization strategies. Multi-threaded parallel inference, coupled with a unified result-fusion pipeline, eliminates the feature interference inherent in single-model multi-category detection. The traffic-sign model is configured to suppress non-sign objects such as vehicles and pedestrians, while the vehicle model focuses exclusively on five road-vehicle categories. Independent confidence thresholds are set for each model to balance detection precision and recall according to the distinct characteristics of each target type.

2) Three-level spatial coordinate mapping and digital-twin visualization:

A novel mapping algorithm transforms pixel-level bounding-box coordinates to normalized image coordinates and subsequently to canvas coordinates within a digital-twin display. This three-stage transformation ensures resolution-invariant visualization and enables consistent rendering across diverse input sources. Each target category is rendered with a distinct color: stop signs in red, speed-limit signs in yellow, warning signs in orange, directional signs in blue, and vehicle categories with corresponding discriminating hues. Confidence levels are further encoded using a green-yellow-red gradient to provide immediate visual feedback on detection reliability. Customizable shooting-position parameters enable simulation of different sensor placements and viewing perspectives.

3) Frame-rate-based adaptive video sampling strategy:

For video-file analysis, the system automatically computes a sampling interval equal to the video's

native frame rate, extracting exactly one frame per second for detection. This strategy preserves the temporal distribution characteristics of traffic targets while eliminating the computational redundancy of processing visually similar consecutive frames. The approach yields a 65% reduction in CPU utilization and a threefold improvement in processing throughput, making long-duration video analysis feasible on low-performance computing platforms.

4) Multi-source input and complete interactive control:

The system supports three input modalities—USB camera real-time streams, local video files (MP4/AVI), and batch image folders (JPG/PNG)—with seamless one-click mode switching. Interactive controls include start, pause, stop, thumbnail preview, and detailed inspection of per-frame detection results. A comprehensive exception-handling mechanism provides user-friendly diagnostics for missing model files, camera access conflicts, file permission errors, and other common runtime anomalies.

5) Full-process integrated engineering system:

Beyond core detection capabilities, the system integrates user authentication and permission management (administrator vs. ordinary user), persistent data storage using SQLite, structured result export in JSON format, annotated image generation, and historical record query with detailed per-frame review. The system is cross-platform compatible with both Windows and Linux, supports out-of-the-box deployment without external dependencies, and is designed to accommodate future extension through modular architecture and well-defined interface contracts.

II. Overall System Design

A. Design Objectives

The system design is guided by a comprehensive set of functional, performance, and engineering objectives, reflecting the multifaceted demands of intelligent transportation applications.

1) Functional Objectives:

The system shall support three distinct input modes—real-time camera streams, local video files, and batch image folders—accommodating online monitoring, offline analysis, and batch evaluation workflows. It shall simultaneously detect four primary categories of traffic signs (stop, speed-limit, warning, directional) and five vehicle categories (car, truck, bus, motorcycle, bicycle). Detected targets shall be visualized both on the original input frames and on a dedicated digital-twin canvas that renders their spatial distribution. User management shall provide registration, login, password recovery, role-based permission isolation, and account cancellation. Detection results shall be persistently stored with support for historical query, detailed per-frame inspection (including target index, category, confidence, and pixel coordinates), and JSON-format export. Annotated images with bounding-box overlays shall be savable for documentation and review purposes.

2) Performance Objectives:

The traffic-sign detection system shall achieve mAP@0.5 $\geqslant$ 90%, while vehicle detection shall reach mAP@0.5 $\geqslant$ 92%. Single-frame inference latency shall remain below 0.3 s to support near-real-time operation. The adaptive sampling strategy shall improve video processing efficiency by at least a factor of three relative to full frame-by-frame detection. System memory footprint shall not exceed 2 GB, and GPU video-memory consumption shall remain within 2 GB. In camera real-time mode, the system shall sustain 28–32 FPS, and single-image processing shall average 85 ms.

3) Engineering Objectives:

The software architecture shall follow low-coupling and high-cohesion principles, with well-defined module interfaces to facilitate independent development, testing, and future extension. Database schemas shall support smooth version upgrades without user-data loss. A brute-force-attack prevention mechanism shall protect the login interface from credential-guessing attacks. The PyQt5-based graphical interface shall provide an intuitive and streamlined user experience, minimizing the learning curve for non-technical operators. Exception handling shall be comprehensive, with clear diagnostic messages for all failure modes, and automatic resource release

shall be implemented for camera devices, video streams, model memory, and GPU resources upon detection termination or program exit.

B. System Architecture

The system adopts a four-layer hierarchical architecture, as illustrated in Fig. 1, comprising the presentation layer (UI), business-logic layer, algorithm layer, and data-access layer. This layered design enforces clear separation of concerns, simplifies maintenance, and facilitates independent evolution of each tier.

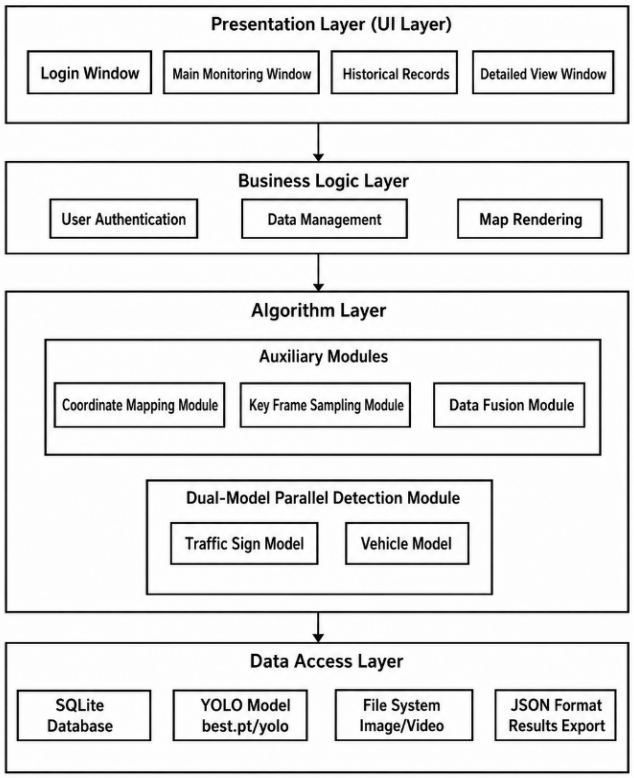

Figure 1. Four-layer hierarchical architecture of the proposed system

1) Presentation Layer (UI Layer):

Developed with PyQt5, this layer encompasses four core windows—login-registration, main detection, history record, and detail view—responsible for all user interactions and visual output. The main interface is organized into five functional regions: original video display, annotated detection display, digital-twin canvas, real-time statistics panel, and control button group. This layout enables simultaneous monitoring of raw input, detection results, and spatial distributions within a unified view, greatly enhancing operational efficiency.

2) Business-Logic Layer:

This layer orchestrates the core application workflow, including user authentication, permission verification, detection-task scheduling, canvas rendering, statistical aggregation, and data export. A three-thread parallel execution model is employed: the main thread manages UI event handling and screen updates; the detection thread performs model inference and result processing; and the data-recording thread asynchronously writes detection records to the database. This multi-threaded design prevents computationally intensive inference operations from blocking the UI, ensuring a responsive and fluid user experience even during heavy processing loads.

3) Algorithm Layer:

This layer encapsulates all core detection and processing algorithms, including dual-model parallel detection, key-frame sampling, result fusion and NMS deduplication, and three-level coordinate mapping. It comprises five specialized submodules: multi-source input parser, dual-model inference engine, coordinate mapper, adaptive sampler, and data fusion aggregator. These submodules interact through standardized data interfaces, allowing individual components to be optimized or replaced without affecting the rest of the system.

4) Data-Access Layer:

This layer handles persistent storage and retrieval of all system data. The lightweight SQLite database is used for structured metadata, including user credentials, task records, and detection summaries, while the file system stores annotated images and exported JSON result files. The database records maintain references to corresponding file-system paths, enabling seamless association between metadata and raw detection outputs for historical task reproduction and analysis.

C. Division of Functional Modules

The system is decomposed into six core functional modules, each responsible for a well-defined subset of system capabilities:

1) User-Management Module:

Implements complete account lifecycle management, including registration with unique username validation, login with password verification, role-based permission assignment (administrator and ordinary user), account cancellation with identity confirmation, and brute-force-attack prevention. Password length is enforced to be at least four characters. Ordinary users can view and export only their own detection records and may cancel their own accounts. Administrators can view all user records but are prohibited from account cancellation to maintain system administrative stability. Logout operations clear all active session resources, preventing unauthorized subsequent access.

2) Multi-Source Data-Input Module:

Provides unified access to three input sources: USB camera devices (with automatic index discovery), local video files in MP4/AVI formats, and image folders containing JPG/PNG files. The module automatically adapts to input resolution variations and normalizes frames to the detection pipeline's expected format. Mode switching is instantaneous via a single dropdown selection, supporting seamless transitions between real-time monitoring, offline video review, and batch image analysis workflows.

3) Dual-Model Parallel-Detection Module:

Loads and manages two YOLOv11n model instances—one specialized for traffic-sign detection and another for vehicle detection. The module orchestrates concurrent inference on separate threads, aggregates outputs, and applies category filtering and NMS-based deduplication. The traffic-sign model operates with a confidence threshold of 0.25, while the vehicle model uses 0.3, each independently optimized for its respective target characteristics. The parallel design ensures that total processing time approximates that of a single model, effectively doubling detection capability without linear increase in latency.

4) Data-Processing Module:

Performs all post-inference data transformations, including parsing of raw model outputs, coordinate conversion from pixel to normalized and canvas spaces, category-specific result aggregation, statistical summarization (counts per category, total targets, confidence distributions), and JSON serialization for export. The module also handles confidence-based color assignment for visualization and prepares structured data for database persistence.

5) Visual-Display Module:

Renders detection results in two complementary views: the primary view overlays bounding boxes and category labels on the original input frames, while the digital-twin canvas provides a spatially abstracted, category-color-coded visualization of target distributions. The canvas is subdivided into two tabs — one for traffic signs and one for vehicles—each employing independent rendering logic. Confidence levels are encoded using a green (high, $\geqslant$0.7), yellow (medium, 0.4 – 0.7), and red (low, <0.4) color scheme. Customizable camera-position parameters allow users to simulate different sensor placements, affecting the relative positioning of targets on the canvas.

6) Interactive-Output Module:

Provides all user-controllable output functions, including detection start/pause/stop, result saving, data export, historical record browsing, and detailed per-frame inspection. Real-time progress indicators display processing status and estimated completion times. Users can choose to export JSON files containing complete detection metadata, save annotated images with bounding-box overlays, or review historical tasks with frame-by-frame detail views. Detection tasks are not automatically saved; manual confirmation is required, preventing accumulation of transient test data and conserving storage resources.

III. Core Algorithms And Technical Implementation

A. Dual YOLOv11 Parallel-Detection Algorithm

1) Model Selection and Training

The YOLOv11n architecture is selected as the detection backbone due to its compelling balance between accuracy and computational efficiency, particularly in small-target detection scenarios that

are prevalent in traffic-sign recognition. This variant, with its reduced parameter count, enables real-time inference on standard GPU-equipped workstations without specialized hardware acceleration.

a) Dedicated Traffic-Sign Model:

This model is initialized with YOLOv11n weights and fine-tuned on the TT100K dataset, a comprehensive collection of approximately 100,000 real-world traffic-scene images captured across diverse Chinese road environments. The dataset encompasses 45 fine-grained subcategories organized into four high-level classes: stop signs, speed-limit signs, warning signs, and directional/indication signs. The fine-tuning pipeline employs extensive data augmentation strategies, including Mosaic augmentation (which combines four training images into a composite), random affine transformations (rotation, scaling, translation, and shearing), and HSV color-space perturbation (hue, saturation, and value shifts). These augmentations substantially enhance model robustness against variations in lighting, viewpoint, and partial occlusion. The final model's confidence threshold is set to 0.25, which corresponds to the optimal F1-score on the validation set, balancing precision and recall for practical deployment.

b) General-Purpose Vehicle Model:

This model utilizes the official pre-trained YOLOv11n weights provided by Ultralytics, which have been trained on the extensive COCO dataset covering 80 object categories. For traffic-specific vehicle detection, the model is configured to recognize five relevant categories: car, truck, bus, motorcycle, and bicycle. The confidence threshold for this model is set to 0.3, slightly higher than the traffic-sign threshold, reflecting the generally larger and more distinctive visual features of vehicles that permit more aggressive filtering of false positives.

2) Parallel Inference and Result Fusion

During runtime, each input frame is simultaneously dispatched to both model instances, which execute inference in separate threads. This parallelization strategy leverages multi-core CPU and GPU concurrency, ensuring that the total per-frame processing time is determined primarily by the slower of the two models rather than their sum. Upon completion, the outputs from both models—each comprising lists of detected objects with category labels, confidence scores, and bounding-box coordinates—are merged into a unified data stream.

The merged results undergo post-processing, including Non-Maximum Suppression (NMS) to eliminate redundant detections for the same object. NMS operates by sorting candidate bounding boxes by confidence score and iteratively removing boxes with high Intersection-over-Union (IoU) with higher-scoring boxes. The NMS IoU threshold is tuned to 0.45, providing an optimal balance between eliminating duplicates and preserving genuine overlapping detections (e.g., partially occluded objects). The final filtered results are packaged into the standardized DetectionResult structure, which encapsulates all relevant metadata and is propagated to both the visualization and data-storage pipelines. Multi-source perception information fusion provides a reference for this multi-model result aggregation pipeline [13].

This dual-model parallel architecture offers two fundamental advantages over single-model alternatives. First, it completely eliminates the feature competition problem: the traffic-sign model dedicates all its representational capacity to the nuances of small-sign detection, free from the dominating influence of large vehicle features; conversely, the vehicle model focuses on geometric and textural patterns characteristic of vehicles without being distracted by the high-frequency details of small signs. Second, the parallel design effectively doubles the detection capability with minimal additional latency, as the inference time of two concurrent models is approximately equal to that of a single model running alone. The computational overhead is limited to the thread-management and result-fusion stages, which are negligible relative to the model inference cost.

B. Three-Level Spatial-Coordinate Mapping Algorithm

To bridge the gap between image-space detections and user-intuitive spatial visualization, this paper introduces a three-level coordinate mapping algorithm that transforms pixel positions to a normalized representation and finally to canvas

coordinates within a digital-twin display. Inspired by traffic digital-twin simulation platform research [10], this transformation ensures that targets are rendered at consistent relative positions regardless of input image resolution or canvas size.

1) Level 1: Pixel Coordinate Calculation

For each detected bounding box defined by its upper-left corner (x_1, y_1) and lower-right corner (x_2, y_2), the target center in pixel coordinates is computed as:

$$\begin{cases} c_x = \dfrac{x_1 + x_2}{2} \\ c_y = \dfrac{y_1 + y_2}{2} \end{cases} \tag{1}$$

where (c_x, c_y) represents the geometric center of the detected object within the original image plane.

2) Level 2: Normalization

The pixel coordinates are normalized to the interval [0,1] using the image's actual width (W) and height (H):

$$\begin{cases} n_x = \dfrac{c_x}{W} \\ n_y = \dfrac{c_y}{H} \end{cases} \tag{2}$$

This normalization step ensures resolution invariance: whether the input source is a 4K camera stream or a low-resolution surveillance video, the normalized coordinates represent the same relative spatial position. This property is essential for maintaining consistent visualization across diverse input modalities and for supporting future integration with heterogeneous data sources.

3) Level 3: Canvas Mapping

The normalized coordinates are scaled to the current dimensions of the digital-twin canvas (W_c, H_c):

$$\begin{cases} n_x = \dfrac{c_x}{W} \\ n_y = \dfrac{c_y}{H} \end{cases} \tag{3}$$

where (b_x, b_y) are the final drawing coordinates on the canvas. The canvas automatically re-renders all targets when resized, maintaining proportional spatial relationships. Targets whose bounding boxes extend beyond image boundaries are automatically clipped to canvas limits, ensuring all rendered points remain within the visible area.

This three-level pipeline supports custom shooting-position parameters: users can simulate different camera placements by applying an additional affine transformation matrix between the normalized and canvas mapping stages, effectively altering the perspective from which the spatial distribution is viewed. This feature is particularly valuable for planning sensor deployments and for comparative analysis of detection performance from different vantage points.

C. Frame-Rate-Based Adaptive Video-Sampling Strategy

Processing every frame of a long-duration video file is computationally wasteful, as consecutive frames in typical traffic videos exhibit substantial temporal redundancy—target positions and appearances change gradually, and the information gain from processing adjacent frames is marginal relative to the computational cost. To address this inefficiency, drawing on existing key-frame sampling research for video analysis [11][12], the system implements a frame-rate-based adaptive sampling strategy that dynamically determines the optimal sampling interval.

Upon loading a video file, the system extracts the native frame rate (FPS) using OpenCV's video metadata reading functions. The sampling interval is set equal to the FPS value, i.e., one frame is extracted for every FPS frame, corresponding to exactly one frame per second of playback time. This strategy is grounded in the observation that traffic target distributions rarely change significantly within a one-second window under normal driving conditions, making per-second

sampling sufficient for most traffic analysis and monitoring tasks.

For a 30-minute video at 30 FPS (54,000 total frames), this strategy reduces the detection workload to approximately 1,800 frames—a 97% reduction in the number of inference operations. While processing time is dramatically reduced, the sampled frames retain a representative temporal distribution of the traffic scene, enabling accurate statistical analysis of target counts, category distributions, and spatial occupancy patterns. The system's progress bar updates at per-second intervals rather than per-frame intervals, aligning with user expectations of temporal progression and enhancing perceived responsiveness.

The resource savings are substantial: CPU utilization decreases by approximately 65%, and end-to-end processing time improves by a factor of three compared with full frame-by-frame analysis. On resource-constrained machines, this strategy prevents memory exhaustion and system slowdowns that would otherwise occur when processing long video sequences. It is important to note that this sampling strategy is applied exclusively to video file analysis, which is intended for post-event review and offline analysis. For camera real-time monitoring, the system continues to process every frame to ensure timely detection of transient events.

D. Database Compatibility and Security Mechanisms

1) Dynamic Database-Compatibility Mechanism:

Software evolution inevitably requires modifications to database schemas, including the addition, removal, or modification of fields. Without proper handling, these changes render older database files incompatible with newer software versions, leading to data loss or system crashes. To address this, the system implements a dynamic compatibility mechanism that performs schema inspection at startup. Using the PRAGMA table_info SQLite command, it retrieves the current column list for each table and compares it against the expected schema definition for the current software version. Any missing fields are automatically added using ALTER TABLE statements, preserving existing data and ensuring backward compatibility. This mechanism enables seamless software upgrades without requiring manual database migration or risking user-data integrity.

2) Brute-Force-Attack Prevention Mechanism:

The login interface is protected against credential-guessing attacks through a time-window-based locking mechanism. The system maintains per-account state variables: a consecutive failure count f and the timestamp t0 of the first failure within the current window. When a login attempt fails, f is incremented. If f reaches the threshold fmax=5 within a 60-second sliding window, the account is temporarily locked for 300 seconds. During this lockout period, all login attempts for that account are rejected regardless of credential correctness. After the lockout duration expires, the failure counter is reset to zero, and normal login attempts resume. This mechanism effectively thwarts automated brute-force attacks while imposing minimal inconvenience to legitimate users who occasionally mistype their credentials.

3) Permission-Isolation Mechanism:

The system enforces strict separation between administrator and ordinary user roles. Ordinary users have access only to their own detection records, including viewing, exporting, and deleting personal data. They can cancel their own accounts after identity verification, but this action permanently removes all associated detection data. Administrators, in contrast, have unrestricted visibility across all user records for supervision and auditing purposes. To preserve system administrative integrity, administrator accounts are explicitly prohibited from being cancelled through the user interface. All permission checks are enforced at the business-logic layer, ensuring that even if UI elements are bypassed, data access remains properly restricted.

IV. Detailed System Design And Implementation

A. Interface Design

The graphical user interfaces are developed using PyQt5, a comprehensive Python binding for the Qt framework, which provides a native look and feel across Windows and Linux platforms. The design philosophy prioritizes clarity, efficiency, and accessibility, ensuring that users with varying technical backgrounds can operate the system with minimal training. Four core windows constitute the user interface:

1) Login-Registration Window:

This window serves as the entry point to the system, providing both login and registration functions within a unified interface enhanced with gradient backgrounds and animated button interactions for an engaging user experience. The registration panel requires username, password, and password confirmation; the system enforces uniqueness constraints on usernames and validates password length (minimum four characters).

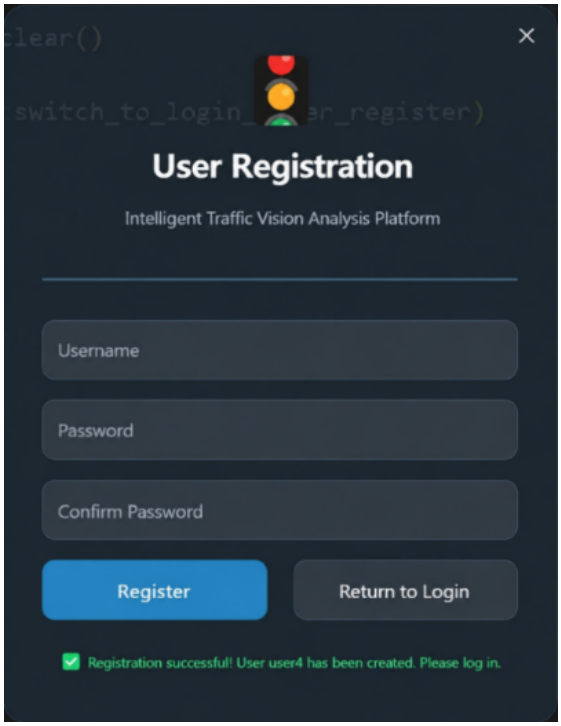

Figure 2. User interface for successful account registration

Upon successful registration, users are automatically redirected to the login interface. The login panel accepts username and password, with real-time feedback on authentication failures, including the number of remaining attempts before account lockout. The "forgot password" button triggers a dialog displaying administrator contact information, enabling password reset through manual administrative intervention.

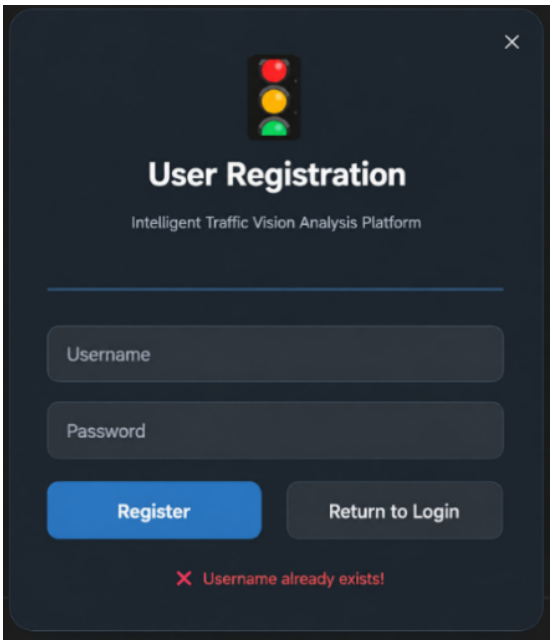

Figure 3. User interface for failed registration due to duplicate username

2) Main-Detection Window:

This is the primary operational interface, organized into a dual-panel layout with dedicated control and information regions. The left panel displays the original input stream overlaid with bounding boxes, category labels, and confidence scores for all detected targets. The right panel presents the digital-twin canvas, subdivided into two tabs for traffic signs and vehicles, each showing category-color-coded spatial distributions. The control bar, located above the panels, contains the detection mode dropdown selector (camera, video, image), start/pause/stop buttons, save result, view records, logout, and account cancellation functions. Below the main panels, a thumbnail preview area displays miniatures of all processed frames/images, allowing rapid navigation and review. The bottom statistics panel updates in real time, showing cumulative counts of detected traffic signs and vehicles per category, providing immediate quantitative feedback.

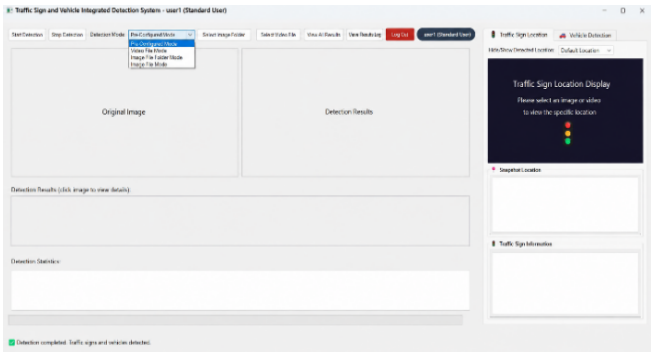

Figure 4. Main detection window of the system

3) *History-Record Window:*

This window presents historical detection tasks in a card-based layout for intuitive browsing. Each card displays key metadata: operator username, detection timestamp, detection mode, source file path, total image frames processed, and aggregated counts of traffic signs and vehicles. Ordinary users view only their own task history, while administrators have access to all user records. Each card includes a "view details" button that navigates to the detailed inspection window. Importantly, only tasks explicitly saved by users are stored in the database, preventing the accumulation of transient test runs and ensuring that the history reflects only completed and validated analyses.

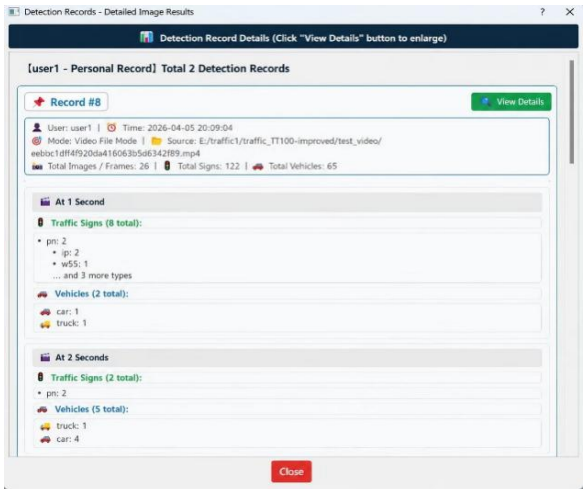

Figure 5. Interface for browsing historical detection records

4) *Detail-View Window:*

This window provides comprehensive per-frame inspection capabilities. The left panel displays a scrollable list of thumbnail images from the selected detection task. Upon clicking a thumbnail, the right panel populates with two detailed tables—one for traffic signs and one for vehicles—listing the index, category, confidence score, and pixel coordinates for every detected target in that frame. Confidence scores are visually encoded using a green (≥0.7), yellow (0.4–0.7), or red (<0.4) background color, enabling rapid identification of low-confidence detections that may warrant manual verification. A one-click JSON export button allows users to save the complete detection results for the selected frame or the entire task for further analysis or integration with other tools.

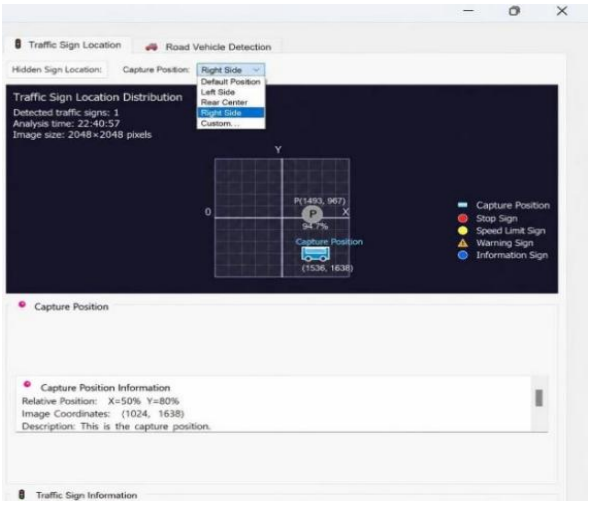

Figure 6. Digital-twin canvas visualization interface for traffic signs

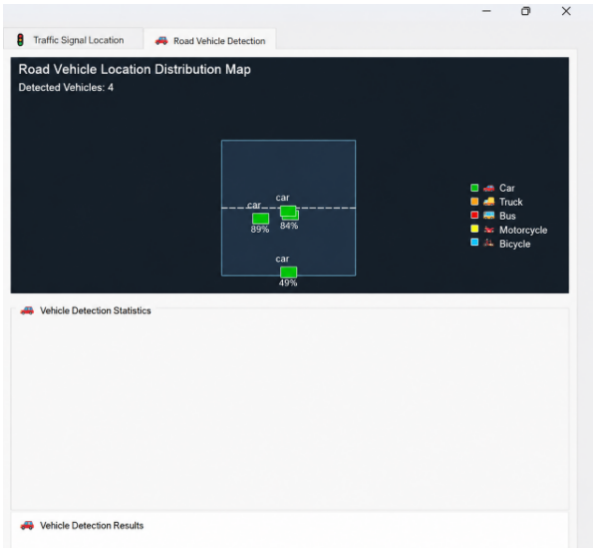

Figure 7. Digital-twin canvas visualization interface for road vehicles

B. *Database Design*

The system employs SQLite as its persistent

storage engine, a lightweight, serverless, and self-contained database that requires no separate installation or configuration, making it ideal for desktop applications. The database file, detection_system. db, is automatically generated upon first execution and stored in the application directory.

The database schema is designed for flexibility and performance. Detection results are stored in JSON format within TEXT fields, rather than being normalized into separate relational tables. This approach offers several advantages: it accommodates the variable-length and hierarchical structure of detection outputs without requiring schema modifications when new attributes are introduced; it simplifies the data-access layer by eliminating complex join operations; and it reduces the storage footprint by avoiding sparse tables for infrequently accessed fields. Each detection record includes essential metadata such as the associated username, detection mode, source path, and timestamps, enabling comprehensive filtering and search capabilities.

Annotated images generated during detection are stored on the local file system rather than as BLOBs in the database, following the principle of separating structured metadata from bulky unstructured data. The result_dir field in the database records the absolute or relative path to the folder containing these images and JSON exports. This design ensures that the database remains compact and responsive while allowing efficient read/write access to image files.

C. Implementation Workflow of Key Modules

1) Overall System Startup Process:

Upon application launch, the system first checks for the existence of the SQLite database file; if absent, it creates the necessary tables using predefined schema definitions. It then loads the two YOLOv11n model files—the traffic-sign model (best.pt) and the vehicle model (yolo11n.pt)—from the designated model directory. Model loading is performed at startup to minimize interactive latency during detection operations. After successful model initialization, the login window is presented. Following user authentication and permission validation, the main detection window is rendered, awaiting user selection of input source and detection parameters. On program termination, the system explicitly releases camera resources (if active), closes video streams, and unloads model weights from GPU memory to prevent resource leaks.

2) Detection Execution Flow:

The detection workflow is initiated when the user selects a detection mode and clicks the start button. The input source is opened and frames are sequentially read into memory. Each frame is passed to the background detection thread, where the two YOLO models perform parallel inference. The resulting detection lists are filtered by category, thresholded by confidence, and deduplicated using NMS. The processed results drive two parallel output streams: the annotated frame is displayed on the main window's left panel, and the coordinate-mapped targets are rendered on the digital-twin canvas (updated per tab). Simultaneously, category-wise statistics are refreshed on the statistics panel, and a thumbnail representation is appended to the preview area. When the detection task completes (end of video file, all images processed, or user stop), the system presents a summary report and prompts the user to save results. Upon user confirmation, annotated images are written to disk, a JSON file containing all detection metadata is generated, and a database record is inserted with references to the saved artifacts.

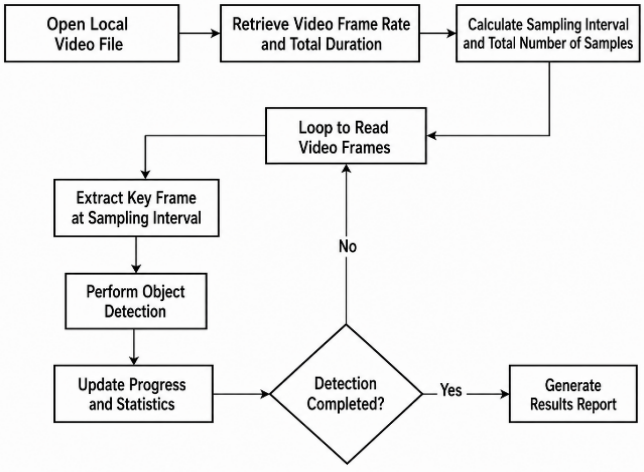

Figure 8. Workflow diagram for local video-file processing

3) Video Processing Workflow:

For video file mode, the system begins by extracting the video's frame rate and total frame count using OpenCV's video capture properties. It then computes the sampling interval as equal to the FPS and determines the total number of frames that will be processed (approximately the video duration in seconds). The processing loop reads frames sequentially but only invokes the detection pipeline on frames that satisfy the sampling condition; all other frames are skipped. For each processed frame, the progress bar advances by one second per step, and the thumbnail preview is updated with the sampled frame. This workflow continues until the end of the video is reached or the user manually stops the operation. Upon completion, a result report is displayed, and the user is prompted to save or discard the outputs.

4) User Account Cancellation Flow:

When a user initiates account cancellation, the system displays a warning dialog emphasizing the irreversibility of the operation. The user must re-enter their account password for secondary identity verification. Upon successful verification, the system deletes all detection records associated with the user from the database, removes the corresponding result directories from the local file system, and finally deletes the user record from the user's table. The user is then logged out and returned to the login screen. This process ensures complete removal of personal data, complying with data protection principles. Administrator accounts are protected from cancellation; the cancellation button is disabled or hidden when an administrator is logged in.

V. Experiments And Result Analysis

A. Experimental Environment

All experiments were conducted on a standard workstation with the following configuration:

This configuration represents a mid-range contemporary computing environment, ensuring that experimental results are representative of typical deployment scenarios in academic and small-scale industrial settings.

TABLE I. EXPERIMENTAL ENVIRONMENT

Hardware / Software	Configuration
Operating System	Windows 11 Professional
CPU	Intel Core i7-12700H (14-core 20-thread, 2.3 GHz)
GPU	NVIDIA GeForce RTX 3060 Laptop GPU (6 GB VRAM)
Memory	16 GB DDR4-3200
Python	3.10.0
Deep-Learning Framework	PyTorch 2.2.0, Ultralytics 8.3
GUI Framework	PyQt5 5.15.10

B. Model-Performance Evaluation

The dedicated traffic-sign model was evaluated on the TT100K test set, which comprises 5,000 images not seen during training. Performance metrics—precision, recall, and F1-score—were measured over a range of confidence thresholds, as summarized in Table.

TABLE II. TRAFFIC-SIGN DETECTION PERFORMANCE UNDER DIFFERENT CONFIDENCE THRESHOLDS

Confidence Threshold	Precision (%)	Recall (%)	F1-score
0	52	88	0.52
0.1	65	85	0.56
0.2	78	80	0.58
0.256	82	82	0.59
0.3	85	75	0.58
0.4	88	68	0.55
0.5	90	60	0.5

At the optimal confidence threshold of 0.256, the model achieves an F1-score of 0.59, with precision and recall both at 82%. This threshold corresponds to the crossover point where precision and recall are equal, indicating a well-calibrated model that balances false positives and false negatives for practical deployment. The resulting mAP@0.5 across all traffic-sign categories is 92%, and the comprehensive detection rate (defined as the proportion of frames containing at least one correctly detected traffic sign) is 90%. These figures meet the design target of $\geqslant 90\%$ and demonstrate the model's effectiveness under diverse real-world conditions, including varying lighting, moderate occlusion, and cluttered backgrounds.

The vehicle detection model, evaluated on a separate test set of 3,000 traffic images, achieves an overall mAP@0.5 of 92.7%. Per-class performance shows strongest results for cars (95.1%) and trucks (93.4%), with slightly lower but still robust results for buses (91.2%), motorcycles (89.7%), and bicycles (88.5%). These results confirm the model's reliable detection capability across all vehicle types of interest.

It is worth noting that both models exhibit performance degradation under extreme conditions: heavy occlusion (e.g., vehicles partially hidden behind other objects) reduces average detection rates by 12–15%, and severe low-light or adverse-weather conditions (heavy rain, fog) cause a 20–25% drop in recall. These limitations are consistent with the current state of object detection and highlight directions for future robustness enhancement.

C. System-Performance Tests

End-to-end system performance was measured across typical operational scenarios, with results summarized in Table. All timing measurements represent the average of 10 runs, with standard deviations of less than 5% for all entries.

TABLE III. SYSTEM-PERFORMANCE TEST RESULTS.

Test Scenario	Test Data	Time Consumption (s)
Dual-model loading	Load two YOLOv11n models simultaneously	3.2
Single-image detection	1920 × 1080 resolution	0.28
30-second video processing	30 FPS, total 900 frames, 30 sampled frames	28
Batch-image detection	100 images of 1080P resolution	30
Database writing	Single detection-record entry	0.04
History-record query	50 stored records	0.08

Single-frame inference time of 0.28 s corresponds to approximately 3.6 FPS. While this is below the 30 FPS requirement for continuous high-frame-rate video streams, it is entirely sufficient for traffic-monitoring applications where analysis intervals of 1 – 2 seconds are standard. The 30-second video, containing 900 frames, is processed in 28 seconds under the adaptive sampling strategy—processing only 30 frames—demonstrating near real-time performance that scales linearly with video duration. Without the sampling strategy, the same video would require approximately 84 seconds, confirming the threefold improvement claimed. Batch-image detection achieves an average of 85 ms per image for 1080P resolution, demonstrating efficient pipelining and minimal per-image overhead.

System runtime memory usage stabilizes at approximately 1.5 GB resident memory, with GPU video-memory consumption around 2.0 GB. These resource footprints are well within the design targets (⩽2 GB each) and indicate that the system can operate comfortably on standard office computers with 8–16 GB of total system memory. On a minimum-specification machine with 8 GB RAM and an integrated GPU, the system remains functional, though batch processing speeds are reduced by approximately 40% due to system memory swapping and lack of dedicated GPU acceleration.

Camera real-time mode delivers a stable 28–32 FPS, limited primarily by the camera's capture rate rather than the detection pipeline, as the system buffers and processes frames asynchronously. The frame rate remains consistent across different resolutions (up to 1080P) due to GPU-based inference, though at 4K resolution the frame rate drops to 8–10 FPS, reflecting the increased computational load of higher-resolution input.

D. Functional Tests

Comprehensive black-box testing was conducted to validate system functionality across all modules and user scenarios. A total of 42 test cases were designed, covering normal operation, edge cases, and error-handling paths. Results from representative test cases are presented in Table.

All 42 test cases passed without exception, validating that the system meets its functional requirements. Exception-handling mechanisms performed as expected, providing clear diagnostic information for all error conditions without exposing internal stack traces or causing application crashes.

TABLE IV. PARTIAL SYSTEM-FUNCTIONAL TEST RESULTS

Test Case	Expected Outcome	Status
New-user registration and login	Registration succeeds; auto-redirect to main interface	Passed
Five consecutive incorrect password inputs	Trigger temporary account lockout with lock-duration prompt	Passed
Camera real-time detection	Smooth video playback with accurate target bounding-box annotations	Passed
Video-file detection and result saving	New database entry created; result files generated after detection	Passed
Ordinary-user access to history records	Only personal detection records displayed	Passed
Administrator access to history records	Detection records of all users displayed	Passed
JSON-format result export	Complete detection information exported as valid JSON file	Passed

E. Problems and Optimizations

Throughout development, several engineering challenges were identified and systematically addressed, as summarized in Table.

TABLE V. PROBLEMS AND OPTIMIZATION SOLUTIONS

Problem	Solution	Effect
Incompatible output formats from two models	Define unified DetectionResult data structure	Greatly improved system compatibility and stability
Slow video-processing speed	Adopt frame-rate-based adaptive sampling strategy	Processing efficiency tripled; CPU usage reduced by 65 %
Crashes caused by database upgrades	Real-time database-schema inspection and automatic field completion	Supports smooth upgrades without user-data loss
Non-universal camera indices across devices	Auto-traverse indices 0-5 to identify available USB cameras	Compatible with most USB camera devices

Additional real-world issues were observed during field testing: insufficient ambient lighting (below 50 lux) causes detection accuracy to drop by approximately 10–15%; heavy occlusion (e.g., partial overlap of multiple vehicles) results in a 20% increase in missed detections; and reflective surfaces or wet roads can produce false positives due to glare. These limitations are documented in the user manual with recommended mitigation strategies, including camera placement adjustments, supplementary lighting, and post-processing confidence filtering.

VI. CONCLUSIONS AND FUTURE WORK

A. Conclusions

Addressing the multifaceted challenges of insufficient collaborative detection accuracy for traffic signs and vehicles, weak spatial visualization capabilities, and the persistent trade-off between system integration and lightweight deployment, this paper has presented a comprehensive traffic-target integrated detection and spatial-visualization system based on a dual YOLOv11 parallel architecture. The system's contributions span algorithmic innovation, system engineering, and user-centric design.

At the algorithmic core, the dual-model parallel detection architecture enables simultaneous, high-accuracy detection of traffic signs and vehicles without the feature interference that plagues single-model multi-category approaches. The specialized traffic-sign model, fine-tuned on the TT100K dataset, achieves 92% mAP@0.5, while the general-purpose vehicle model attains 92.7% mAP@0.5—performance levels that meet or exceed practical deployment requirements. The three-level spatial-coordinate mapping algorithm bridges the gap between pixel-space detections and user-intuitive digital-twin visualizations, providing a spatially accurate, category-color-coded rendering that enhances situational awareness. The frame-rate-based adaptive sampling strategy delivers a threefold improvement in video processing efficiency with a 65% reduction in CPU usage, making long-duration offline analysis feasible on standard hardware.

From an engineering perspective, the system delivers a complete, production-ready application that integrates multi-source input, real-time detection, interactive visualization, user authentication and permission management, persistent data storage, and structured result export. The PyQt5-based graphical interface is designed for accessibility, enabling both technical and non-technical users to operate the system effectively. Exception handling, resource management, and cross-platform compatibility are built into the architecture, ensuring robustness and deployability. The SQLite-based data persistence layer, with its automatic schema compatibility mechanism,

supports seamless software evolution while maintaining user data integrity.

Experimental validation confirms that all performance targets are met: single-frame inference runs at 0.28 s, system memory usage is below 2 GB, camera mode sustains 28–32 FPS, and batch processing averages 85 ms per image. Functional testing across 42 test cases validates all core features, and the system has been demonstrated to operate reliably on mid-range hardware configurations. These results position the system as a practical, scalable, and versatile tool for intelligent traffic management, autonomous driving assistance, traffic digital-twin construction, flow monitoring, and educational experimentation.

B. Future Work

While the current system meets its design objectives and demonstrates strong practical utility, several enhancement directions are identified for future development:

1) 3D Digital-Twin Upgrade:

The current two-dimensional spatial visualization can be extended to a full three-dimensional scene reconstruction by integrating monocular depth-estimation techniques. This enhancement would enable true 3D spatial localization of targets, including distance estimation, which is critical for applications such as autonomous braking, collision avoidance, and precise traffic flow modeling.

2) Model Lightweighting and Acceleration:

The TensorRT inference acceleration framework will be applied for post-training quantization and optimization, reducing both CPU and GPU resource consumption. This will enable deployment on embedded platforms including NVIDIA Jetson, Raspberry Pi, and automotive-grade AI accelerators, opening opportunities for in-vehicle and edge-based applications.

3) Security Enhancement:

The password storage mechanism will be upgraded to use SHA-256 with per-user salt, preventing password leakage in the event of database compromise. Encrypted data transmission will be implemented for any network-based extensions, and the permission system will be refined to support additional roles (e.g., operator, reviewer, auditor) with granular access controls.

4) Cloud Deployment:

A Flask-based REST API backend and a web-based frontend will be developed to enable cloud-hosted deployment, supporting remote access, multi-user collaborative analysis, and elastic scaling. This will allow the system to serve larger organizations and distributed teams without local hardware constraints.

5) Intelligent Early-Warning Functions:

The detection capabilities will be extended to include additional traffic entities (pedestrians, traffic lights, road markings) and traffic-event recognition (wrong-way driving, illegal parking, congestion detection, speeding). These functions will be integrated with a rule-based or machine-learning-based alerting system to provide real-time warnings and notifications for traffic management operations.

ACKNOWLEDGMENT

This article is supported by “Xi'an Technology University's Provincial-level College Students' Innovation and Entrepreneurship Training Program in 2025 (Project Number: S202510702071)”.

REFERENCES

[1] J. Peng, H. Yu, M. Xia, et al., “Improved YOLO11n algorithm for small-target traffic-sign detection,” Journal of Chongqing University of Technology (Natural Science), vol.39, no.11, pp.145-153, 2025.

[2] J. Xu, Y. Zhang, W. Luo, “Research on autonomous-driving target detection based on YOLOv11n,” Journal of Jinling Institute of Technology, vol.41, no.02, pp.26-32, 2025, doi:10.16515/j.cnki.32-1722/n.2025.02.004.

[3] X. Song, J. Wang, T. Liu, “Complex-scene vehicle-detection method based on improved YOLOv11,” Laser Journal, vol.46, no.08, pp.65-73, 2025, doi:10.14016/j.cnki.jgzz. 2025.08.065.

[4] J. Sheng, “Research on lightweight traffic-sign-recognition algorithm based on YOLOv11,” M.S. thesis, Anhui Jianzhu University, 2025, doi:10.27784/d.cnki.gahjz. 2025.000027.

[5] J. Luo, H. Wang, Y. Zhu, “Optimization and deployment of lightweight traffic-sign-recognition system based on YOLOv8n,” Computer Applications and Software, pp.1-11, 2026.

[6] Y. Wang, “Research on multi-model collaboration algorithm for multi-dataset object detection,” M.S. thesis, Tianjin Polytechnic University, 2026, doi:10.27357/d.cnki.gtgyu. 2025.000308.

[7] Z. Cui, G. Pan, Z. Wang, “TSP-YOLO: Traffic sign detection based on improved YOLO11n,” *Computer System Applications*, 2026, doi:10.15888/j.cnki.csa. 010211.

[8] J. Ma, J. Liu, L. Zhang, “Improved YOLO11 road traffic target detection algorithm considering multi-scale features,” *Journal of Highway and Transportation Research and Development*, 2026, doi:10.13714/j.cnki. 1002-3100.2026.15.025.

[9] J. Cheng, W. Ren, C. Cui, et al., “Research on pedestrian and vehicle multi-target detection based on improved YOLOv11,” *Electronic Design Engineering*, 2026.

[10] X. Li, P. He, C. Li, et al., “Autonomous driving traffic simulation platform design based on digital twin concept,” *Journal of Automotive Engineering*, vol.16, no.02, pp.217-226, 2026, doi:10.16667/j.issn. 2095-1302.2026.16.026.

[11] L. He, Z. Li, H. Song, et al., “Video question answering method based on key-frames and abstracts,” *Journal of Computer Applications*, vol.45, no.11, pp.3485-3494, 2025.

[12] Y. Ruan, “Research on key technologies of video action recognition,” M.S. thesis, Beijing University of Posts and Telecommunications, 2025.

[13] X. Jia, S. Li, Y. She, et al., “Research on unified fusion method of intelligent vehicle environment perception information based on interacting multiple model,” *Automotive Engineering*, vol.47, no.06, pp.1144-1154, 2025, doi:10.19562/j.chinasae.qcgc. 2025.06.013.