JOINT TRANSPORTATION
RESEARCH PROGRAM
INDIANA DEPARTMENT OF TRANSPORTATION
AND PURDUE UNIVERSITY

Further Refinement and Integrated Platform for INDOT Traffic Management and Safety Toolset

Stanley Chien, Shu Hu, Yaobin Chen, Mei Qiu, William Lorenz Reindl, Noah Meng, Liya Koshy, and Sajan Kumar

SPR-4930 • Report Number: FHWA/IN/JTRP-2025/36 • DOI: 10.5703/1288284318603

RECOMMENDED CITATION

Chien, S., Hu, S., Chen, Y., Qiu, M., Reindl, W. L., Meng, N., Koshy, L., & Kumar, S. (2025). *Further refinement and integrated platform for INDOT traffic management and safety toolset* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2025/36). West Lafayette, IN: Purdue University. https://doi.org/10.5703/1288284318603

AUTHORS

Stanley Chien, PhD
Professor of Electrical and Computer Engineering
Purdue University
(317) 374-4854
yschien@purdue.edu
Corresponding Author

Shu Hu, PhD
Assistant Professor of Applied and Creative Computing
Purdue University

Yaobin Chen, PhD
Director of TASI
Professor of Electrical and Computer Engineering
Purdue University

Mei Qiu, PhD
Former Graduate Student
Electrical and Computer Engineering
Purdue University

William Lorenz Reindl
Undergraduate Student
Electrical and Computer Engineering
Purdue University

Noah Meng
Undergraduate Student
Electrical and Computer Engineering
Purdue University

Liya Koshy
Graduate Student
Electrical and Computer Engineering
Purdue University

Sajan Kumar
Graduate Student
Electrical and Computer Engineering
Purdue University

ACKNOWLEDGMENTS

This work was supported by the Joint Transportation Research Program (JTRP), administered by the Indiana Department of Transportation (INDOT) and Purdue University. The authors would like to thank all members of the INDOT Study Advisory Committee (SAC), including Shuo Li, Project Advisor (PA); Nathan Shellhamer, the Business Owner (BO); Jack Gallagher; Randy Myers; Mark Muenz; Michelle Witkowski; and Drew Sorenson, for their guidance, advice, and technical support throughout the project period. Special thanks go to Dr. Darcy Bullock, Director of JTRP, and his staff at Purdue University for their administrative support and service.

JOINT TRANSPORTATION RESEARCH PROGRAM

The Joint Transportation Research Program serves as a vehicle for INDOT collaboration with higher education institutions and industry in Indiana to facilitate innovation that results in continuous improvement in the planning, design, construction, operation, management and economic efficiency of the Indiana transportation infrastructure. Learn more at engineering.purdue.edu/JTRP.

Published reports of the Joint Transportation Research Program are available at docs.lib.purdue.edu/jtrp/.

NOTICE

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA/IN/JTRP-2025/36		

4. Title and Subtitle	5. Report Date
Further refinement and integrated platform for INDOT traffic management and safety toolset	December 5, 2025
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Stanley Chien, PhD (https://orcid.org/0000-0002-1492-9959) Shu Hu, PhD (https://orcid.org/0000-0003-1446-4140) Yaobin Chen, PhD (https://orcid.org/0000-0002-9875-4083) Mei Qiu, PhD William Lorenz Reindl Noah Meng Liya Koshy (https://orcid.org/0009-0008-5238-2043) Sajan Kuma	FHWA/IN/JTRP-2025/36

9. Performing Organization Name and Address	10. Work Unit No.
Joint Transportation Research Program Hall for Discovery and Learning Research (DLR), Suite 204 207 S. Martin Jischke Drive West Lafayette, IN 47907	11. Contract or Grant No. SPR-4930

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Indiana Department of Transportation (SPR) State Office Building 100 North Senate Avenue Indianapolis, IN 46204	Final Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Conducted in cooperation with the U.S. Department of Transportation, Federal Highway Administration.

16. Abstract

The project aims to further refine the previously developed Anomaly Detection and Weaving Analysis programs and create a unified, Web-based user interface for users to access TASI-developed software tools conveniently.

1. The INDOT Application Suite has been successfully developed for centralizing the digital platform developed by Purdue University's Transportation and Autonomous Systems Institute (TASI). It offers a secure, single point of access to a range of specialized applications, consolidating multiple tools into a unified system.
2. A scalable, lane-wise highway traffic anomaly detection framework that operates exclusively on video-based data was designed. By leveraging AI-driven vision techniques, the anomaly detection system extracts interpretable lane-specific traffic features, including vehicle count, occupancy, and truck percentage. We introduced a novel dataset derived from real-world highway surveillance, capturing fine-grained traffic dynamics across multiple time periods and anomaly types.
3. The Highway Weaving Analysis Program has been successfully developed. Due to the technical challenges of vehicle re-identification from CCTV footage, the automated matching process can produce false positives. To address this, the system incorporates a human-in-the-loop verification step in the *Web-Based Vehicle Matching Verification and Result Generation GUI*. Based on this verified data, the system calculates the final lane-to-lane weaving percentages. The results are presented in a comprehensive, user-friendly report that includes Sankey diagrams visualizing the traffic flow for all vehicles, as well as for cars and trucks separately.

The entire system operates on a cloud maintained by Purdue University.

17. Key Words	18. Distribution Statement
road-dependent anomaly detection, road-independent anomaly detection, lane-wise traffic analysis, multi-model anomaly detection, weaving analysis	No restrictions. This document is available through the National Technical Information Service, Springfield, VA 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	46	

EXECUTIVE SUMMARY

Introduction

From 2019 to 2022, Transportation Autonomous Systems Institute (TASI) at Purdue University and the Indiana Department of Transportation (INDOT) collaborated to develop an artificial intelligence (AI)-based method utilizing highway camera videos to detect lane-based traffic flow in real-time. A software tool was successfully installed and deployed on INDOT's computer systems. This project received a 2023 American Association of State Highway and Transportation Officials (AASHTO) award. In 2022–2023, using the knowledge gained from this project, TASI and INDOT developed lane-based, origin-destination, vehicle-count software that uses highway or drone camera footage. This software can significantly enhance the productivity of weaving analysis and reduce costs. The JTRP (SPR-4855) project identified anomalies using traffic data captured from specified highway cameras and demonstrated the feasibility of training and using AI anomaly detection models. This report aims to further improve the Anomaly Detection and Weaving Analysis programs and to create a unified web-based interface that allows users to conveniently access past and future TASI-generated software tools.

Findings

1. The INDOT Application Suite was successfully developed to centralize the software published by TASI. It offers a secure, single point of access to a range of specialized applications, consolidating multiple tools into a unified system. Designed with a user-focused approach, the platform aims to streamline access, improve efficiency, and support ongoing research. Key features include:
   - **Centralized Dashboard**: After logging in, users are directed to a main dashboard that displays all available applications as individual cards. Each card provides a quick overview and direct access to its corresponding tool, reducing the need for multiple logins and simplifying navigation.
   - **Application Management**: The platform allows administrators to add, deactivate, or remove applications as needed, ensuring flexibility and adaptability.
   - **Integrated Feedback System:** Users can report issues, suggest improvements, and track the status of their feedback through a built-in feedback loop. This promotes user involvement in the platform's design and ensures continuous improvement.
2. Our system employs AI-powered vision models to automatically detect lane boundaries and traffic directions and extracts lane-wise traffic features, such as vehicle count, occupancy, and truck percentage, from optimally learned detection regions. We focus on detecting sequential or collective traffic anomalies using these lane-wise features as input, where anomalies arise not from isolated outliers but from patterns of abnormal behavior over sequences of data points.
3. To address the limitations of the dataset, we introduce a novel dataset on lane-wise traffic anomalies derived from Indiana highway surveillance videos. Unlike previous datasets focused on urban intersections or aggregated data, our dataset provides structured time series signals for each segment of the highway lane, supporting both lane and road anomaly detection. Through extensive data analysis and expert collaboration, we define specific types of anomalies based on the collected data. We further propose a modular, multistream Anomaly Detection System capable of capturing both road-independent and road-dependent anomalies.
4. Our contributions and findings are summarized as follows:
   - We construct and release a novel lane-wise highway traffic dataset capturing segment-level flow dynamics from five highway cameras over six months, including 73,139 lane-wise samples with vehicle count, occupancy, and truck percentages.
   - We proposed an anomaly labeling pipeline combining machine learning, manual verification, and expert validation. Three types of traffic anomalies (lane blockage and recovery, foreign object intrusion, and sustained congestion) and one type of sensor anomaly are identified.
   - We developed a scalable anomaly detection framework integrating rule-based logic, machine learning, and deep learning to improve both robustness and interpretability.
   - The system can be seamlessly integrated into the INDOT surveillance platform.
   - The research paper for this project has been accepted by the IEEE International Conference on Intelligent Transportation Systems (ITSC) 2025.
5. The Highway Weaving Analysis Program has been successfully developed. To start a new analysis, the user assigns a unique name to the weaving scenario and uploads two video files: one for the entry point of the weaving section and one for the exit point. In the Weaving Area Specification GUI, the user graphically defines the roadway geometry on a frame-by-frame basis for each video. This includes specifying the number of lanes for each road, marking the center of each lane, and providing descriptive names for the roads and lanes. This user-provided information is crucial for both the accuracy of the analysis and the clarity of the final report. Once the user submits the scenario details, the system initiates the *Vehicle Detection and Matching Program*. Due to the technical challenges of vehicle re-identification from CCTV footage, the automated matching process can produce false positives. To address this, the system incorporates a human-in-the-loop verification step. After the backend processing is complete, the user is directed to the *Web-Based Vehicle Matching Verification and Result Generation GUI*. This interface presents the user with pairs of matched vehicle images from the entry and exit points. The user verifies whether each pair is a true or false match. Additionally, the user can verify the vehicle counts for each lane. Based on this verified data, the system calculates the final lane-to-lane weaving percentages. The results are presented in a comprehensive, user-friendly report that includes Sankey diagrams visualizing the traffic flow for all vehicles, as well as for cars and trucks separately.

Implementation

The entire system is deployed and running on a cloud maintained by Purdue University and is ready to be installed on the INDOT computer system. The research team will install the system on INDOT computers for road traffic monitoring operations. A detailed user instructional document was provided.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

From 2019 to 2022, the Transportation Autonomous Systems Institute (TASI) at Purdue University and the Indiana Department of Transportation (INDOT) collaborated to develop an artificial intelligence (AI)-based method that uses highway camera footage to detect real-time lane-based traffic flow. A software tool was successfully installed and deployed on INDOT's computer systems. This project received a 2023 American Association of State Highway and Transportation Officials (AASHTO) award. In 2022–2023, using the knowledge gained from this project, TASI and INDOT developed lane-based origin-destination vehicle count software, which utilizes highway or drone camera videos. This software can significantly enhance the productivity of weaving analysis and reduce costs. Leveraging the capabilities from TASI's preceding Joint Transportation Research Program (JTRP) projects, the JTRP (SPR-4855) project from Christopher et al. (2024) identified anomalies using traffic data captured from specified highway cameras. The project results have demonstrated the feasibility of training and using AI anomaly detection models. This project aims to:

1) Develop a unified computation, storage, and user interface environment together with INDOT to integrate the software developed for this proposed project and the software tools developed from several past successful TASI-conducted JTRP projects. This integrated software environment aims to enable a unified user interface and simplify system installation and maintenance for software tools developed in the past, and to specify the requirements and template for new tool development in the foreseeable future.
2) Further improve the anomaly detection method, which includes:
   a) Refine the capability of the AI anomaly detection model developed in SPR-4855 from Christopher et al. (2024) to make it more general and adapt it to other cameras.
   b) Extend the anomaly detection beyond the week/month to longer periods (year). With increased data, conduct an extended test of anomaly model robustness using known traffic variations (e.g., holidays, time of day, time of year).
   c) Allow retraining/updating the anomaly model (under user control) to adapt to changes in traffic patterns. This task also allows user feedback swith simple ground-truth tagging (anomaly or non-anomaly) to be used for updating the model.
3) Modify the weaving analysis program:
   a) Make the user interface web-based.
   b) Improve the readability of the Sankey diagram that shows the weaving analysis results.

All programming and testing of these tasks have been successfully completed in the Purdue computing environment, which is configured to mimic the INDOT-recommended computing environment. All software developed in this project will be installed in the INDOT computing environment.

2. THE UNIFIED ENVIRONMENT DEVELOPMENT

The INDOT Application Suite is a centralized digital platform developed by TASI. It offers a secure, single point of access to a range of specialized applications, consolidating multiple tools into a unified system. Designed with a user-focused approach, the platform aims to streamline access, improve efficiency, and support ongoing development. Key features include:

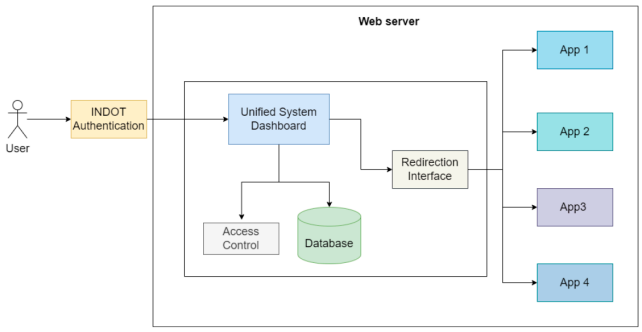


Figure 2.1 High-Level System Architecture of the INDOT Application Suite.

- **Centralized Dashboard**: After logging in, users are directed to a main dashboard that displays all available applications as individual cards. Each card provides a quick overview and direct access to its corresponding tool, reducing the need for multiple logins and simplifying navigation.
- **Application Management**: The platform allows administrators to add, deactivate, or remove applications as needed, ensuring flexibility and adaptability.
- **Integrated Feedback System:** Users can report issues, suggest improvements, and track the status of their feedback through a built-in feedback loop. This promotes user involvement in the platform's development and ensures continuous improvement.

2.1 Unified System Architecture

The system uses Portal Architecture. It serves as a central hub or main entrance, providing access to multiple separate applications. The structure of the unified system is shown in Figure 2.1.

Below is the step-by-step operation flow:

- Authentication: The user logs in once through a central INDOT Authentication service.
- Dashboard: After successful login, they are directed to the main Dashboard.
- Personalization: The Dashboard checks with the Access Control module to see what applications the user is allowed to access and pulls the application details from the Database. This creates a personalized view for the user based on their roles.
- Accessing an App: When the user clicks on an application card, a Redirection Interface securely sends them to the actual application.

The most critical feature of this design is its decoupled architecture. The central portal—comprising the dashboard, access control, and database—is functionally and technically separate from the individual end applications (App 1, App 2, etc.). This separation provides three key advantages:

- Resilience: The failure or maintenance of a single application does not affect the availability of the central portal or any other application.
- Flexibility: New applications can be integrated into the suite, or existing ones updated, with minimal impact on the core portal infrastructure.
- Scalability: The architecture allows both the portal and the individual applications to be scaled independently to meet demand.

In essence, the "unification" is achieved at the user experience level—providing one login and one central hub—while preserving the technical independence and robustness of a distributed system.

2.2 Authentication

2.2.1 Current Implementation

For development and testing, a provisional authentication mechanism has been implemented. This system is designed to be straightforward, allowing developers and stakeholders to focus on application functionality and user interface elements.

- Login Process: Users can currently access the system by entering any validly formatted email address and using a static, universal password.
- Session and Role Assignment: Upon this basic login, the system initiates a session and assigns a predefined role to that user. This role is used for Role-Based Access Control (RBAC) capabilities. Based on the assigned role, the dashboard dynamically displays the specific applications and features to which the user is authorized.

This simplified approach is intentional for this stage of development.

User Roles and Login Setup With Predefined Roles

The system employs a Role-Based Access Control (RBAC) model to manage permissions and tailor the user experience. RBAC is a security method that restricts access based on a user's assigned role. Instead of managing permissions for everyone, the system grants access rights to specific roles, and users inherit the permissions of the roles they are assigned.

Following this model, three primary roles have been established, each with predefined credentials for the development phase:

- Standard User: For general users who need access to applications and the ability to submit feedback.
- TASI Developer: For technical team members responsible for building, maintaining, and managing feedback for applications.
- Superuser (Administrator): For platform administrators with full privileges to manage applications, user roles, and system settings.

A dedicated App-User Role Mapping Table governs access to applications. This table determines which application cards are displayed on a user's dashboard based on their assigned role. Superusers have the authority to modify these role-based permissions for any application. By design, the Superuser and TASI Developer roles retain universal access to all applications, ensuring effective oversight and maintenance.

2.2.2 Planned Installation of the Applications on the INDOT Environment

The production version of the Application Suite is designed to integrate seamlessly with INDOT's official Single Sign-On (SSO) infrastructure. Based on initial discussions with INDOT IT teams, the technical work required to implement this integration will be handled directly by INDOT's internal IT teams, according to the following assumptions to ensure that all user management, password policies, and security protocols adhere strictly to INDOT's established enterprise standards, providing a secure and consistent experience for all users.

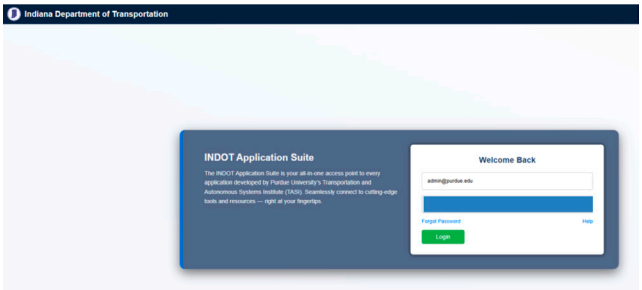


Figure 2.2 INDOT Application Suite Login Page.

- Delegated Authentication: In the target architecture, the suite will not handle user credential validation. Instead, it will delegate the entire authentication process to INDOT's enterprise system. When a user attempts to log in, they will be redirected to the standard INDOT login portal.
- Secure Session Management: Once the INDOT service successfully authenticates the user, it will securely pass the user's identity and session information back to the Application Suite. The suite will then use this trusted information to establish the user's session and apply the appropriate access control rules.

2.3 Core Platform Modules

This section provides a detailed examination of the primary interfaces and modules that constitute the user experience within the INDOT Application Suite, outlining the functionality from initial login to application management and feedback submission.

2.3.1 The Login and User Dashboard

The user's interaction with the suite begins with the secure login screen, which serves as the single, controlled point of entry for all personnel (see Figure 2.2).

Upon successful authentication, the system directs the user to the TASI Dashboard (Figure 2.3). This dynamic, personalized space acts as the user's command center, presenting all authorized tools and resources in an intuitive card-based layout.

- Role-Based Visibility and Personalization: The dashboard is intelligently filtered based on the user's assigned role. A Standard User, for instance, will only see the specific applications relevant to their duties, providing a clean, focused, and uncluttered workspace. In contrast, Superusers and TASI Developers are granted a comprehensive view by default, displaying all applications within the suite to facilitate systemwide management and development tasks.
- Direct and Seamless Application Access: A single click on any application card instantly launches the corresponding tool, eliminating the need for separate logins or navigation.

2.3.2 Application Management Module (Superuser Access)

The Application Management module is accessible exclusively to Superusers. It provides a comprehensive set of tools for managing the entire lifecycle of the platform's application portfolio (Figure 2.4).

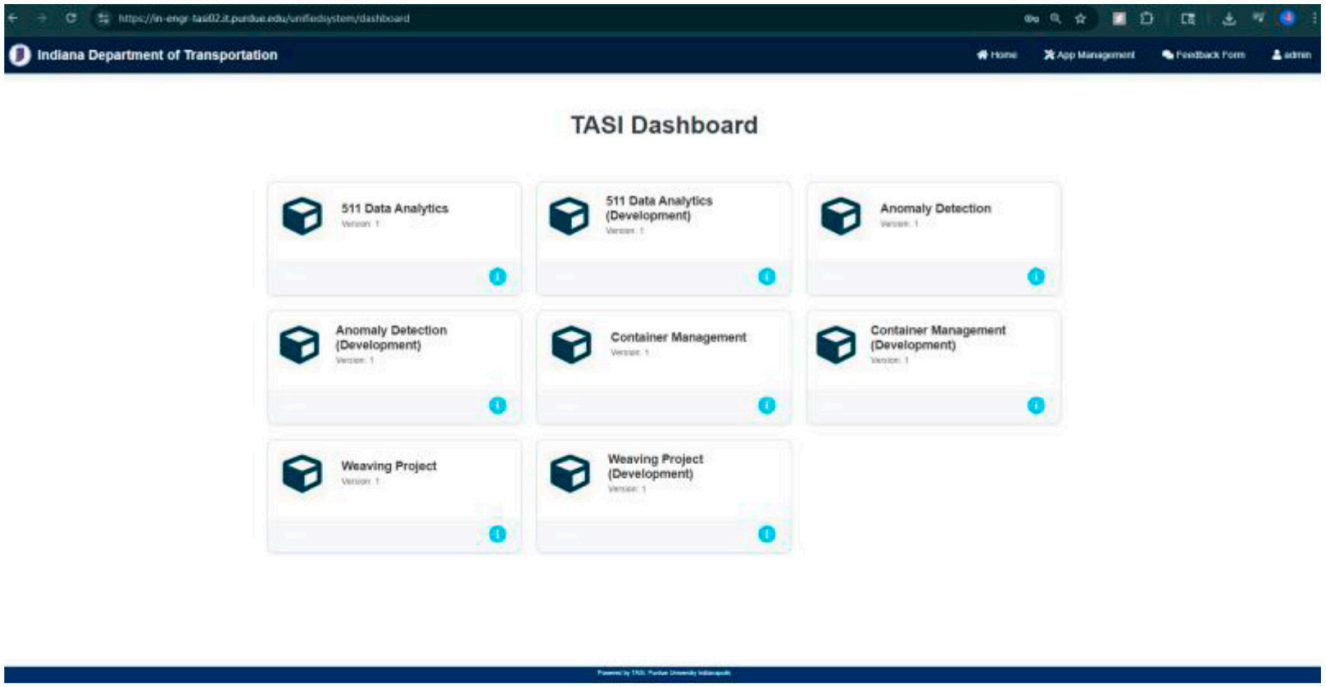


Figure 2.3 The TASI Dashboard is Personalized Based on the User's Role.

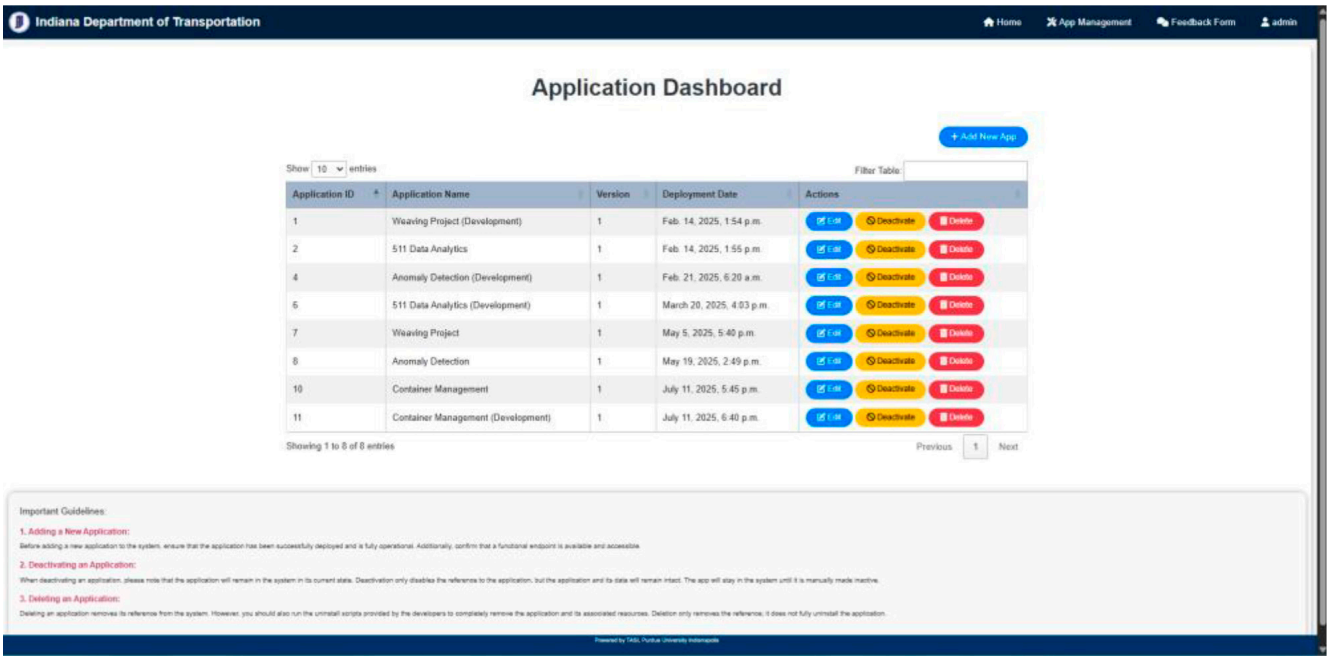


Figure 2.4 Application Management Dashboard.

The module's key functionalities are detailed below:

- Adding New Applications: Administrators can onboard new tools via a structured process. Clicking the "Add New App" button opens a modal form (Figure 2.5) where they must input essential metadata, including the application's official name, a detailed description, and the direct deployment URL. As a critical quality control measure**, an application must be fully deployed and tested before it can be registered in the suite**. Upon its first entry, the version is automatically set to "1."
- Editing, Versioning, and Role Assignment: The main view of this module features a searchable and paginated table that acts as a central registry for all applications. From here, an administrator can perform several key actions:
  - Edit:
    - Modify an application's metadata, such as its name or URL.

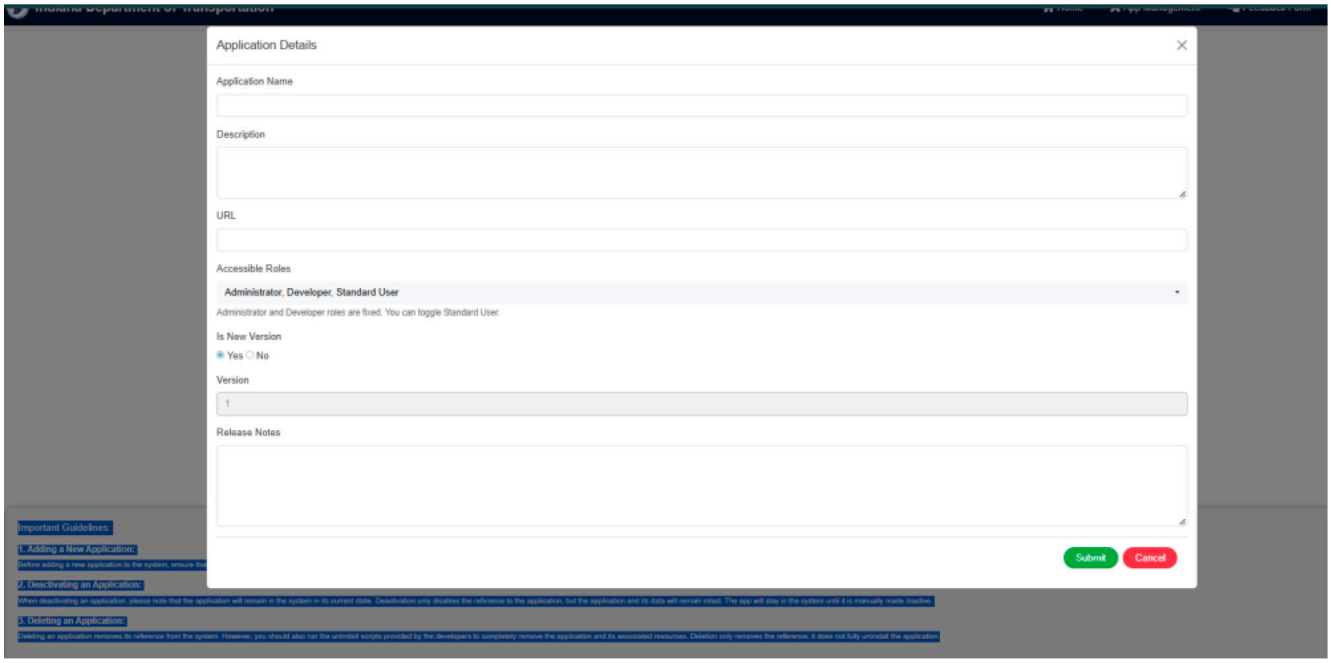


Figure 2.5 Pop-Up Window for Adding Application Details.

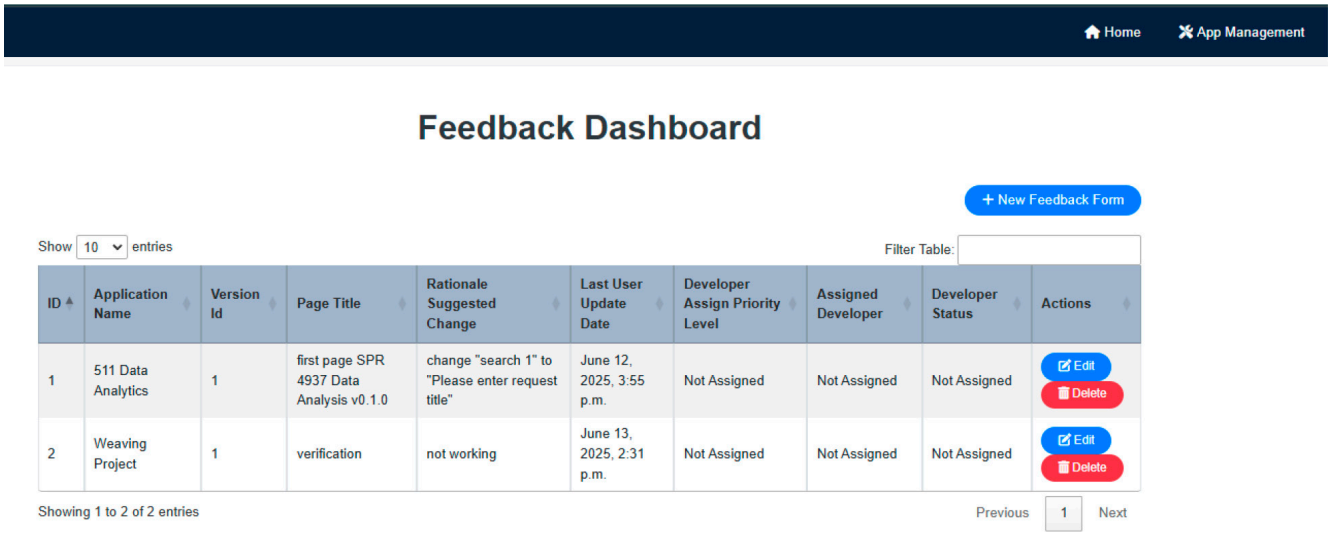


Figure 2.6 Feedback Dashboard.

- Assign Roles: Control application visibility by assigning it to specific user roles.
- Manage Versions: Update an application's release notes and increment its version number to track changes and inform users of new features or fixes.

– Application Removal: To decommission an application, administrators have two distinct options:

- Deactivate: A nondestructive, reversible action that hides the application from all user dashboards. This is ideal for temporary maintenance or for gradually phasing out a tool. The application's data is preserved, and it can be reactivated with a single click.
- Delete: A permanent, irreversible action that removes the application and all of its associated data from the system. This is reserved for applications that are fully retired.

2.3.3 Feedback System

The TASI Unified System incorporates a Feedback and Suggestion Form, establishing a direct and organized communication channel between the end-users and the development team. The main feedback dashboard presents submissions in a structured table (Figure 2.6) and users can add new feedback via a pop-up form (Figure 2.7).

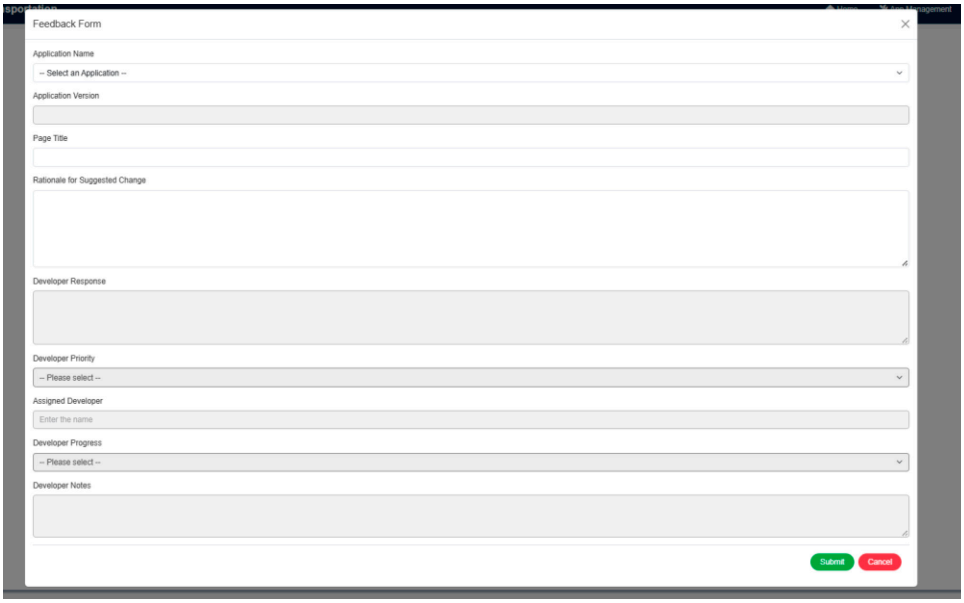


Figure 2.7 Feedback Pop-Up.

This system's workflow is governed by carefully defined permissions to ensure clarity and efficiency:

- Role-Based Submission and Visibility: A user can only submit feedback for applications they are authorized to access. To maintain context and relevance, the system also segregates visibility: Standard Users can only view feedback submitted by other Standard Users. Conversely, Superusers and TASI Developers have global visibility, allowing them to review all feedback submissions across all roles to identify trends and systemwide issues.
- Segregated Editing Permissions: To protect the integrity of the feedback process, editing rights are partitioned. Superusers (administrators) and standard users can edit the user-submitted fields of a feedback entry. However, they cannot modify the fields designated for the development team, such as "Developer Response," "Priority," or "Progress Status." When editing an entry, developers and other users can add more details as permitted by their roles. This structure ensures that the user's original request remains unaltered while giving the development team full autonomy over their internal workflow and response management.

2.4 Software Tools Used in Development

The INDOT Application Suite is built on a modern, robust technical stack designed for scalability and maintainability.

- Backend Framework: The core application is developed using Python with the Django framework, running on a Gunicorn WSGI server. The application is containerized using Docker for consistent deployment and scalability.
- Frontend Technology: The user interface is rendered using the Jinja templating engine, with styling and responsive design handled by the Bootstrap framework.
- Database: The primary data store is a PostgreSQL database (Unified_System_INDOT) hosted at ***in-engr-tasi02.it.purdue.edu.***
- Development Environment: The application was developed using the IntelliJ IDEA integrated development environment.

Core Dependencies: The project relies on the following key Python packages:

- asgiref==3.8.1
- asn1crypto==1.5.1
- backports.zoneinfo==0.2.1
- Django==4.2.17
- django-sslserver
- gunicorn
- psycopg2==2.9.10
- python-dateutil==2.9.0.post0
- scramp==1.4.5
- six==1.17.0
- sqlparse==0.5.3
- typing_extensions==4.12.2
- tzdata==2024.2
- whitenoise

2.4.1 Database

This database schema is designed for a centralized web portal that manages access to multiple applications. It uses a normalized structure to reduce data redundancy and relies heavily on foreign keys for data integrity. A key feature is the extensive use of database views to abstract complexity and provide simplified, ready-to-use datasets for the application's backend.

2.4.2 Core Table Definitions

These are the fundamental tables that store the platform's state.

- applicationdetails
  - Purpose: This table is the master catalog for every application managed by the suite. It stores the core, static information about each application.

  - Key Columns:
    - applicationid: A unique serial number (auto-incrementing integer) that serves as the primary key for an application. All other tables reference this ID.
    - applicationname: The official name of the application.
    - Description: A text field for a detailed description of the app's purpose.
    - websiteurl: The direct URL where the application is hosted.
- applicationversioning
  - Purpose: To track the release history and versioning of each application. This allows for documentation changes over time.
  - Key Columns:
    - deploymentid: The primary key for a specific version entry.
    - applicationid: A foreign key linking this version record to a specific application in applicationdetails.
    - version: The version number string (e.g., "1.0", "2.1.3").
    - releasenotes: A text field for developers to describe what is new in this version.
    - isactive: A Boolean flag to indicate if this is the currently active/displayed version.
    - Relationships: A one-to-many relationship with applicationdetails. The ON DELETE CASCADE clause means that if an application is deleted from applicationdetails, all of its associated version records will be automatically deleted as well.
- applicationstatus
  - Purpose: Manages the lifecycle status of an application, separate from its version. This allows an administrator to deactivate or "soft delete" an application without permanently removing its data.
  - Key Columns:
    - applicationid: A foreign key linking to applicationdetails.
    - isactive: A boolean for temporarily deactivating an app (hiding it from users).
    - isdeleted: A boolean for "soft deleting" an app. The application data remains in the database but is hidden from almost all queries.
    - deletedon: A timestamp to record when the soft delete occurred.
  - Relationships: A one-to-one relationship with applicationdetails. ON DELETE CASCADE ensures the status record is removed if the application is hard deleted.
- user_role
  - Purpose: Defines the set of roles available within the system, which is the foundation of RBAC.
  - Key Columns:
    - role_id: The primary key for a role.
    - role_name: The unique name of the role (e.g., "Superuser," "Standard User"). The UNIQUE constraint prevents duplicate role names.
- application_rolemapping
  - Purpose: This is the critical join table that implements RBAC. It explicitly defines which roles have access to which applications.
  - Key Columns:
    - applicationid: Foreign key to applicationdetails.
    - role_id: Foreign key to user_role.
  - Relationships: This table creates a many-to-many relationship between applications and roles. A single row in this table grants access for one role to one application. ON DELETE CASCADE cleans up mappings automatically if an associated app or role is deleted.
- lookuptable
  - Purpose: A highly flexible, generic table used to populate UI dropdown menus (e.g., for feedback priority or status). This design pattern avoids HARDCODING options in the application code, allowing administrators to modify them by just changing data in this table.
  - Key Columns:
    - table_name & column_name: These specify which form field the options belong to (e.g., table userfeedbacktable, column dev_progress_status_id).
    - lookup_option: The human-readable text displayed in the dropdown (e.g., "In Progress").
    - lookup_value: The actual value stored in the database (often the same as the option or a numeric ID).
  - Performance: An index (idx_lookup_table_column) is created on (table_name, column_name) to make fetching the options for a specific dropdown very fast.
- userfeedbacktable
  - Purpose: To store all feedback submissions from users for specific applications.
  - Key Columns:
    - applicationid: Foreign key linking feedback to a specific app.
    - rationale_suggested_change: The main text of the user's feedback.
    - developer_response: A field for the development team's public reply.
    - developer_assign_priority_id & dev_progress_status_id: Foreign keys that point to the lookuptable to get their values for "Priority" and "Status."

2.4.3 Database View Definitions

Views are saved queries that act like virtual tables. They are used here to simplify data retrieval for the application.

- Applicationstatusview: A simple view that joins applicationdetails and applicationstatus to show application info alongside its active/deleted status.
- Applicationstatuswithdeletedappsfilteredoutview: Purpose is imilar to the above, but it crucially filters out any applications marked as isdeleted = false. Provides a clean list of all nondeleted applications, likely used in adminrator panels.
- Applicationversionview: Joins applicationdetails with applicationversioning but only includes versions where isactive = true. Use Case: Quickly gets the current, active version details for every application.
- Applicationdetailedwithdeletedappsfilterdoutview: A more advanced view that combines the logic of the previous two. It gets an active version of information for all non-deleted applications.
- application_detailed_with_all_apps_filtered_del_app: Purpose: This is the most complex and important view in the schema. It joins details for all non-deleted applications with their active version information and their role mappings.
- This is the powerhouse view for the main user dashboard. The application backend can query this single view with a simple WHERE role_id = [current_user's_role_id] to get the exact list of applications that a specific user is authorized to see, complete with all necessary details to display the application cards. This greatly simplifies the application logic.

2.5 Deployment of the Unified Environments

2.5.1 Overview

The INDOT Application Suite is designed with a multi-tiered, service-oriented architecture that separates user-facing applications from backend services and specialized computational resources. This design, illustrated in Figure 2.8, ensures scalability, security, and maintainability.

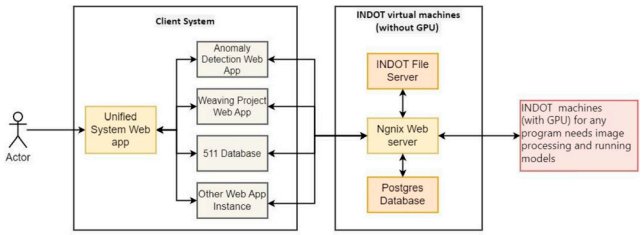


Figure 2.8 System Structure of the Unified System.

The system logically separates concerns into three distinct zones:

- The Unified Portal: The user-facing layer where all interaction begins. Users access a central web application to launch individual, containerized tools.
- The Backend Service Infrastructure: A set of non-GPU virtual machines that host the web applications, the reverse proxy, central database, and file storage.
- The Specialized Compute Infrastructure: A dedicated pool of GPU-equipped application servers for handling computationally intensive model execution and data processing tasks.

This decoupled design ensures that user-facing web services remain responsive and stable, even when backend processes are under heavy load.

2.5.2 Virtual Machines

The backend infrastructure runs on a set of virtual machines, with responsibilities logically divided.

- Web and Data Services (Non-GPU VMs)
  - Nginx Web Server: This high-performance server acts as the "front door" or reverse proxy for all backend services. It manages all incoming network traffic and performs several critical functions:
  - Request Routing: Intelligently directs incoming requests to the correct application container by reading the URL and forwarding the traffic to the appropriate host port
  - Postgres Database: Hosted within the same VM environment, the PostgreSQL database is the system's single source of data. It utilizes a highly normalized schema to store all structured data efficiently and reliably.

2.5.3 Specialized Compute Services (Application Server with GPU)

For tasks that demand significant computational power, the architecture offloads work to dedicated servers equipped with GPUs. This critical separation ensures that intensive background processing does not degrade the performance or responsiveness of primary web applications.

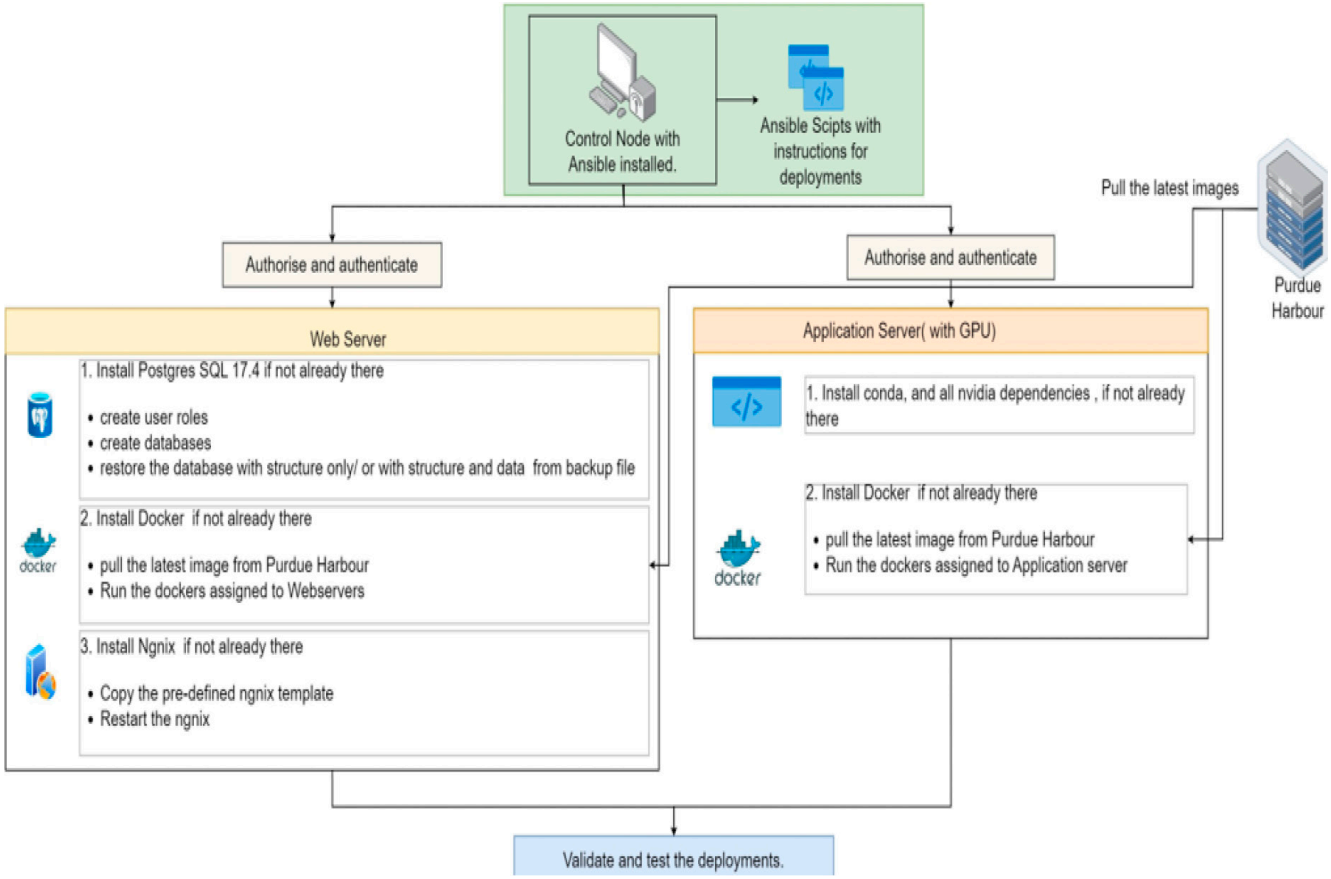


Figure 2.9 The Deployment Process.

2.5.4 Deployment and Automation Strategy (CI/CD)

The entire deployment process shown in Figure 2.9 is automated to ensure consistency, reliability, and security.

- Containerization with Docker: Every application is packaged into a Docker container. This creates a standardized, portable unit that includes the application's code, libraries, and dependencies, ensuring it runs identically in any environment.
- Purdue Harbor (Container Registry): Finished Docker images are pushed to Purdue Harbor, a private and secure container registry. This acts as a centralized, version-controlled repository for all application images, ensuring that deployments only use tested and approved versions.
- Automation with Ansible: The entire server setup and application deployment are managed by Ansible. This "Infrastructure as Code" approach uses a series of roles to automate every step:
  - Server Preparation: Installs base packages and configures user accounts.
  - Database Setup: Installs PostgreSQL and runs SQL scripts to create the necessary schemas, roles, and tables.
  - Docker Deployment: Installs Docker, securely authenticates with Purdue Harbor, and pulls the latest application images to run as containers. It uses pre-defined port mappings to expose each container on a unique port on the host server.
  - Web-server Configuration: Installs and configures Nginx as a reverse proxy, setting up the rules to route traffic to the correct application containers.

To maintain system stability, there is a clear quality gate for introducing new tools. A new application is first deployed to the environment using the same Ansible automation. Only after it has been thoroughly tested and approved is it formally registered in the Unified System.

2.6 Usage Guidelines

To ensure smooth and effective management of the application suite, administrators should adhere to the following guidelines:

- Adding a New Application: Before adding a new application to the system, ensure that the application has been successfully deployed and is fully operational. Additionally, confirm that a functional endpoint is available and accessible.
  - In this context, a functional endpoint is the final, publicly accessible URL for a newly deployed application. Think of it as the application's unique web address. After deploying an application, one would use a web server like Nginx to act as a reverse proxy. Nginx makes the application available at a clean, stable URL (e.g., https://your-new-app.com instead of something like http://ip_address:8080). This final URL created by Nginx is the endpoint. It is the "point" where the main platform communicates with your new application. You must provide this URL to the unified system app so it can successfully add and display the application.
- Deactivating an application: When deactivating an application, please note that the application will remain in the system in its current state. Deactivation only turns off the reference to the application, but the application and its data will remain intact. The app will stay in the system and can be manually reactivated.
- Deleting an application: Deleting an application removes its reference from the system. However, the user should also run the uninstall scripts provided by the developers to remove the application completely and its associated resources. Deletion only removes the reference; it does not fully uninstall the application.

2.6.1 Note on INDOT's Implementation Role

For the final production environment, the following critical security functions are the responsibility of INDOT's technical teams, replacing the provisional systems used during development:

- Login Implementation: INDOT is responsible for implementing the final login and authentication mechanism via its enterprise SSO system.
- Session Management: The current development system uses a temporary session management approach where user details (username, user_id, role_name, isSuperUser) are stored in browser cookies upon login. These cookies have a limited lifespan (e.g., four hours) and are cleared upon logout. For the production environment, **INDOT must implement a robust, enterprise-grade session management system to ensure that unauthorized users cannot bypass the unified portal and access individual application URLs directly. This is essential for enforcing a single point of entry and maintaining platform security.**

3. ANOMALY DETECTION

3.1 Introduction

INDOT operates approximately 600 digital cameras along highways across the state to monitor traffic conditions. Currently, human operators manually observe these video feeds to identify traffic anomalies, a process that is labor-intensive and inefficient for real-time surveillance. This research project aimed to develop an automated, real-time anomaly detection system for deployment within INDOT's infrastructure, followed by preliminary results from the 2023–2024 JTRP SPR-4855 anomaly detection project from Christopher et al. (2024).

This effort was a collaboration among the TASI at Purdue University, INDOT's Traffic Management Center, and the Traffic Engineering Division. The objective was to build a system capable of analyzing INDOT CCTV feeds to monitor traffic flow and detect anomalies based on statistical patterns and learned features.

The research team developed complete system architecture, including both hardware and software components. Leveraging the existing INDOT CCTV network, the team established a database to store traffic data extracted from video streams and built a web-based interface to display detected anomalies. The specific focus of the JTRP SPR-4930 project was the development and deployment of a high-performance automated anomaly detection system within INDOT. The research paper for this project has been accepted for presentation at the IEEE International Conference on Intelligent Transportation Systems (ITSC) 2025 (Qiu et al., 2025).

3.2 Dataset Preparation

A lane-wise traffic anomaly data collection and ground truth labeling pipeline was adopted, as shown in Figure 3.1. Lane-wise traffic data (e.g., vehicle count, occupancy, and truck percentage) is extracted from highway surveillance videos.

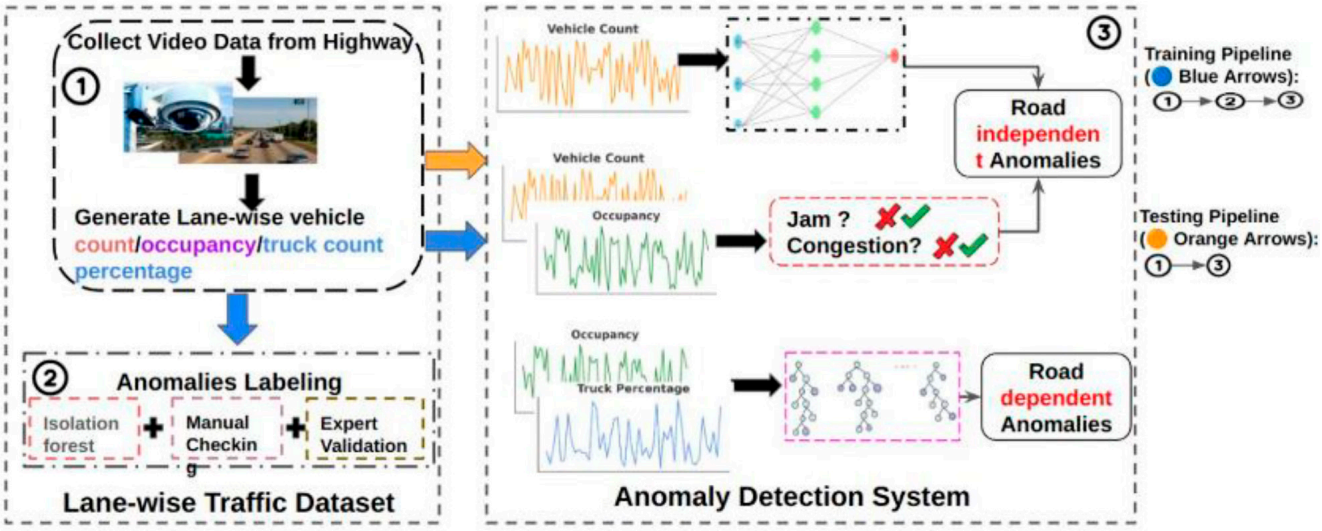


Figure 3.1 Overview of the Proposed Lane-Wise Anomaly Detection Framework.

Anomalies are identified through the use of Isolation Forest, manual checking, and expert validation. The system detects road-independent anomalies using deep learning and rule-based methods, and road-dependent anomalies using machine learning models applied to occupancy and truck percentage data. The anomaly labeling is only implemented during the training stage.

3.2.1 Data Collection

A total of 8,746 videos, each 15 min in length, were collected from five fixed cameras installed along highways in Indiana over a six-month period. The details of the cameras are as follows and also shown in Table 3.1:

- ID: 15, I-465/15.0 W 21ST ST, Indianapolis, IN
- ID: 18, I-465/18.0 W 46TH ST, Indianapolis, IN
- ID: 216, I-69/217.6 S OF SR 38, Indianapolis, IN
- ID: 461, I-65/254.0 73RD AVE, Indianapolis, IN
- D: 501, I-94/17.9 W OF SR 249, Indianapolis, IN

Leveraging the developed framework (Qiu et al., 2024) from the past JTRP project (SPR-4855 from Christopher et al. [2024]), two optimal regions, along with the corresponding lane centers within each region, were identified for vehicle counting. Vehicle detection was performed using YOLOv5x-CBAM (Qiu et al., 2022) and Deep SORT (Wojke et al., 2017) was employed for vehicle tracking. Only "car" and "truck" are detected and tracked. Figure 3.2 shows five road views with the learned optimal regions, lane centers, and lane identities.

TABLE 3.1
An Overview of Data From Five Cameras.

Camera ID	Number of Lanes	Number of 15-min Video Samples	Total Number of Lanes
15	11	2.368	26,048
18	10	2.291	22,910
216	6	1,268	7,608
461	6	1,701	10,206
501	6	1,118	6,708

Assume that there are L lanes learned in our observation regions (light blue rectangles in Figure 3.2), each indexed by $i \in \{1, 2, \ldots, L\}$ or $i \in \{-1, -2, \ldots, -L\}$ based on traffic directions.

For each lane i, we introduce the following notation:

- c_i: Total number of vehicles detected in lane i during each observation period T (T = 30 seconds).
- t_i: Total number of trucks detected in lane i during the same period.
- F_{ri}: Vehicle flow rate of lane i, defined as:

$$F_{r_i} = \frac{c_i \cdot 3600}{T}$$

(vehicles per hour).

- $TruckPerc_i$: Truck percentage in lane i, computed as:

$$TruckPerc_i = \frac{t_i}{c_i} \times 100\%$$

The truck percentage is introduced to quantify the proportion of trucks relative to the total vehicle count in that lane. Incorporating this information provides a richer context for distinguishing between typical fluctuations and anomalies driven by the influence of heavy vehicles.

- Occ_{pi}: The occupancy contribution of vehicle i is given by h_i/H_{ROI} where h_i represents the height of the bounding box for the i-th detected vehicle within the ROI, and H_{ROI} represents the height of the region of interest (ROI), which is the vertical size of the blue rectangle shown in Figure 3.2.

The average lane occupancy during the data collection interval is estimated based on the size of the ROI and the heights of

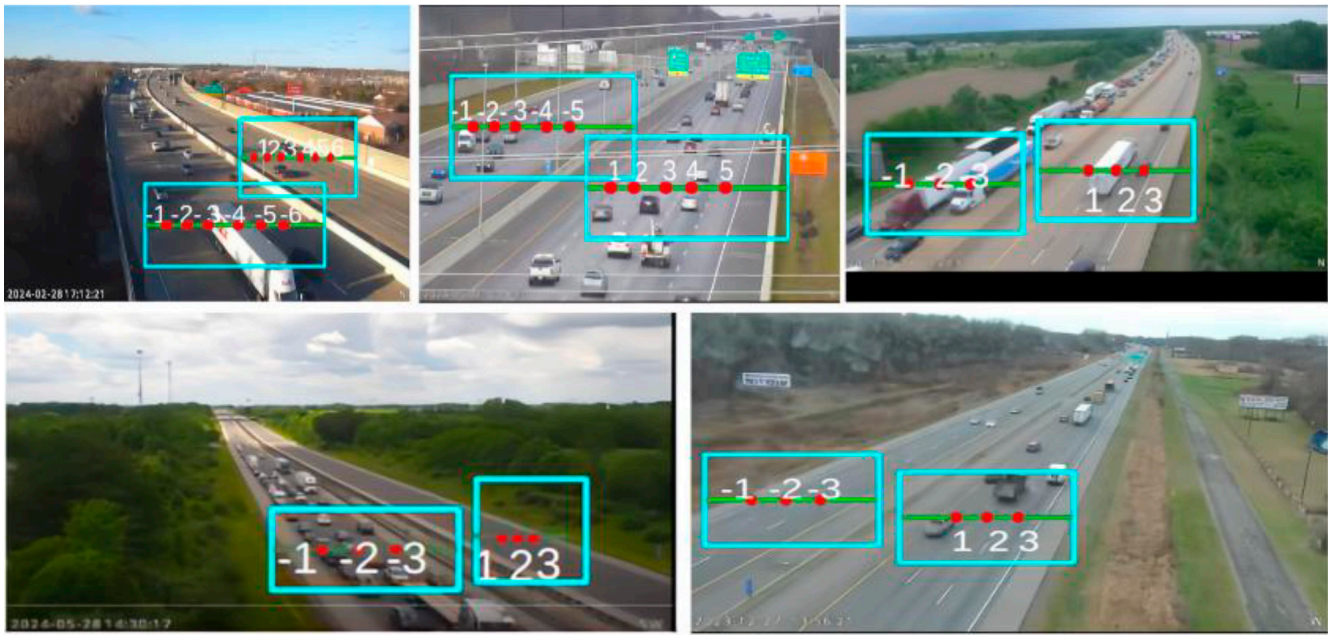


Figure 3.2 Five Road Views With Detection Regions, Lane Centers, and Counting Lines. Light Blue Rectangles Show the Learned Regions of Interest (ROIs), Green Lines Mark Optimal Counting Lines, and Red Dots Indicate Lane Centers. White Lane IDs Use Signs to Indicate Direction: Negative for Towards the Camera, Positive for Away From the Camera.

vehicles detected passing through this ROI. Each detected vehicle contributes to the occupancy proportionally to its height relative to the ROI height. Let *W* be the number of vehicles detected during the interval *T,* and *M* be the number of frames processed within *T*. The average occupancy O_T over the interval T is defined as the mean of these contributions per frame:

$$\overline{o_T} = \frac{1}{M}\sum_{i=1}^{N} Occ_{p_i}$$

Throughout the remainder of this report, occupancy refers specifically to the average lane occupancy. The upper row of Figure 3.3 shows lane-wise vehicle counts, occupancies, and truck percentages from a single 15-min video. The bottom row presents the distribution of these features across all data collected from a single camera over 24 hours, aggregated across all data collection days.

3.2.2 Anomaly Labeling and Definition

Due to the lack of a universally accepted definition of traffic anomalies, we adopt a practical approach rooted in traffic management principles to construct a verified anomaly dataset.

- First, we apply the Isolation Forest algorithm (Liu et al., 2012) to identify a broad set of outlier cases from lane-wise vehicle count sequences. The **Isolation Forest algorithm** is an unsupervised learning model that isolates anomalies by using a tree-based approach. It works on the principle that anomalies are "few and different" and therefore easier to isolate than normal data points. Each sequence $x_i = [c_{(i,1)}, c_{(i,2)}, \ldots, c_{(i,n)}]$ represents 30 vehicle counts collected over a 15-min video, with each xi corresponding to a 30-s interval. To ensure broad coverage of potential anomalies, we use a relatively high **contamination rate** of 0.3 for the hyperparameter of the Isolation Forest algorithm. The **contamination rate** is a hyperparameter that specifies the expected proportion of outliers in the dataset. A higher value, such as 0.3, instructs the algorithm to anticipate a greater number of anomalies, thereby ensuring that we do not miss any potential outliers.
- Second, normal cases are manually removed, and the remaining samples are reviewed and validated by traffic engineers. The verified anomalies are then used to train the Deep Learning (DL)-based anomaly detection model. The model assigns an anomaly score s_i to each sequence:

$$s_i = F(x_i)$$

Where *F(.)* is the Isolation Forest model function, the final decision for anomalies is based on the following rule:

$$\text{Anomaly}(x_i) = \begin{cases} 1, & \text{if } s_i < 0, \\ 0, & \text{otherwise.} \end{cases}$$

Sequences flagged as anomalous were manually reviewed to eliminate false positives. Following this refinement, we consulted traffic experts for validation. Based on their input, we identified and categorized three primary types of traffic anomalies and one type of sensor anomaly:

1. Lane blockage and recovery, typically caused by vehicle malfunctions and tow truck intervention.
2. Foreign object intrusions, such as a tire entering the roadway and disrupting specific lanes.
3. Sustained congestion, often due to truck accumulation and prolonged slow-moving traffic.
4. Sensor anomaly: Camera angle changes or shifts.

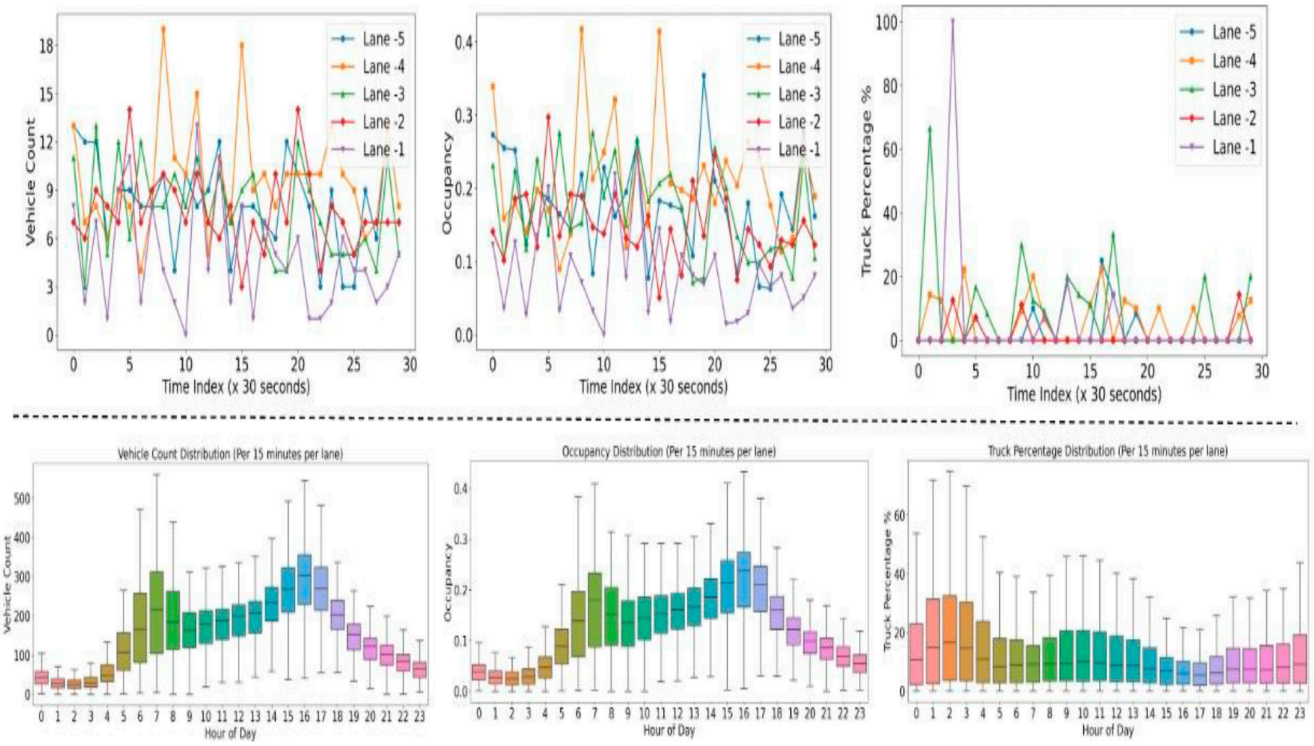


Figure 3.3 Top Row: Lane-Wise Vehicle Count, Occupancy, and Truck Percentage From a Single 15-min Video. Bottom row: The Distribution of These Features Across All Data Collected From a Single Camera Over 24 hr, Aggregated Across All Data Collection Days.

From the full video dataset, 46 anomalous clips were retained after expert validation. This yielded a final dataset of 73,139 normal lane-wise samples and 341 labeled anomaly samples. Figure 3.4 illustrates representative examples of these real-world anomaly cases.

3.3 Anomaly Detection Method

The entire framework for the proposed fusion method is illustrated in the Anomaly Detection block of Figure 3.1. The details of the data input information are presented in Table 3.2. The data overview for each module's input is shown in Figure 3.5).

3.3.1 Deep Learning-Based Anomaly Detection

Figure 3.6 illustrates our deep learning (DL)-based anomaly detection framework, which identifies deviations in traffic patterns across frequency bands. Lane-wise vehicle counting over 15-min windows is converted into Continuous Wavelet Transform (CWT) spectrograms (Aguiar-Conraria & Soares, 2014), then processed by a Vector Quantized Variational Autoencoder (VQ-VAE A; van den Oord et al., 2017). Anomalies are detected via reconstruction errors, using adaptive thresholds based on time-of-day groups (e.g., morning, noon). We assume minimal information loss from the CWT, that anomalies yield higher reconstruction errors, and that anomalies are sparse in training data.

1) *Wavelet Transform on Traffic Data*: CWT analyzes localized variations in traffic signals across multiple frequencies. Applied to lane-specific counts, it captures both temporal and spatial patterns. Scaling adjusts frequency sensitivity—stretching reveals long-term trends, such as congestion, while shrinking detects abrupt events, like lane blockages—making CWT effective for detecting diverse traffic anomalies. Mathematically, the CWT of a signal x(t) is defined as:

$$CWT_x(a,b)=\frac{1}{\sqrt{a}}\mathrm{CWT}_x(a,b)=\frac{1}{\sqrt{a}}\int_{-\infty}^{\infty}x(t)\Psi^*\left(\frac{t-b}{a}\right)dt$$

where a is the *scale parameter* (inversely proportional to frequency), b is the *translation parameter* (shifting the wavelet in time), and $\psi(t)$ is the *mother wavelet*. The asterisk * denotes complex conjugation. The scale parameter a spans 21 integer values from 4 to 24, while the time shift parameter b corresponds to the sampled time points {0, 30, 60, 90, ..., 870} seconds.

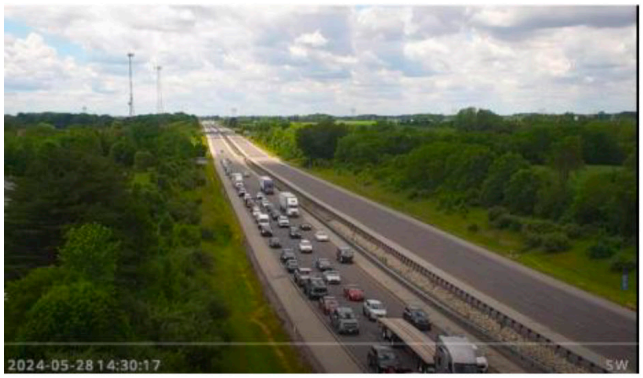
To allow different wavelet spectrograms to be directly compared in scale-independent visualizations, we define the normalized CWT as follows:

$$\mathrm{CW\tilde{T}x}(a,b)=\frac{|CWT_x(a,b)|}{\max\limits_{a',b'}|CWT_x(a',b')|}$$

where $\mathrm{CWT}_x(a, b)$ is the original complex-valued CWT at scale a and time b, |.| denotes the magnitude (absolute value), the denominator is the *global maximum* over all scales a' and

(a) Foreign object (a tire) on the highway for 7 min (marked with a red circle)

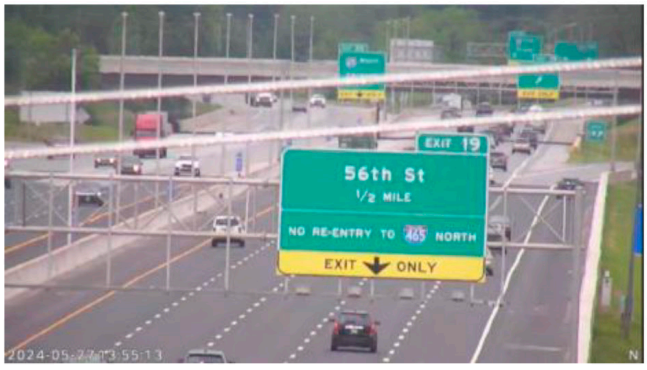
(a) Truck malfunction and tow truck intervention affecting multiple lanes (marked with red circles)



(b) Vehicles are only on the left roadway with congestion

(c) Traffic jam and congestion in the left lanes

(d) Camera angle shift



(e) The camera zoomed in too much

Figure 3.4 Examples of Observed Traffic Anomalies (a Through d) and Sensor Anomalies (e and f).

TABLE 3.2
An Overview of the Input Data Structure for Different Modules.

Method	Input Size	Data Type	Row	Column
DNN-based	30 * 1	Vehicle count	Time index	Lane number
Rule-based	30 * 1	Occupancy and flow rate	Time Index	Lane Number
ML-based	60 * N	Occupancy and truck percentage	Time index (stacked)	Lane Number

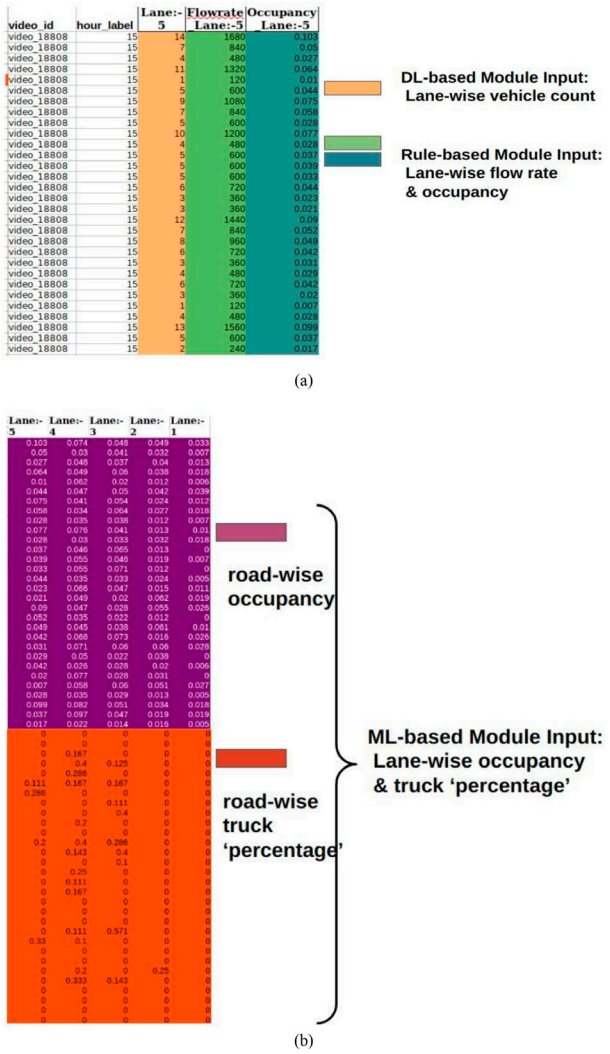


Figure 3.5 Input Data Overview for Each Module. (a): Lane-Wise Input Data; (b): Road-Wise Input Data.

translations *b'* across all samples, and $CW\tilde{T}_x(a, b)$ is the *normalized* wavelet coefficient, ranging from 0 to 1.

2) *VQ-VAE (van den Oord, et al., 2017):* It is a generative model that combines the power of discrete latent representations with neural networks for efficient data reconstruction. Unlike standard VAEs that use continuous latent spaces, VQ-VAE maps input to a finite set of embeddings (codebook vectors). In our framework, the normalized CWT spectrogram $CW\tilde{T}_x(a, b)$ of each lane-wise vehicle count sequence is used as the input to the VQ-VAE. Specifically, the encoder $f_\theta(CW\tilde{T}_x)$ maps the normalized CWT input to a latent vector z_e, which is quantized to the nearest codebook vector e_k from $\{e_1, e_2, ..., e_k\}$. The quantized latent vector $z_q = e_k$ is then passed to the decoder $g_\phi(z_q)$ for reconstruction. The loss function for VQ-VAE consists of three terms:

$$L = \|CW\tilde{T}_x - g_\varphi(z\_q)\|^2 + \|sg[z_e] - e_k\|^2 + \beta\|z_e - sg[e_k]\|^2$$

where sg[.] is the stop-gradient operator, and β controls the commitment cost to encourage the encoder outputs to stay close to the codebook vectors.

3) *Anomaly Detection with Adaptive Thresholds*: Anomalies are detected during the testing phase using dynamic thresholds derived from the reconstruction error distribution of the training data. Reconstruction error distributions from the final training epoch are grouped by time-of-day intervals: Night (0–5), Morning (6–11), Afternoon (12–17), and Evening (18–23). For each group, the 95th percentile of the corresponding error distribution is used as the anomaly detection threshold, as illustrated by the vertical dashed lines in Figure 3.7.

Let E_i denote the reconstruction error for a test sample i, and let T_g represent the threshold for the time-of-day group $g \in \{\text{Night, Morning, Afternoon, Evening}\}$. The threshold T_g is defined as:

$$T_g = Percentile95\left(\left\{E_j \mid j \in Training\ data\ in\ group\ g\right\}\right)$$

where E_j are the reconstruction errors of training samples belonging to group g. A test sample i from group g is classified as anomalous i:

$$E_i > T_g$$

3.3.2 Rule-Based Anomaly Detection

While the DL-based model effectively detects various anomalies, it may miss gradual congestion patterns that do not cause large reconstruction errors. To address this, we directly monitor F_{ri} and occupancy Occ_{pi} for each lane i, defining the traffic status $status_i$ as:

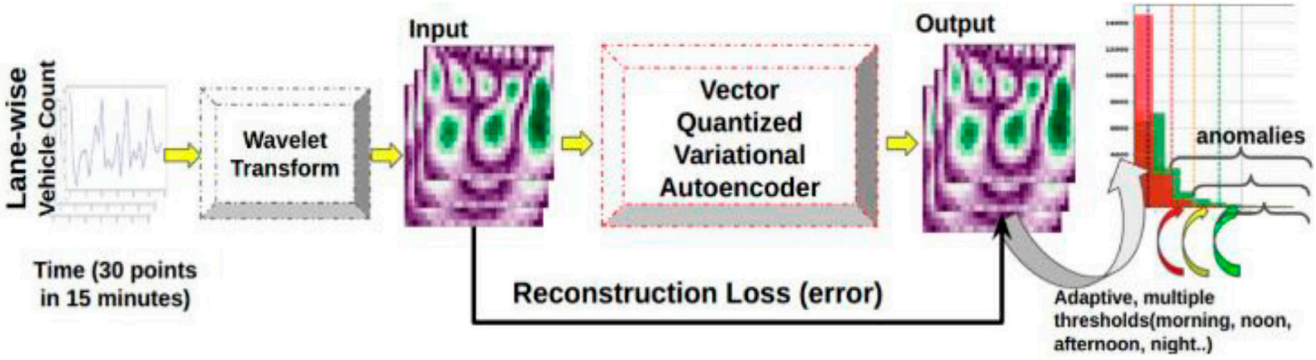


Figure 3.6 Deep Learning-Based Anomaly Detection Method.

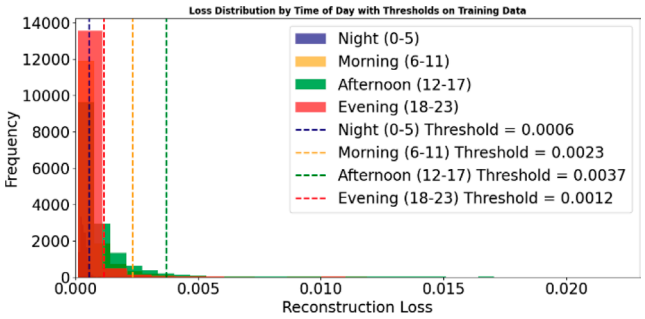


Figure 3.7 Different Thresholds are Computed by Applying the 95th Percentile to the Reconstruction Loss Distribution Within Each Time-of-Day Group (e.g., Night, Morning, Afternoon, Evening).

$$\text{status}_i = \begin{cases} \text{Jam}, & \text{if } F_{r_i} < 600 \text{ and } \text{Occ}_{p_i} > 0.6, \\ \text{Slow}, & \text{if } 600 < F_{r_i} < 900 \ \& \ 0.4 < \text{Occ}_{p_i} < 0.6, \\ \text{Normal}, & \text{otherwise}. \end{cases}$$

This rule-based framework (as shown in Figure 3.8) enables the detection of jam and congestion anomalies by identifying lane segments exhibiting low vehicle throughput but high physical occupancy.

3.3.3 Machine Learning-Based Anomaly Detection

The machine learning (ML)-based module (framework shown in Figure 3.9) is critical for detecting road-dependent anomalies that are not effectively captured by DL-based or rule-based methods. To identify complex patterns—such as truck-induced slowdowns, where heavy vehicles occupy more space and accelerate slowly, contributing to stop-and-go congestion—we incorporate both occupancy and truck percentage as input features. While occupancy alone provides partial insight, the truck percentage offers additional context to characterize road-specific traffic disruptions better.

For each video, road-dependent anomaly detection is performed at the ROI level of each traffic direction. Let the occupancy data be denoted as $O \in R^{N*L}$, where N is the number of time intervals and L is the number of lanes. We vertically stack the occupancy data and truck percentage data $T \in R^{N*L}$ to construct the feature matrix:

$$X = [O ; T] \in R^{2N \times L}$$

where [.;.] denotes vertical concatenation. As an example, for Camera 15 on the left road segment with five lanes, the occupancy matrix and truck percentage matrix are each of size 30 × 5 (30 time intervals across 5 lanes). After vertical concatenation, the resulting feature matrix has a size of 60 × 5. This feature matrix is used for road-level anomaly detection by inputting X into an Isolation Forest model F. The contamination rate is set to 0.1. The model outputs a binary label for each sample: a value of -1 indicates an anomaly, while 1 denotes a normal case, effectively separating outliers from typical traffic patterns.

3.4 Anomaly Detection Experiments

3.4.1 Experimental Settings

1) *Data*: 80% of the anomaly-free lane-wise count sequences are used to train the VQ-VAE model, while the remaining 20% serve as validation data. For testing, 341 anomalous samples from 43 videos are combined with 341 normal samples randomly selected from the validation set. This test set is used for evaluating all models (DL-based, ML-based, and Rule-based).
2) *Performance Evaluation Metric*: Anomaly detection performance is commonly evaluated using several key metrics derived from the confusion matrix.
   – Accuracy (Acc) measures the proportion of correctly classified instances:

$$\text{Accuracy} = (TP+TN) / (TP+TN+ FP+FN)$$

where TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

- Precision (Pre) quantifies the proportion of true anomalies among all detected anomalies:

$$\text{Precision} = TP / (FP+TP)$$

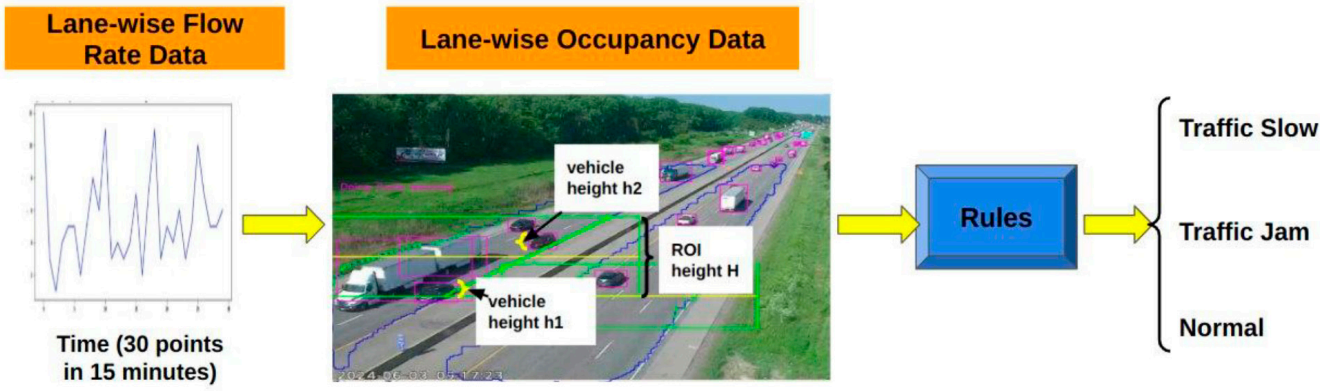


Figure 3.8 Rule-Based Anomaly Detection Method.

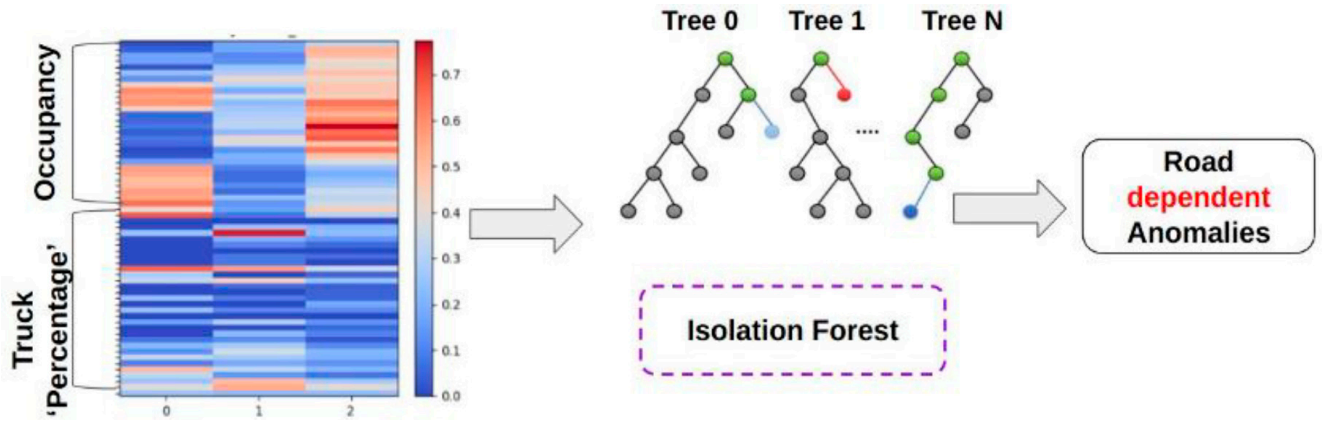


Figure 3.9 ML-Based Anomaly Detection Framework.

- Recall (Re), or sensitivity, indicates the proportion of correctly detected anomalies among all actual anomalies

$$\text{Recall} = TP / (FN+TP)$$

- The F1-score (F1) provides a harmonic mean of precision and recall:

$$F1 = (2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

- Additionally, the False Positive Rate (FPR) and False Negative Rate (FNR) are given by FPR = FP/ FP+TN and FNR = FN/ FN + TP, respectively, representing the rates of incorrect anomaly detection and missed anomalies.

3) *Implementation Details*: The VQ-VAE consists of an encoder with two Conv2d layers (1 * 64 and 64 * 32, kernel size 4, stride 2, padding 1), each followed by BatchNorm and ReLU. A 32 * 32 pre-quantization Conv2d feeds into a codebook of 64 embeddings (dimension 32). Post-quantization, a 32 * 8 Conv2d layer prepares features for the decoder, which mirrors the encoder with two ConvTranspose2d layers (B * 64 and 64 * 1) and a final Tanh activation. The commitment loss weight β is set to 0.25. The model is trained using Stochastic Gradient Descent (SGD) with a learning rate of $1 * 10^{-3}$, weight decay of $1 * 10^{-5}$, batch size 128, and up to 150 epochs. Early stopping is applied by monitoring validation loss to prevent overfitting. The Isolation Forest model used in this work is implemented using the Isolation Forest class from the ensemble module in scikit-learn.

3.4.2 Results

1) *Results compared with other methods*: Table 3.3 presents a comparison of precision, recall, and F1-score across various anomaly detection methods applied to our dataset. For each baseline method, the input consists of our lane-wise vehicle count data, with data from all lanes in each video concatenated into a single sequence. The original configurations of these methods were preserved as specified in their respective implementations. Our method outperforms all baselines (e.g., Isolation Forest [Liu et al., 2012], OWAM [Choudhary & Hassani, 2023], TimeVQVAE-AD [Lee et al., 2024], TranAD [Tuli et al. 2022]), achieving the highest F1-score of 0.8841, which indicates superior precision and recall.

TABLE 3.3
Performance Comparison With Other Methods.

Approach	Method	Precision	Recall	F1
Isolation Forest	Isolation Forest	0.177	0.6047	0.2738
Density Distribution	OWAM	0.268	0.2558	0.2558
Generative Modeling	TimeVQVAE-AD	0.45	0.37	0.4061
Reconstruction and Prediction	TranAD	0.32	0.49	0.3872
Reconstruction	**TASI method**	**0.8077**	**0.9767**	**0.8841**

Note: Best performance is highlighted in bold.

TABLE 3.4
Performance With Different DL-Based Modules.

Method	Acc	Pre	Re	F1	FPR	FNR
DL-based	0.5279	0.7368	0.9767	0.84	0.045	0.023
DL-based & Rule-based	0.5293	0.7414	1.0	0.851	0.045	0
DL-based & Rule-based & ML-based	0.9787	0.8431	1.0	0.9149	0.024	0

Note: Best performance is highlighted in bold.

2) *Ablation Study*: Table 3.4 highlights the effectiveness of combining detection modules. The DL-based module alone achieves an F1-score of 0.84, increasing to 0.851 when integrated with the Rule-based module. The full framework, which combines DL-based, Rule-based, and ML-based modules, achieves the best results with an accuracy of 0.9787 and an F1-score of 0.9149. Each module contributes uniquely: DL captures temporal anomalies, Rule-based detects abrupt disruptions via occupancy thresholds, and ML targets road-dependent patterns using occupancy and truck percentage. Their integration ensures robust and precise anomaly detection across varied traffic scenarios.

We also evaluate the impact of wavelet transformations in the ML-based framework. As shown in Figure 3.10, applying CWT significantly boosts detection performance. Figure 3.11 further demonstrates that CWT-transformed inputs result in clearer separation between normal and anomalous data in the Uniform

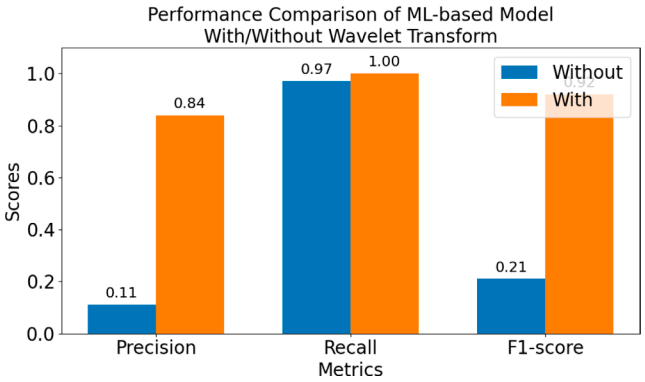


Figure 3.10 Performance Comparison of ML-Based Models With and Without Wavelet Transform.

Manifold Approximation and Projection (UMAP)-projected latent space, whereas models trained on raw time series inputs produce more isotropic and entangled representations, making anomaly detection more challenging.

In addition, the wavelet analysis in Figure 3.12 suggests that structural differences in wavelet patterns could provide valuable cues for distinguishing normal and anomalous traffic behaviors. In Figure 3.12, the top part shows normal cases, while the bottom row illustrates anomaly cases. For each case, the upper row displays vehicle counts over a 15-minute period, and the lower panel presents the corresponding wavelet transform (normalized absolute values). Normal traffic exhibits a dominant periodic pattern with a period of approximately 9 minutes. In contrast, anomalous traffic shows two to three prominent periodic patterns at approximately 2, 4, and 9 minutes (marked with red rectangles).

Normal samples typically exhibit a dominant periodic pattern (e.g., around 9 min), while anomalies display multiple distinct periodic components at shorter intervals (e.g., 2, 4, and 9 min). This observation indicates the potential to systematically link wavelet pattern structures with anomaly characterization, offering a new direction for more interpretable anomaly detection.

TABLE 3.5
Performance With Different Threshold Strategies in ML-Based Module.

Threshold Rule	Acc	Pre	Re	F1	FPR	FNR
90% (time dependent)	0.5117	0.6143	**1.0**	0.7611	0.08	0
95% (time dependent)	0.5279	0.7368	0.9767	0.84	0.045	0.023
99% (time dependent)	0.5425	**0.9512**	0.907	**0.9286**	**0.006**	0.093
99% (time independent)	**0.9202**	0.6226	0.7674	0.6872	0.06	0.23

Note: Best performance is highlighted in bold.

3) *Sensitivity Analysis:* We compare the performance of DL-based anomaly detection with different threshold setting strategies in Table 3.5. Thresholds are set at the 90%, 95%, and 99% percentiles of reconstruction loss from the last training epoch. In time-dependent strategies, thresholds are computed separately for different time-of-day groups (e.g., morning, afternoon, night), while the time-independent strategy uses a single threshold for all data. The 99% time-dependent threshold yields the best performance. In contrast, the time-independent threshold strategy yields a lower F1-score and a higher false negative rate, indicating that time-aware thresholds more effectively capture diurnal traffic variations and enhance anomaly detection.

3.5 System Architecture and Development

As part of INDOT's requirements, we developed a full-stack architecture for production use (Figure 3.13). The back-end system comprises a centralized PostgreSQL database (Figure 3.14), a scalable Docker cluster for running our anomaly detection and car count software, and a storage server for housing our anomaly models and unprocessed camera videos. Raw camera footage is recorded from the user-selected cameras and processed by our

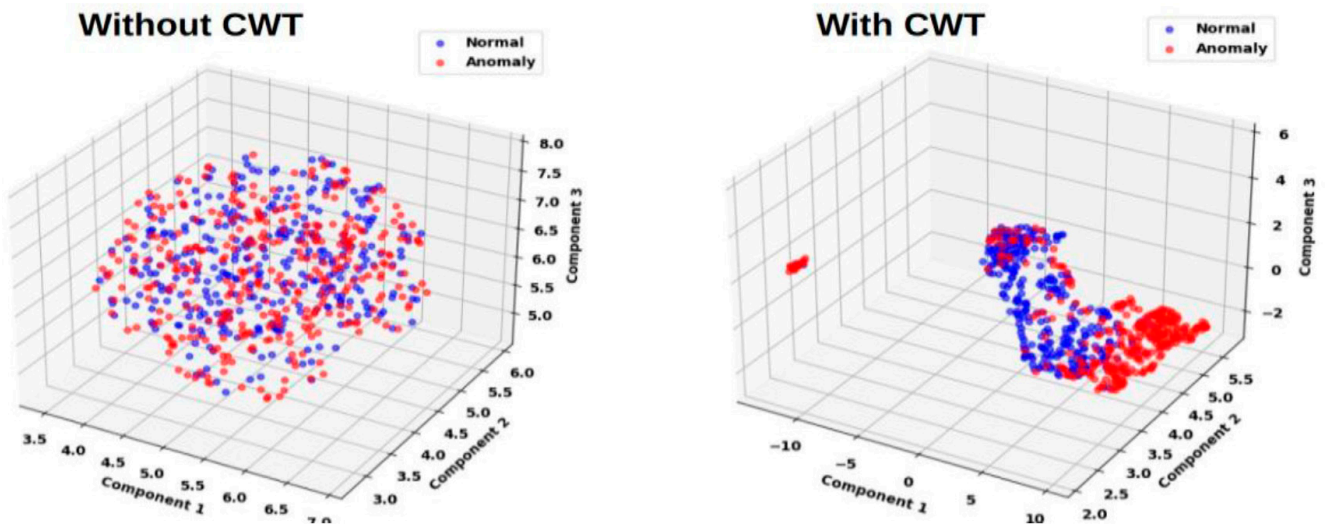


Figure 3.11 UMAP Visualization of 3D Latent Spaces Learned by VQ-VAE. Left: Without CWT; Right: With CWT. Blue Points Represent Normal Samples, and Red Points Represent Anomalies.

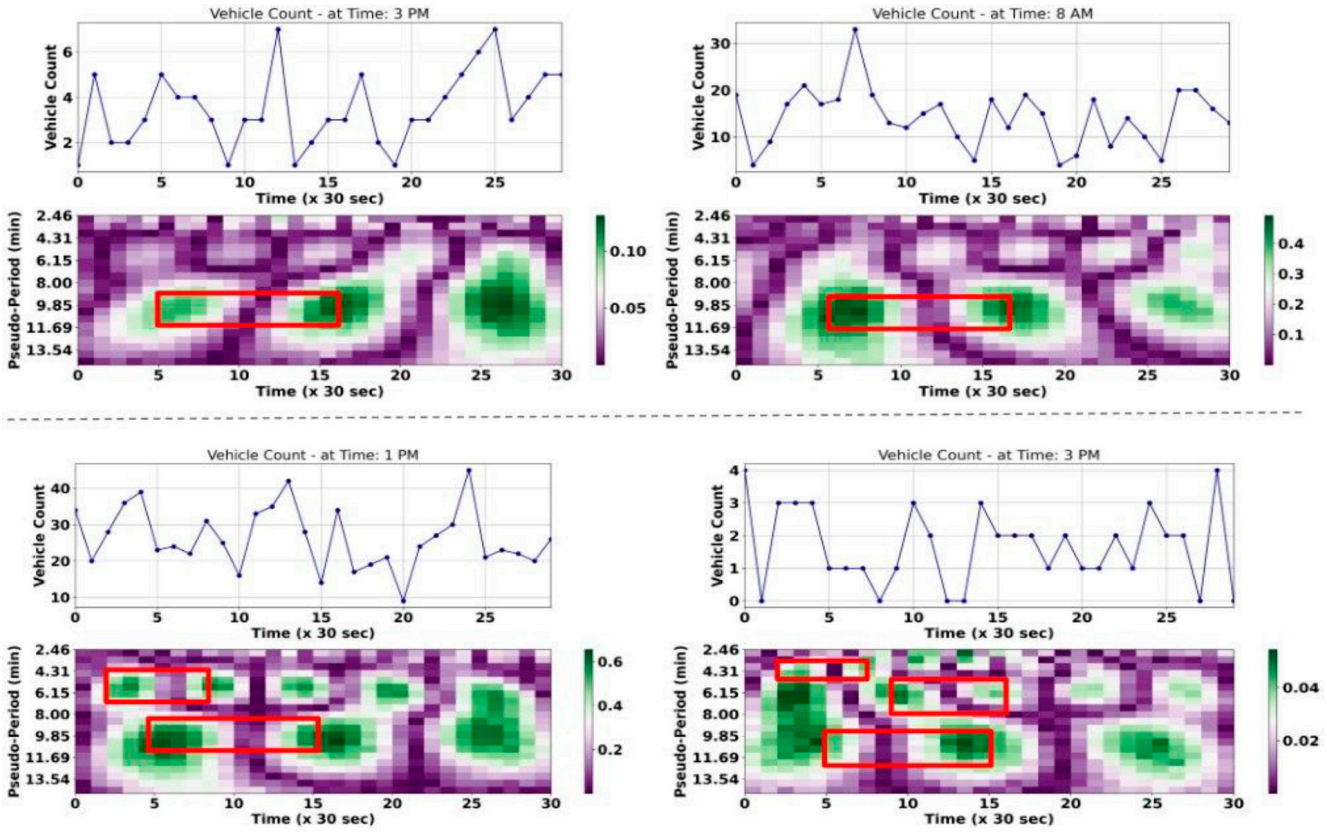


Figure 3.12 Wavelet Analysis of Traffic Data From a Single Lane.

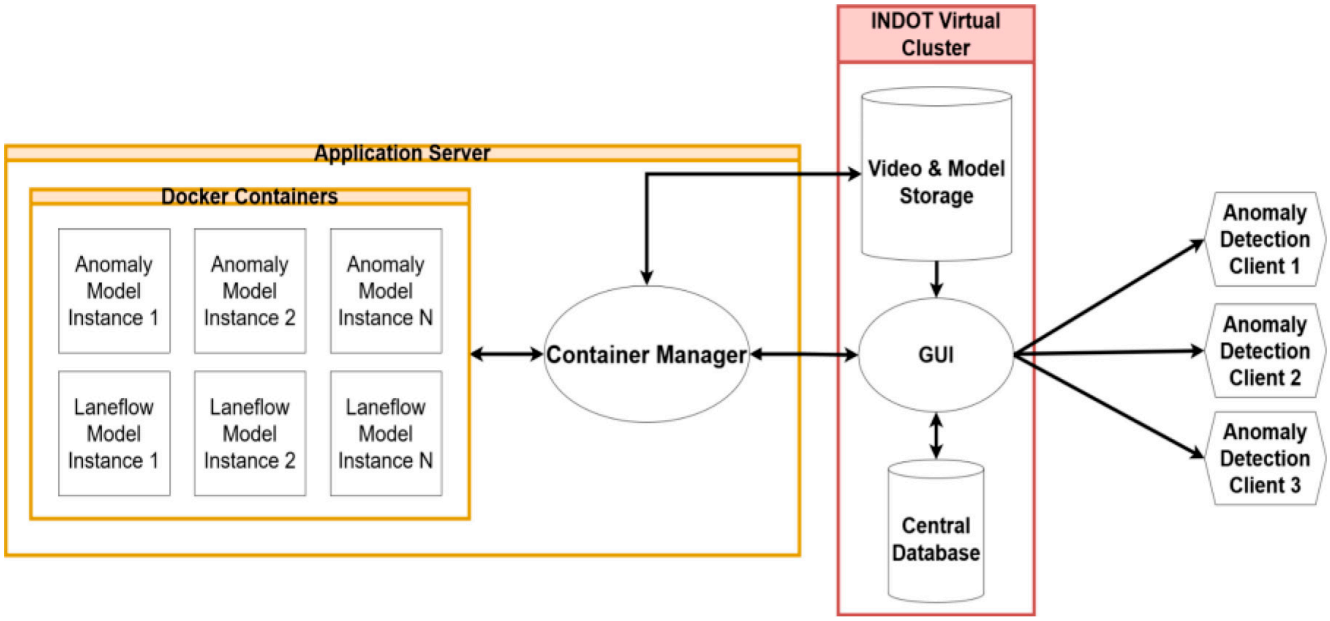


Figure 3.13 Integrated System Architecture Depicting an Application Server Running Container Software and Anomaly/Lane-Flow Code. In Addition to a Virtual Cluster Containing The Database, Web UI Code, and File Storage.

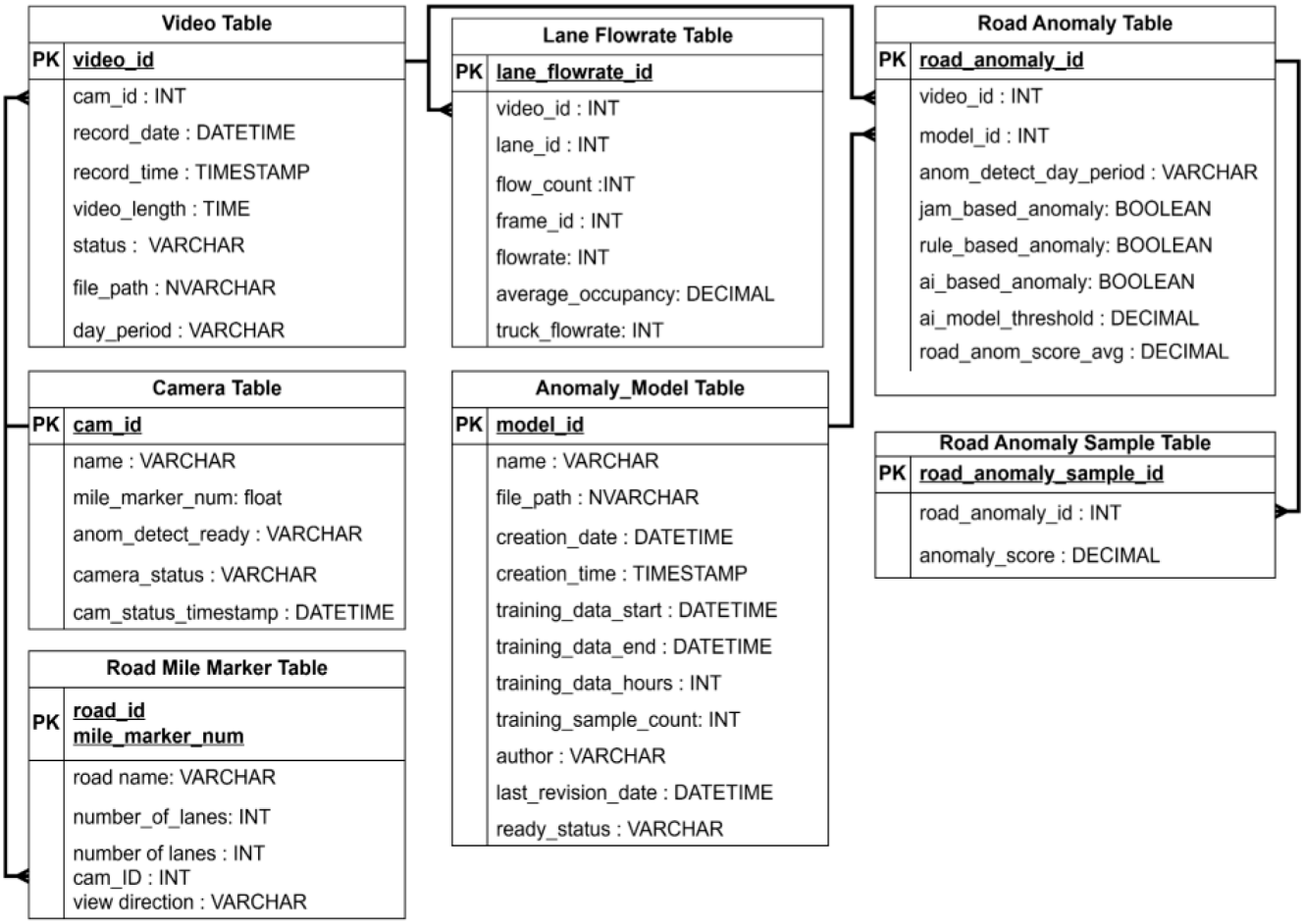


Figure 3.14 Entity Relationship Diagram of Anomaly PostgreSQL Database.

vehicle counting software. These vehicle counts, occupancies, and flow rates are then inserted into our centralized database. After processing a 15-min segment of video, the data are then sent to our anomaly model for analysis of potential anomalies. All analyzed data (anomalous or not) are inserted into the database for historical archive. Finally, anomalies are then displayed to the operator through the web-based Graphical User Interface (GUI) we developed.

All software (excluding the UI) dynamically scales to the number of cameras being processed in our system. We achieve this using a program developed specifically for this project, aptly named The Container Management program. Operators issue a command for a particular camera through the web-based UI, and data are sent to the Container Management program via API calls. All handling of the various backend programs developed in the paper is delegated to this software, enabling future scaling of the system across the entire INDOT camera network (limited only by hardware).

3.6 GUI Interface

We also developed a web-based platform (Figure 3.15), which allows operators to easily view anomalies generated by the traffic system. It consists of three main components: Generating Car Counts from Video, Detecting Historical Anomalies, and Detecting Real-Time Anomalies. This process segmentation enables the operator to quickly schedule data preprocessing for the anomaly model, view past anomalies, or view current anomalies generated in real-time.

Car count data generation is the most computationally expensive process, and due to hardware considerations, operators must manually initiate data generation. The component "Generate Car Count From Video" starts the car count program on a given camera for a selected time range. The "Detect Historical Anomalies" component (Figure 3.16a) is where already processed car count data of a selected time range are fed into the anomaly model for analysis. All predicted scores are recorded in the database for future reference, and the anomalous scores are displayed to the user. Finally, the "Detect Realtime Anomalies" component (Figure 3.16b) takes live video from the selected traffic camera, generates car count data, analyzes it for anomalies, and displays it to the user all in one step.

Anomalies are communicated to the user using a standard bar chart (Figure 3.17). Each bar represents either a single anomaly score or an average score based on the selected timescale. Each individual bar is color-coded, with the color being

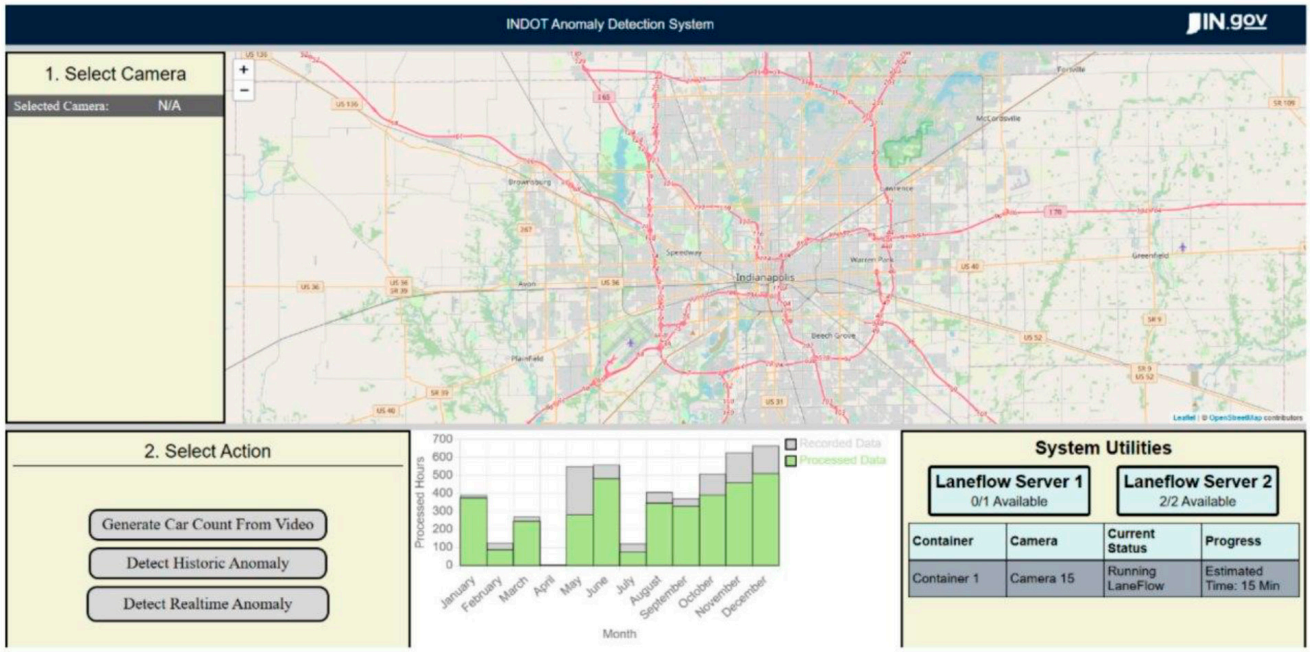
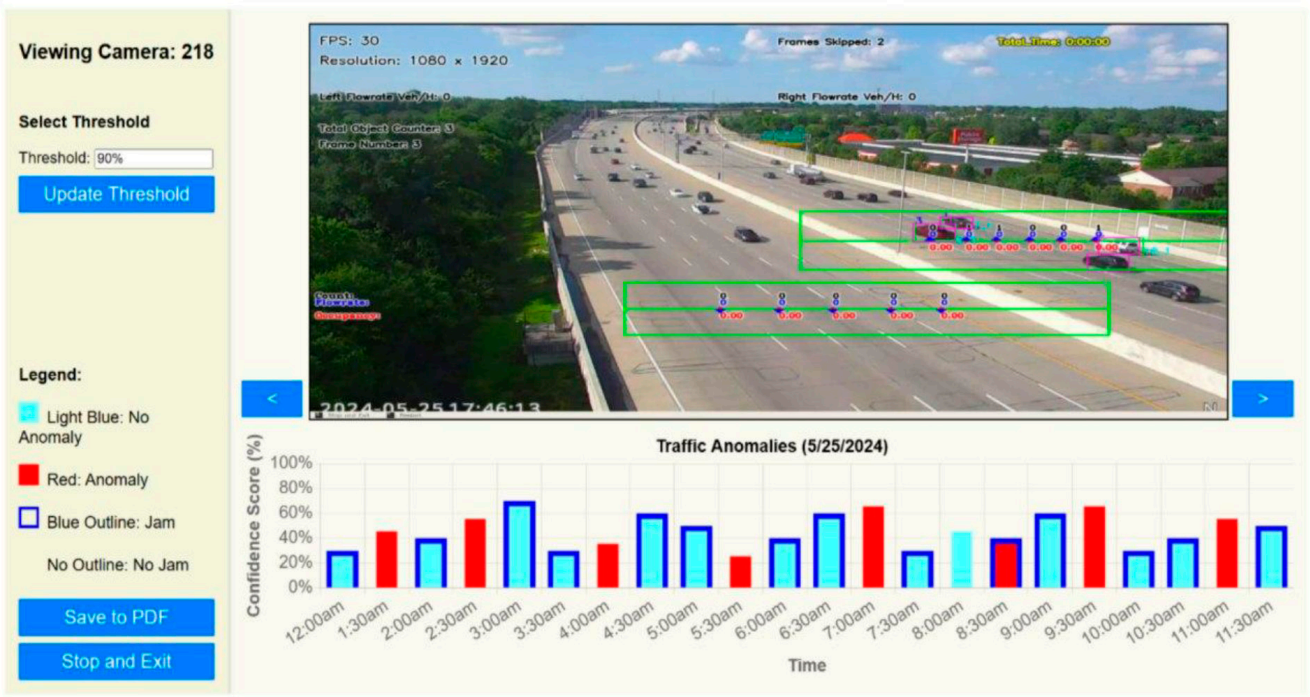


Figure 3.15 Web-Based Platform for Traffic Monitoring and Anomaly Visualization. The Interface Depicted Here Provides Access to Various Components of the Anomaly System, Allowing Users to View Statistics on Already Processed Data or Monitor Program System Utilization.



(a)

Figure 3.16 (a) User Interface for Visualizing Historical Anomaly Detection Results in a Per-Day, Month, or Year View (Day Shown Here). (*Continued*)

(b)

Figure 3.16 (b) Real-Time User Interface for Viewing Anomaly Detection Results as They Are Processed From the Live Camera Feed.

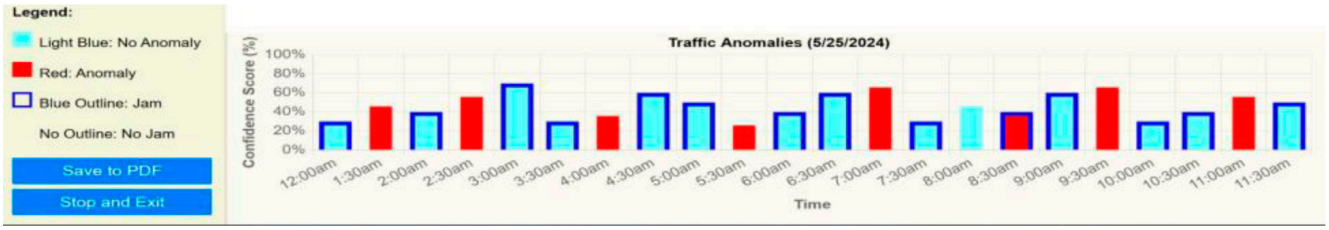


Figure 3.17 Anomaly Bar Chart Showing Traffic Anomalies. Each Bar Here Represents One 15-min Video and Is Color-Coded Based on Whether the Video Is Anomalous and Whether a Traffic Jam Occurs. The X-Axis Is the Timescale of Anomalies (12 hr Shown Here), and the Y-Axis is the Anomaly Score (Shown Here as Confidence Score). Bars Will Switch to Average Anomaly Scores When Operators Select Larger Timescales.

red for anomalous, light blue for normal, and/or a solid blue outline if a traffic jam was detected in that time period. The X-axis represents the timescale of the period selected by the user and can be displayed on a day, month, or year scale. Users can switch scales by clicking the bar to "zoom in" to a smaller scale (e.g., Month -> Day) or by clicking anywhere else in the chart to "zoom out" (E.g., Day -> Month).

3.7 Conclusion and Future Work

In this project, we designed a scalable, lane-wise highway traffic anomaly detection framework that operates exclusively on video-based data. By leveraging AI-driven vision techniques, our system extracts interpretable lane-specific traffic features, including vehicle count, occupancy, and the percentage of trucks. We introduced a novel dataset derived from real-world highway surveillance, capturing fine-grained traffic dynamics across multiple time periods and anomaly types. The framework integrates deep learning, rule-based logic, and machine learning methods to detect both road-independent and road-dependent anomalies. Experimental results demonstrate the robustness and effectiveness of our approach in accurately identifying traffic anomalies, supporting its practical applicability in intelligent transportation systems.

3.7.1 Limitations

This study is constrained by a geographically limited dataset, which may affect the model's generalizability to diverse traffic environments. While a universally accepted definition of traffic anomalies is lacking, new cases can be continuously collected and incorporated into model training to improve coverage. Additionally, the current system lacks active learning, continual updates, and structured expert feedback mechanisms, which

are essential for adapting to evolving traffic dynamics and offer promising directions for future enhancement.

3.7.2 Future Work

To improve robustness and scalability, future work will expand the dataset across diverse locations and road types. We plan to develop a more principled definition of anomalies using hybrid labeling strategies that combine domain knowledge, behavioral clustering, and statistical thresholds. A frequency band filtering approach will be introduced to emphasize temporal patterns relevant to anomalies. Additionally, human-in-the-loop components—such as expert review, active learning, and continual model updates—will support adaptive and interpretable anomaly detection. We also plan to integrate active learning or real-time adaptation mechanisms to further enhance the system's responsiveness.

4. IMPROVEMENT OF THE WEAVING ANALYSIS PROGRAM

The weaving program developed in project SPR-4738 from Chien et al. (2024) successfully captured the traffic weaving pattern in the specified weaving area. The software developed in SPR-4738 was installed on a Linux-based computer at INDOT. Based on user feedback, INDOT made the following requests for software improvements in this project, primarily to enhance user-friendliness and ease of access:

1. To enable convenient user access, the user interface should be web-based
2. The web-based user interface shall be installed and maintained in the Unified Environment.
3. The user interface should be clear and simple so the users do not have questions about what and how to enter the weaving environment information.
4. The Sankey diagram shall present the road and lane information explicitly based on the user's input for the description of the weaving area.

TASI has regularly communicated with INDOT users to enhance the user interface and has completed development to meet all these requirements. Additionally, the following features have been added to improve the program's accuracy when user input is suboptimal or when recorded videos lack sufficient vehicles on certain lanes.

1. In the old program version, when the user specifies a vehicle counting reference line across a road, it is not clear how to determine the proper line width. If the line is too narrow, some lanes may not be detected. Conversely, when it is too wide, lanes on the opposite road may be incorrectly identified as lanes on this road. This issue will not be noticed until the program completes its execution many hours later. In this new version, we requested the user to specify the lane center more accurately and provide the camera location with respect to the road. The program uses this information to adjust the width of the reference lane specified by the user.
2. We use the trajectory of the detected vehicles on the image to determine the position of lanes on the reference line. The detected position of the lanes is more accurate when more vehicles are passing the lane. In the previous version of the weaving analysis program, in cases where highway traffic is very light, almost all vehicles are driven in the slow lane, and few vehicles stay in the fast lane for an extended period. This situation makes the lane detection algorithm fail. In this new program version, we developed an algorithm to compare the number of detected lanes with the number of user-specified lanes. If they do not match, we add the missing lane and remove false-detected lanes by using the locations of the detected lanes in relation to the user-specified lanes and camera locations with respect to the road.

4.1 Weaving Analysis Program Execution Environment Structure

To support the web-based user interface for the Highway Weaving Analysis Program (HWAP), the TASI team interacted with INDOT IT experts to understand the environment available for eventual program deployment to INDOT. The following is the system that INDOT can provide support for with minimal effort on INDOT's part in the virtual machine.

1. A virtual machine without the GPU capability
2. Ubuntu OS
3. Nginx Web Server
4. Postgres database

Since the weaving program requires video processing that needs a GPU, INDOT provides a standalone PC with a GPU as an application server. Therefore, the architecture envisioned for the highway weaving analysis program in INDOT is as shown in Figure 4.1.

To ensure the final deployment and maintenance of the HWAP in the INDOT environment, the TASI team developed two sets of environments to mimic the INDOT environment. One is a network in the TASI lab (see Figure 4.2). This system is used for program development. The other is a Purdue cloud-based system (see Figure 4.3). This system serves as the testing bed for final product deployment, maintenance, and operation.

Therefore, the improved highway weaving analysis program is distributed across two types of computers: virtual machines/ Cloud servers, as well as application servers.

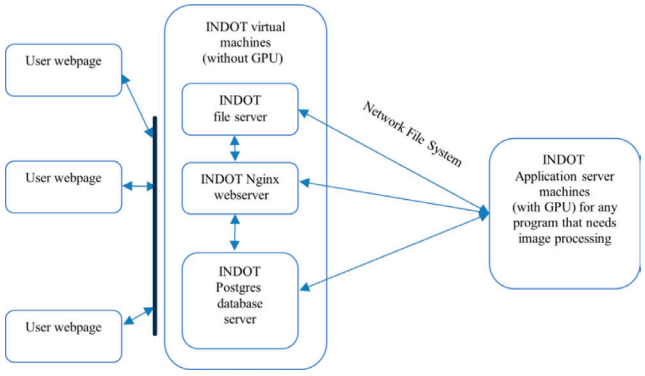


Figure 4.1 INDOT's Desirable System Architecture.

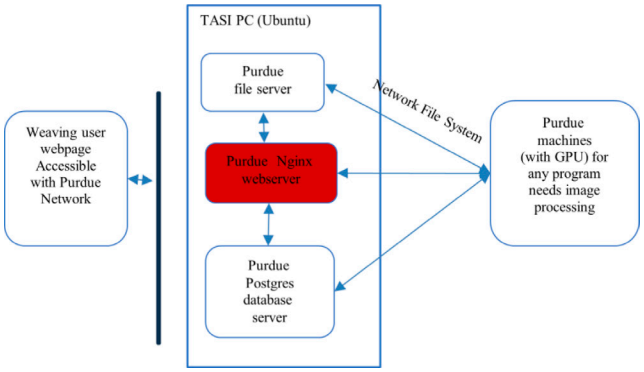


Figure 4.2 TASI's Computation Environment Mimics the INDOT Environment.

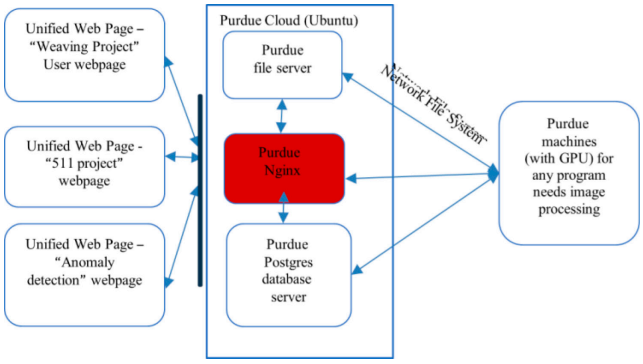


Figure 4.3 Purdue Cloud Computation Environment Mimics the INDOT Environment.

4.2 Highway Weaving Analysis Tool Software Structure

The improved highway weaving analysis program consists of three main components: (1) a Web-Based Weaving Area Specification GUI, (2) a Vehicle Detection and Matching Program, and (3) a Web-Based Vehicle Matching Verification and Result Generation GUI.

The Web-based Weaving Area Specification GUI guides users through specifying the weaving scenario, providing all the necessary information in an easily accessible format. The accuracy and detail of user-provided information significantly impact the weaving analysis and the quality of the presentation result. This component runs on the virtual machine web server. After gathering all required weaving scenario information, this program starts the Vehicle Detection and Matching Program to generate the weaving analysis results.

The Vehicle Detection and Matching Program takes the request from the Web-based Weaving Area Specification GUI. It executes steps as traffic environment learning (vehicle detection and counting, lane identification, lane-based vehicle counting, vehicle feature detection, vehicle matching in the entry and exit locations, and generating the machine learning based weaving analysis results. Since these tasks require high computation power (which typically takes hours to execute for a 10-min weaving video), we run this program on an application server with a reasonably good GPU. Due to the limitations of state-of-the-art technology, the precision of the result is not satisfactory. There are high rates of false positives and false negatives.

The Web-Based Vehicle Matching Verification and Result Generation GUI is designed to address the precision issues in the results of the Vehicle Detection and Matching Program. By treating the detected vehicle matches as samples and removing the false positive matches, we can estimate the lane-based weaving analysis. This GUI presents all machine-decided vehicle matching pairs one by one to the user, and the user determines whether they are a true match. The result will be used to estimate lane-based weaving analysis results. Combining the weaving area environment information provided by the user in the Web-based Weaving Area Specification GUI, this GUI generates Sankey diagrams and PDF reports for the results of weaving analysis. This GUI is executed on virtual machines/clouds.

To enable the information passing between these three program components, the intermediate and results of each analysis of each weaving scenario are stored in the file server under a standard file structure (see Figure 4.4).

4.3 Web-Based Weaving Area Specification GUI

The goal of this GUI is for the user to provide as much known information as possible relevant to weaving analysis with minimal effort. The program maximally utilizes the user-provided information to avoid unnecessary searches and derivations for the needed information to generate weaving analysis results. Figure 4.5 shows the first web page of the GUI. On this page, users can perform the following actions.

a. Run a new weaving analysis scenario by assigning a unique name (character string) to the new study, or continue the started weaving analysis earlier to finish the verification step by selecting the name of a previously studied weaving case.
b. Learn how to use this Highway Weaving Analysis Tool through a demo case
c. Access the instruction document for properly recording the videos for weaving analysis
d. Access the user manual

4.3.1 Run a New Weaving Analysis Scenario

When the user creates a new weaving scenario analysis, they must provide a unique scenario name or Weaving ID (see Figure 4.5). Then the GUI creates a new folder using this new scenario name with a predefined structure for all information generated throughout the analysis for this scenario. Figure 4.6 is the second web page that appears after the user specifies a unique scenario name. The general user interaction is on the left side of the page and flows from top to bottom. The right side displays the weaving area map, enabling the user to specify input information graphically.

A weaving case has two locations: the entry and the exit. On this page, the user must select a previously loaded entry location video file or upload a new one. Since INDOT does not store videos, users may need to upload a video each time a scenario is analyzed. The user is asked to upload the weaving area

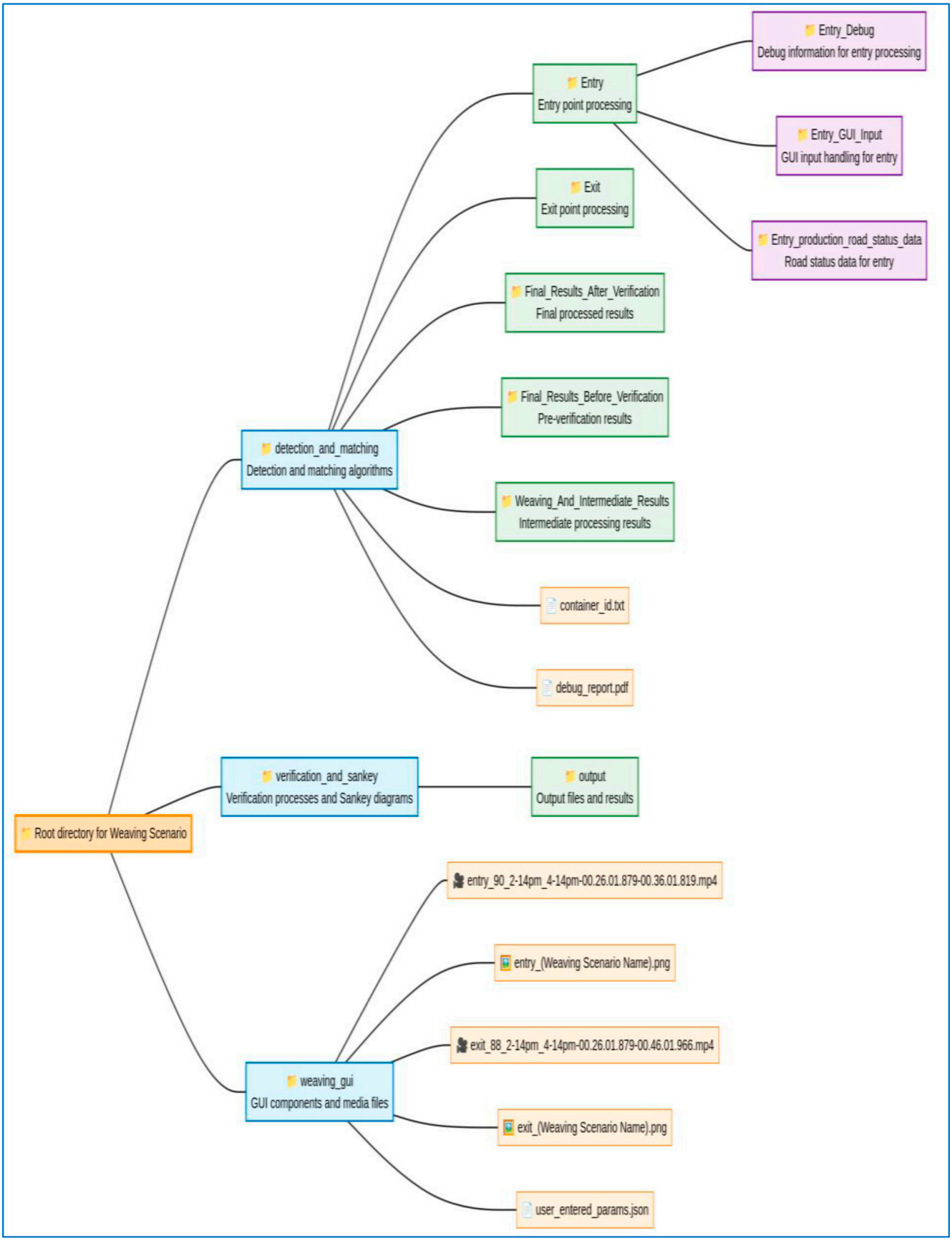


Figure 4.4 The File Structure for Each Weaving Scenario.

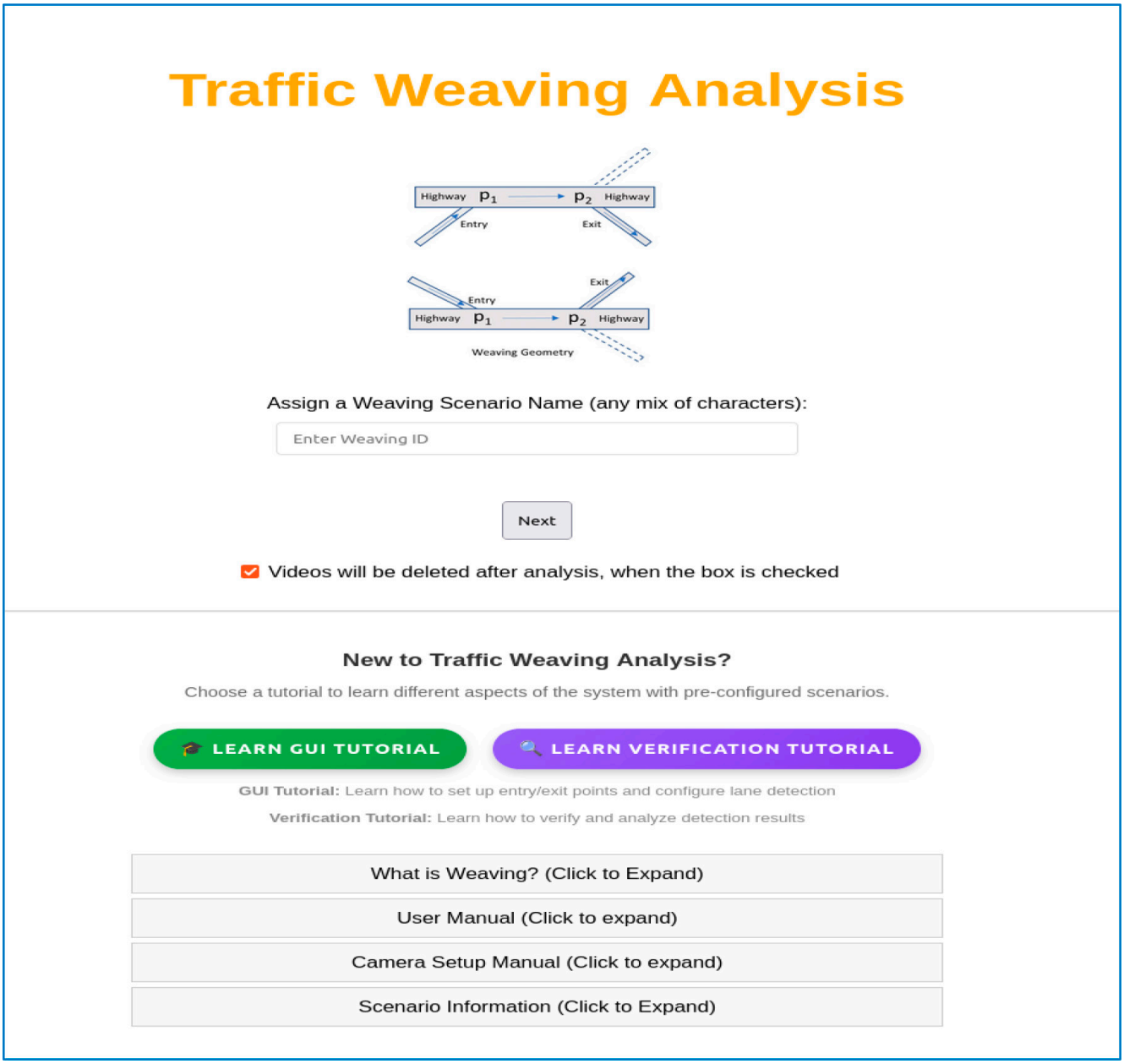


Figure 4.5 First GUI Page for Weaving Area Specification GUI.

videos from their local storage (e.g., a flash drive), as shown in Figure 4.6. This is important as INDOT does not allow storing any camera videos on the computer system after the completion of the weaving analysis. The videos are copied to the INDOT file server temporarily for processing and then deleted after the weaving analysis is completed. Only key frames are stored (no more than one frame per 5 s).

After the user uploads and selects the entry location video file, the GUI proceeds to the next page, as shown in Figure 4.7, which prompts the user to provide entry location information.

Then, the user is asked to indicate where on the road the vehicle counting should be performed by drawing a reference line or baseline segment across the road and another line segment on the other road. The center locations of all lanes on each road are marked by the user on the frame displayed. It should be noted that the user-specified lane centers may differ from the program-perceived road centers, as the camera may be located on the roadsides and view the road at an angle.

In the previous version of the program, the width of the line segment influenced the performance of the weaving program, as lanes on the other road could be mistakenly included as part of this road if the line segment was too long, or a lane might be excluded if the line segment was too short. It was difficult for users to determine the appropriate width of the line segment. In this project, instead of requiring users to specify the line segment width, we pass the lane locations they specify, along

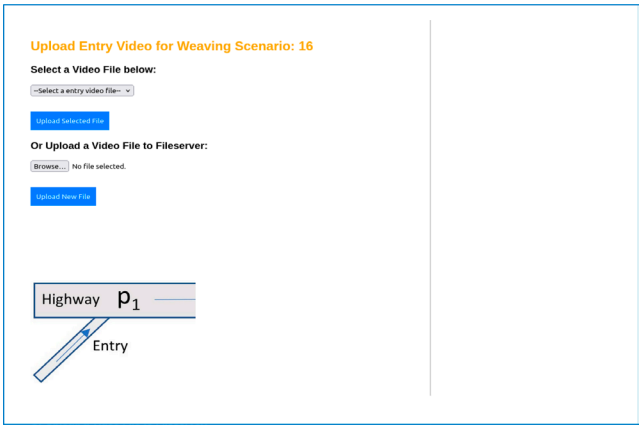


Figure 4.6 The Second Web Page of the Weaving Area Specification GUI.

with the camera's position relative to the road, to the Vehicle Detection and Matching Program. This program automatically adjusts the user-drawn reference line segments to the correct width, thereby significantly improving the user experience.

The user can add a short phrase to describe the scenarios, the entry location, the exit location, and each lane (see Figure 4.8). The descriptions will be incorporated into the analysis of results. The carefully thought-out description can greatly help the understandability and readability of the weaving analysis results.

The same process for the user to enter the entry location information is repeated for the exit location. After the user enters all required information for both entry and exit locations, the information is organized in a JSON file and passed to the application server for weaving analysis (see Figure 4.9). The structure of the JSON description of the weaving entry and exit areas is shown in Figure 4.9. The last web page for this GUI is the status indicator for the weaving analysis program (see Figure 4.10).

4.3.2 Continue From Previously Started Analysis

When the user decides to revisit the previously started weaving analysis, the unique weaving ID should be selected on the first GUI page (see Figure 4.11)

After the user selects an existing weaving scenario name, the GUI directs the user to one of three different places, depending on the execution status of this weaving scenario. Each situation is listed below based on the status:

1. When the user selects a previously started scenario and its Vehicle Detection and Matching Program execution is in progress, a Prompt window pops up indicating that the Vehicle Detection and Matching Program execution is in progress. (see Figure 4.12). The user should then wait for it to be completed. If they want to cancel the started process, the user can go to /containermanagement in the Unified portal, select the weaving program, select the weaving ID they just created, and click on stop.
2. When the user selects a previously started scenario and its Vehicle Detection and Matching Program has correctly finished, the GUI goes to the pages, as shown in Figure 4.11. So, the user can proceed with the verification process.
3. When the user selects a previously started scenario and its Vehicle Detection and Matching Program was not correctly executed, the GUI goes to the pages shown in Figure 4.6 which restarts the

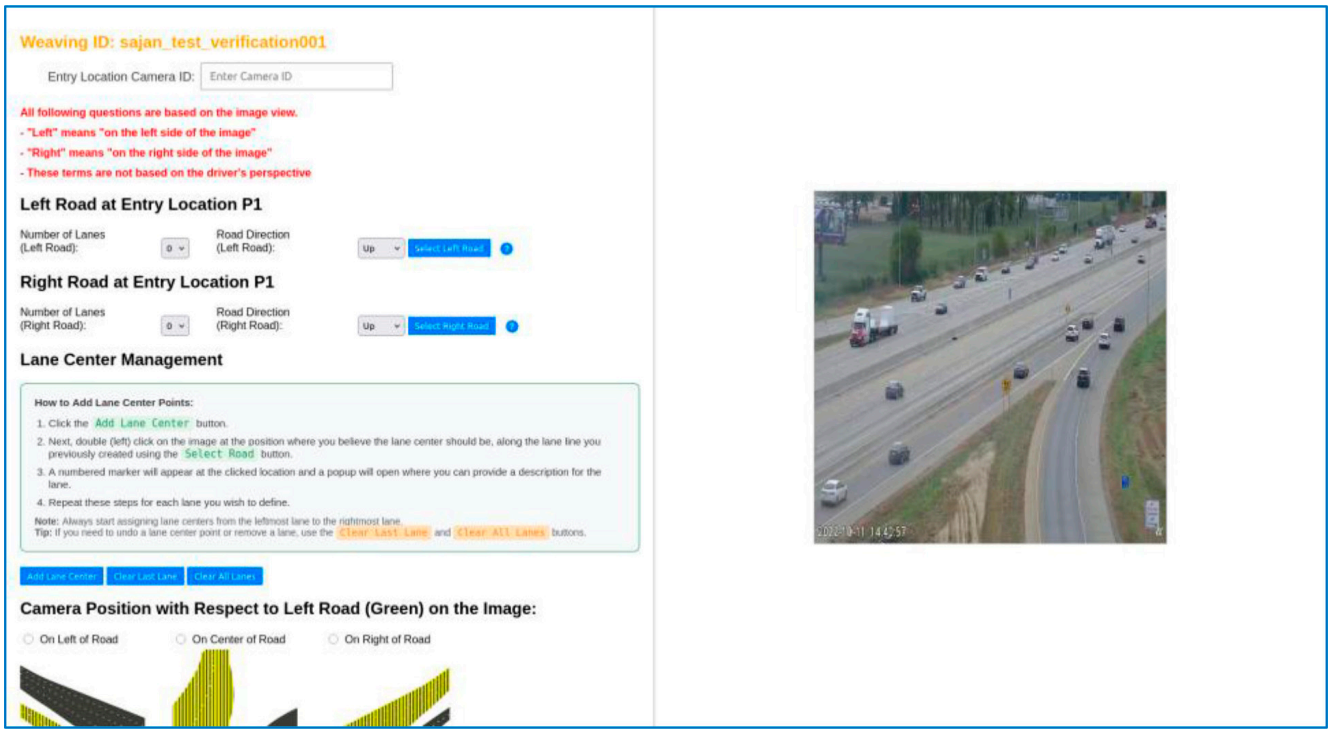


Figure 4.7 The Third Web Page of the Weaving Area Specification GUI.

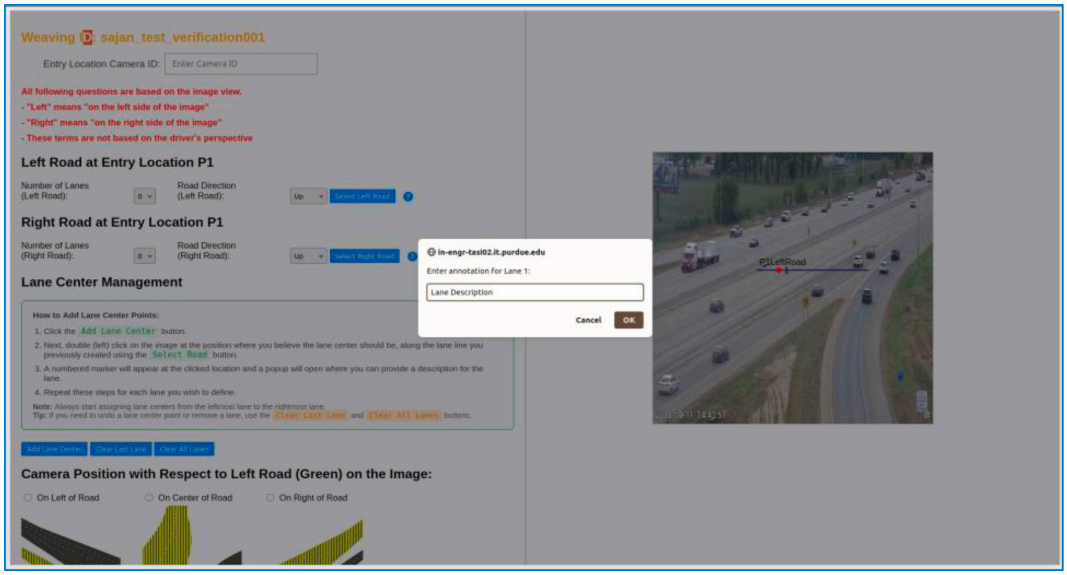


Figure 4.8 A Window to Add a Lane Description.


```
{
  "p2_data": {
    "location": "Exit",
    "cam_id": "88",
    "weaving_id": "sajan_test_verification001",
    "fps": 30,
    "link": "/media/nfs/file_server/weaving/sajan_test_verification001/weaving_gui/exit_88_2-14pm_4-14pm-00.26.01.879-00.36.01.755.mp4",
    "baseline": {
      "P2LeftRoad": [
        [
          25,
          385
        ],
        [
          322,
          385
        ]
      ],
      "P2RightRoad": [

      ]
    },
    "lane_num": {
      "P2LeftRoad": 0,
      "P2RightRoad": 0
    },
    "dir": "Up",
    "lane_centers": [
      {
        "lane_id": 1,
        "road_name": "P2RightRoad",
        "position": [
          136,
          385
        ],
        "annotation": "Center"
      }
    ],
    "exit_left_road_name": "Post Rd Highway",
    "exit_right_road_name": "Post Rd Exit Lane",
    "camera_location_LeftRoad": "Left",
    "camera_location_RightRoad": "Left"
  },
  "drawing_coordinates": {
    "highwayPoints": [
      [
        25.0333251953125,
        384.5
      ],
      [
        322.0333251953125,
        384.5
      ]
    ],
```


Figure 4.9 A JSON File Captures All Users' Input About the Scenario. This File is Stored in the Application Server for the Next Step of the *Vehicle Detection and Matching Program*.

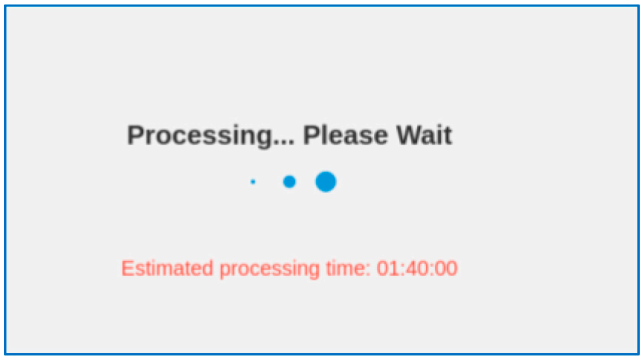


Figure 4.10 The Last Web Page of this GUI is the Status Indicator for the Running Vehicle Detection and Matching Program.

whole process again. Here, the user will have to re-enter the lane details and restart the process. *While this will rewrite the older files, we advise against using the old scenario when it has failed; instead, create a new scenario name and delete the old scenario from the file server (the file and folder clean-up feature will be added in the future).*

4.3.3 Learn How to Use This Highway Weaving Analysis Tool Through a Demo Case

The purpose of this option is to enable users to learn/practice the Highway Weaving Analysis Tool conveniently without the need to upload weaving videos and without starting the execution of a lengthy Vehicle Detection and Matching Program component. It stores video demos in the system and saves the program's previous execution results for learning convenience. While the number of web pages and page sequences in learning mode is the same as running a weaving analysis program, as described in Section 4.3.1, it also provides more assistance to the user at each step.

To access the Weaving scenario specification GUI demo, select the green buttons shown in Figure 4.5. This guide will assist the user in entering lane descriptions and details, as illustrated in Figure 4.13. The user will be able to see some playable videos while entering the lane description, as shown in Figure 4.14, to understand how to perform various actions. Ultimately, the user will be presented with a status page indicating that the demo has been completed, and no actual detection and matching program has been initiated, as this is a demo scenario (see Figure 4.15).

To access the Vehicle Matching Verification and Result Generation GUI (explained in Section 4.5), the user can select the purple buttons shown in Figure 4.16. It is a typical scenario but with additional information and brief video demonstrations to help users become familiar with the software. Here, the user will learn how to use verification-related tasks, such as precision matching and creating Sankey diagrams. The user can do any of the tasks an unlimited number of times. Every time the user completes a verification step, a new Sankey diagram is generated, replacing the previous one.

4.3.4 Access the Video Recording Instruction Document

The quality of the recorded video at weaving entry and exit locations is crucial to the accuracy of the analysis results. We provide a

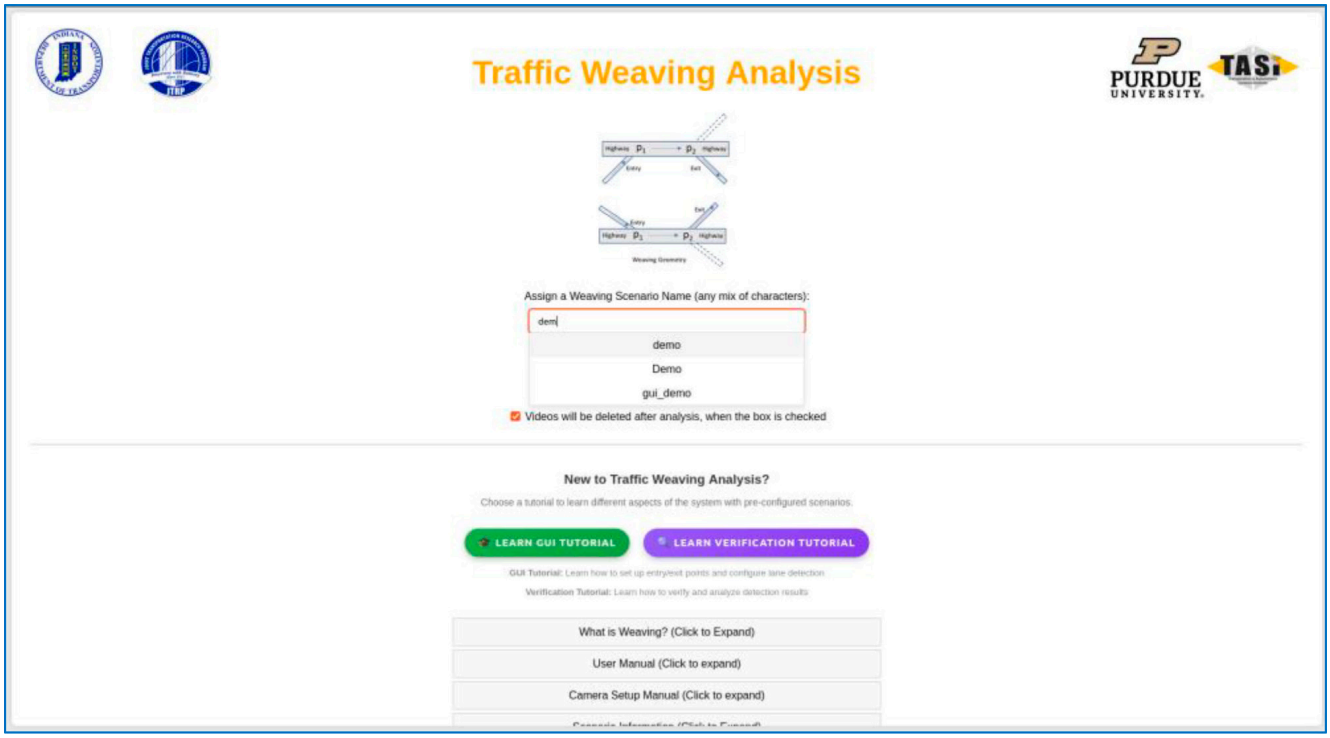


Figure 4.11 Select a Previously Started Weaving Scenario.

Figure 4.12 Already Executing a Weaving Scenario.

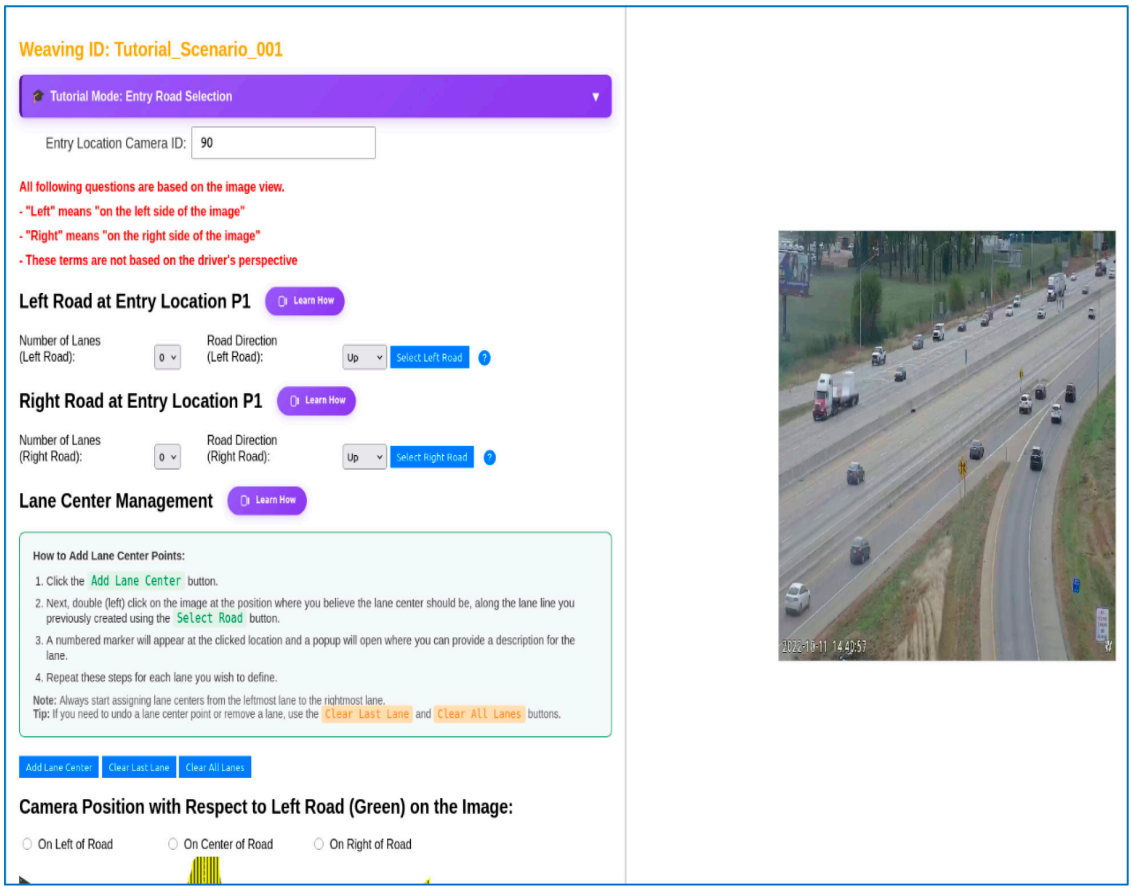


Figure 4.13 Demo GUI Entry Page.

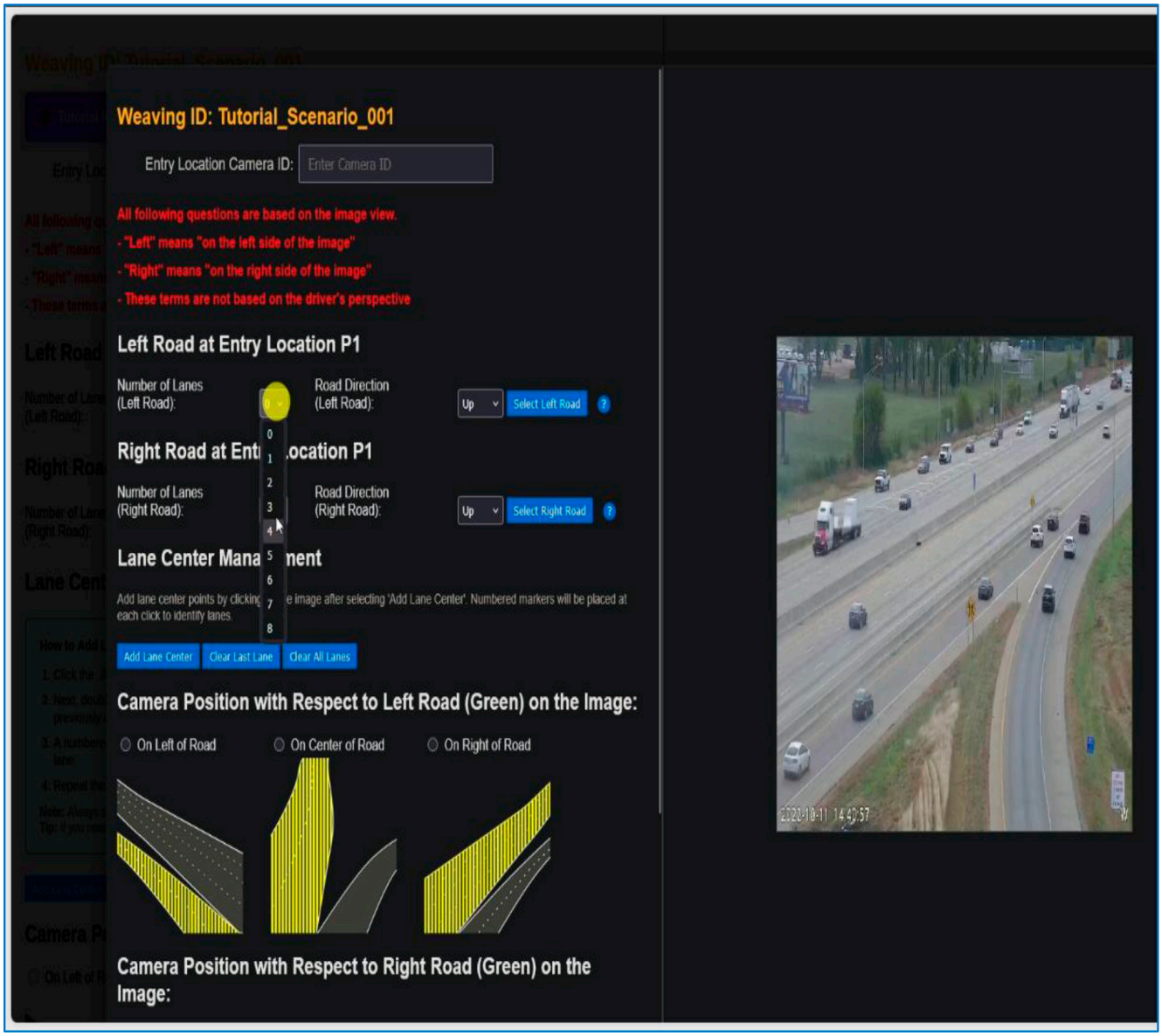


Figure 4.14 Demo Video Can Be Accessed on the Entry/Exit Page to Learn the Functions of the Buttons in Purple in Figure 4.13.

video recording and an instruction document to help users record videos optimally for weaving analysis. This document can be downloaded from this software and saved for future reference.

4.3.5 Access the User Manual

A user manual for using this software is also included in this document, which can be saved and printed for future reference (see Figure 4.12).

4.4 The Vehicle Detection and Matching Program

The Vehicle Detection and Matching program is the technical core of the weaving analysis. The algorithm used is the same as described in the SPR-4738 final report from Chien et al. (2024). The modification in this program focuses on reducing the requirements for the user and improving efficiency.

a. In the previous version of the program, the width of the line segment affected the performance of the weaving program because the lanes on the other road could be mistakenly included as part of this road if the line segment was too long or a lane on the road could be excluded if the line segment was too short. It was difficult for the user to determine the appropriate width for the line segment. In this project, instead of requiring the user to specify the line segment width, we pass the user-defined lane locations and the camera's position relative to the road to the Vehicle Detection and Matching Program. Within this program, we use the reference line locations, the number of lanes, the lane center points, and the camera's position with respect to the roads—information provided via the GUI—to automatically determine

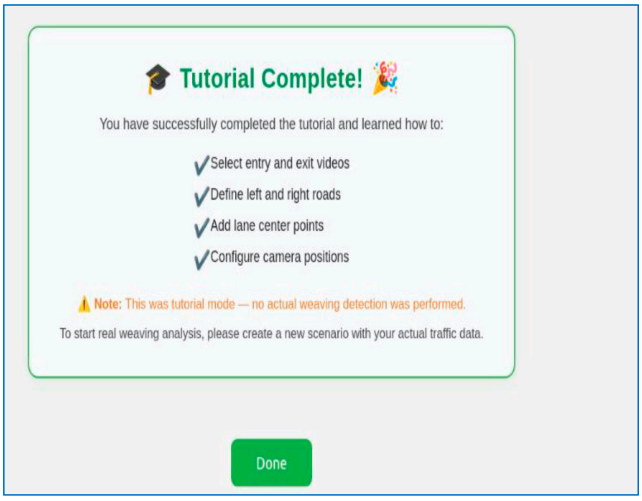


Figure 4.15 Status Page for GUI Demo.

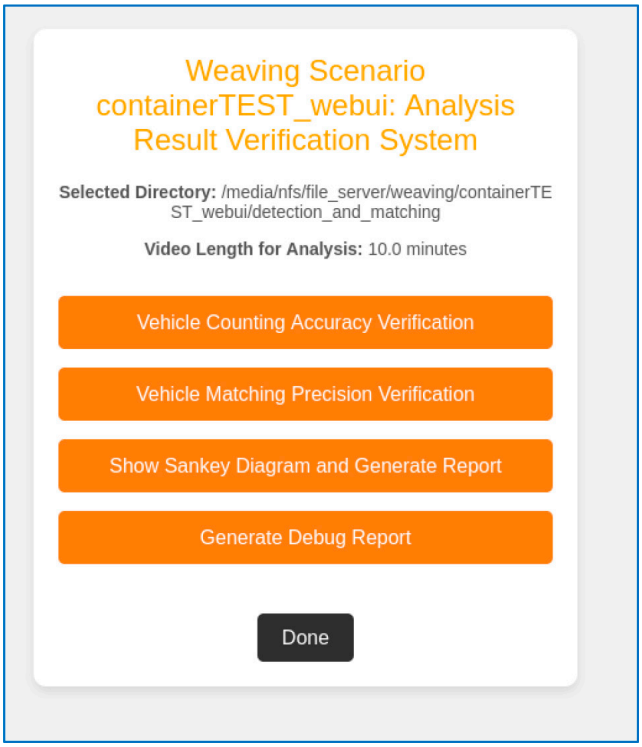


Figure 4.16 The Start Page for the Verification GUI.

the proper reference line segment width. The program then automatically adjusts the user-drawn reference line segments to this proper width, significantly improving the user experience.

b. Reduce the program execution time by optimizing the code. We optimized the program to reduce GPU memory usage, thereby maximizing GPU computation, and achieved a 40% reduction in processing time.

c. Reduce the lane learning time. As the lane locations from the video frames of different cameras or even the same camera captured at different times can be different, the lane centers in the video frames need to be identified. Although we ask the user to specify the lane centers, the information does not match the lane centers observed from the camera based on the detected vehicle motions. The lane centers observed by the camera shift from the user-labeled lane centers to the left or right based on the camera's location with respect to the road. The program statistically detects the lane centers based on the motion trajectories of all vehicles detected in the videos. When there are few vehicles at night, the program waits for a long time to get enough vehicles passing through each lane to determine the lane centers. In the worst case, when the traffic is very light, all vehicles are in the slow lane and no cars are passing through the fast lane. Then the fast lane cannot be detected. Since the program prompts the user to enter the number of lanes and lane centers in the GUI, it can estimate the camera-perceived lane centers based on some lane centers detected using real vehicle detections. Therefore, the program only monitors road traffic for a predefined period. If some lane centers are detected, the lane centers of the remaining lanes will be derived from the user-specified lane center location and the detected lane center locations.

d. All data obtained from the input files provided by the Weaving Scenario Description GUI and the results generated by this program are stored in the JSON file. The JSON file format is shown in Figure 4.9.

The execution of this program is triggered at the end of the Web-Based Weaving Area Specification GUI when the user initiates a new weaving analysis (see Figure 4.5). The status of the program execution can also be checked in the Web-Based Weaving Area Specification GUI (see Figure 4.12).

The input to the Vehicle Detection and Matching program is a JSON file named user_entered_params.json, which contains all the user-provided information, along with two video files (i.e., entry and exit videos). Additionally, two representative video frames from each uploaded video are stored (see Section 3.3).

The output of the Vehicle Detection and Matching program includes data stored on the file server in two main folders: 'detection_and_matching' and 'verification_and_sankey'. These folders contain data for different processes in the vehicle weaving analysis workflow.

The detection_and_matching folder holds information about the steps the weaving algorithm took to detect vehicles and match them between entry and exit points. This directory contains detailed data organized into separate processing phases, with three main subfolders indicating the timing of the detection and matching process (see Figure 4.1 for folder orientations).

4.4.1 Entry Directory

Contains all data and processing results for the entry point analysis:

- Entry_Debug: Comprehensive debugging data including JSON files, raw/processed video recordings, cropped vehicle images, learned parameters, timing validation, and test results
- Entry_GUI_Input: Reserved directory (currently unused)
- Entry_production_road_status_data: Lane learning diagrams from video analysis, essential for subsequent lane weaving program execution

4.4.2 Exit Directory

Contains all data and processing results for the exit point analysis:

- Exit_Debug: Mirror structure of Entry_Debug with comprehensive debugging data for exit processing
- Exit_GUI_Input: Reserved directory (currently unused)
- Exit_production_road_status_data: Lane learning diagrams from exit video analysis

4.4.3 Weaving_And_Intermediate_Results Directory

Stores intermediate processing results and vehicle matching analysis:

- Mapping_Results.npy: Vehicle mapping results array
- matched_pairs.json: Complete matched vehicle pairs dataset
- car_matched_pairs.json & truck_matched_pairs.json: Vehicle-type specific matched pairs
- Filtered results: acc_filtered, time_filtered, Removed_Dup files (both .jpg and .npy formats)
- time_plot.jpg: Temporal analysis visualization

4.4.4 Final_Results_Before_Verification Directory

Contains final analysis results in a visual format before quality verification. Sankey diagrams (HTML and PNG formats) for:

- All vehicles (cars and trucks)
- Cars only
- Trucks only

4.4.5 Final_Results_After_Verification Directory

Destination for verified results after the user verification process (populated post-verification).

4.4.6 Root Level Files

- container_id.txt: Container identification information
- debug_report.pdf: Comprehensive debug report (26.5 MB)

Similarly, ***Verification_and_Sankey holds information about the verification process*** completed by the user, including precision matching, vehicle counting, and the generation of Sankey diagrams. Once the user starts this process, the details are processed and stored under this directory. The contents of this directory and their meaning are as follows:

- Root Level Files
  - All Vehicles (After Verification)_lane_labels.json: Lane labeling data for all vehicle types after the verification process
  - Cars Only (After Verification)_lane_labels.json: Lane labeling data specifically for cars after verification
  - Trucks Only (After Verification)_lane_labels.json: Lane labeling data specifically for trucks after verification
  - entry_vehicle_count.json: Vehicle count statistics at entry points
  - exit_vehicle_count.json: Vehicle count statistics at exit points
  - user_params_reorder.json: User-defined parameters for reordering and configuration
- Output Directory: Contains final verified Sankey diagrams and comprehensive reporting:
  - All Vehicles (After Verification) Sankey Diagram.html: Interactive Sankey diagram for all vehicle types
  - All Vehicles (After Verification) Sankey Diagram.png: Static image version of all vehicles Sankey diagram
  - Cars Only (After Verification) Sankey Diagram.html: Interactive Sankey diagram for cars only
  - Cars Only (After Verification) Sankey Diagram.png: Static image version of the cars-only Sankey diagram
  - Trucks Only (After Verification) Sankey Diagram.html: Interactive Sankey diagram for trucks only
  - Trucks Only (After Verification) Sankey Diagram.png: Static image version of trucks-only Sankey diagram
  - sankey_report.pdf: Comprehensive Sankey analysis report

4.5 Web-Based Vehicle Matching Verification and Result Generation GUI

The goal of this GUI is for the user to verify the ML-based vehicle matching results and generate a user-friendly weaving analysis report. The matching is determined based on the similarity of the features generated from the detected vehicles. The ML-generated matching accuracy is low, and there is a significant rate of false positives, especially when the vehicle resolutions are low, as seen in the INDOT CCTV videos. However, assuming we see m vehicles on the entry video and m vehicles on the exit video, we need to check $m \times m$ pairs of vehicles. If n and m are in thousands, human checking the vehicle matches on this large set of $m \times m$ vehicle pairs is not practical. However, the result of the vehicle detection and matching program reduces the total number of potentially matched vehicle pairs to less than m, which is significantly less than m^2. This makes human checking of the matching results acceptable. This web-based GUI makes the matching check very convenient by bringing pictures of the matched vehicle pairs side by side and only asking the user to determine if they truly match or not. It takes an average of 5 s to check each pair. After the user finishes checking the correctness of machine-generated matching vehicle pairs, the GUI eliminates all false positive matches and scales up the matched pairs to the full vehicle population and generates a report.

The first page of this Web-Based Vehicle Matching Verification and Result Generation GUI asks the user to enter the time length of the scenario video to be verified (see Figure 4.17). Then, the user is presented with a new page that allows for multiple task selection, where various GUI-based human verification and Sankey diagram and report generation actions are performed (see Figure 4.18).

Task Selection:

1. **Vehicle Counting Accuracy Verification:** After selecting this option, the user is presented with processed videos that the user can view. After clicking the Verify Entry/Exit button, the user can click the plus button at the bottom of the screen, adjacent to each lane, when a vehicle passes through the boxed area in the video. This updates the vehicle counter, as shown in Figure 4.19. The video can be replayed to check the counting for each lane.
2. **Vehicle Matching Precision Verification:** In this task, the program presents the image of each ML-generated matched vehicle

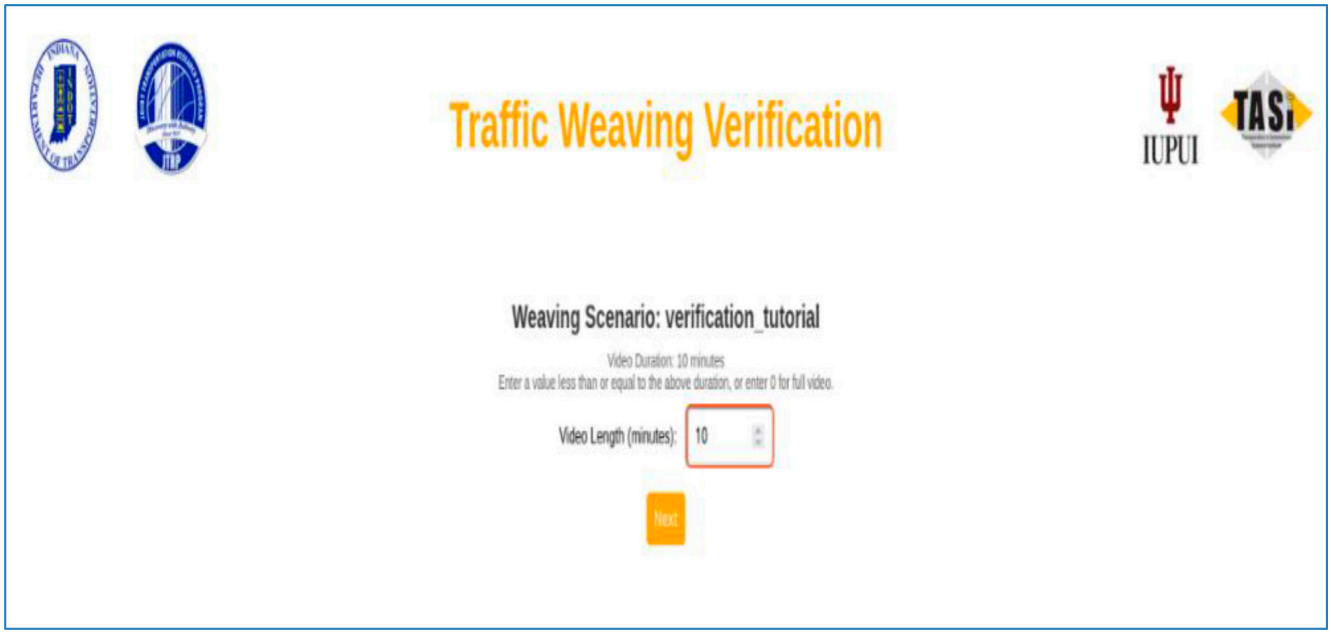


Figure 4.17 Weaving Scenario Verification for Duration Selection.

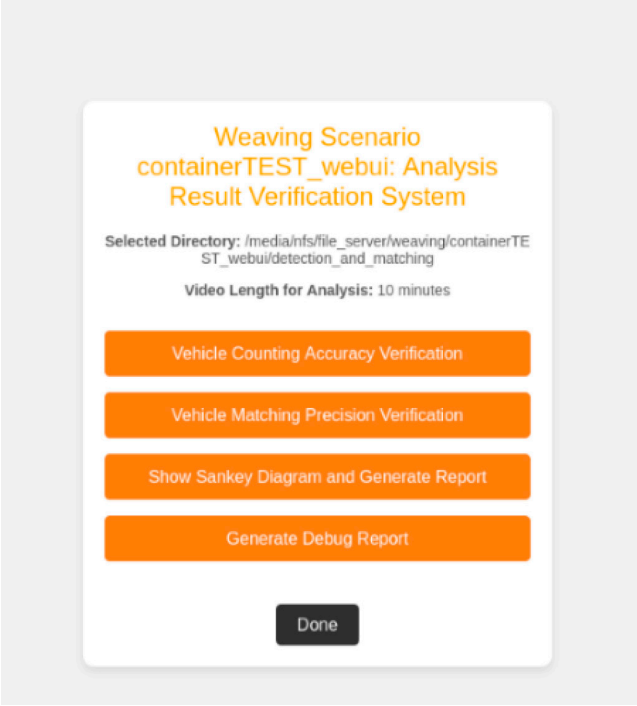


Figure 4.18 Weaving Scenario Verification Task Selection.

pair one by one and lets the user decide if it is a true or false match. If it is difficult to determine the match, the user can select different images of the same vehicle, hopefully to get a better view of the vehicle. Once a decision is made, the user presses on True Match or False Match, as shown in Figure 4.20.

3. **Show Sankey Diagram and Generate Report:** There are two ways to view the report generated:
   a. The report shown on the webpage (see Figure 4.21)
   b. The report can be downloaded as a PDF file (see Figure 4.22)

Figure 4.21 shows the report as it appears on the webpage. The report includes a video frame of the entry location, a video frame of the exit location, and all user-provided information related to the scenario. The center of the report features a Sankey diagram. The left side of the Sankey diagram represents the entry location, while the right side represents the exit location. The lanes at the entry and exit points are listed from top to bottom, in order. The report displays the name of the entry location, exit location, and each lane provided by the user in the Weaving Scenario Description GUI. The width of the curved lines, connecting lanes from entry to exit, indicates the relative number of vehicles between the two lanes. The report includes data for all vehicles, as well as separate data for cars and trucks. The same report can be downloaded.

4. **Debug Report:** The debug report contains the results of all intermediate execution steps. It is intended for software development purposes and is not intended for general use. A page of an example debug report is in Figure 4.22.

4.6 Highway Weaving Analysis Program Version Control, Packaging, and Deployment Process

The Highway Weaving Analysis Program is a thorough system that employs various modern software tools and a structured deployment process to ensure its functionality and ease of maintenance. This section explains the version control, packaging, and deployment procedures established for the program. These steps are essential for managing the development lifecycle, ensuring consistent and reliable installations, and supporting

Figure 4.19 The Vehicle Counting Verification Page.

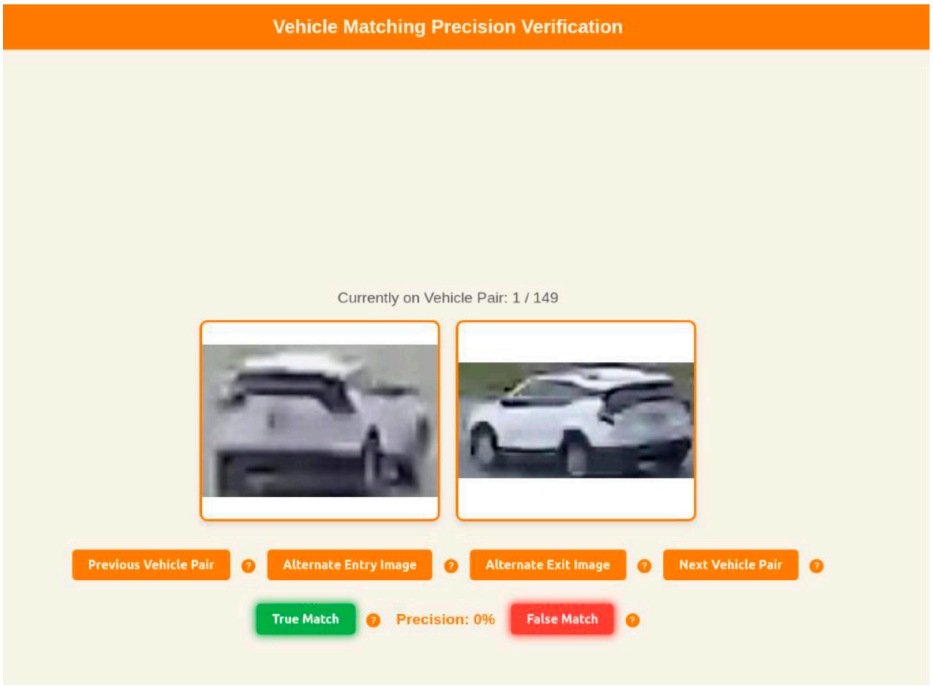


Figure 4.20 The Vehicle Matching Precision Verification Page.

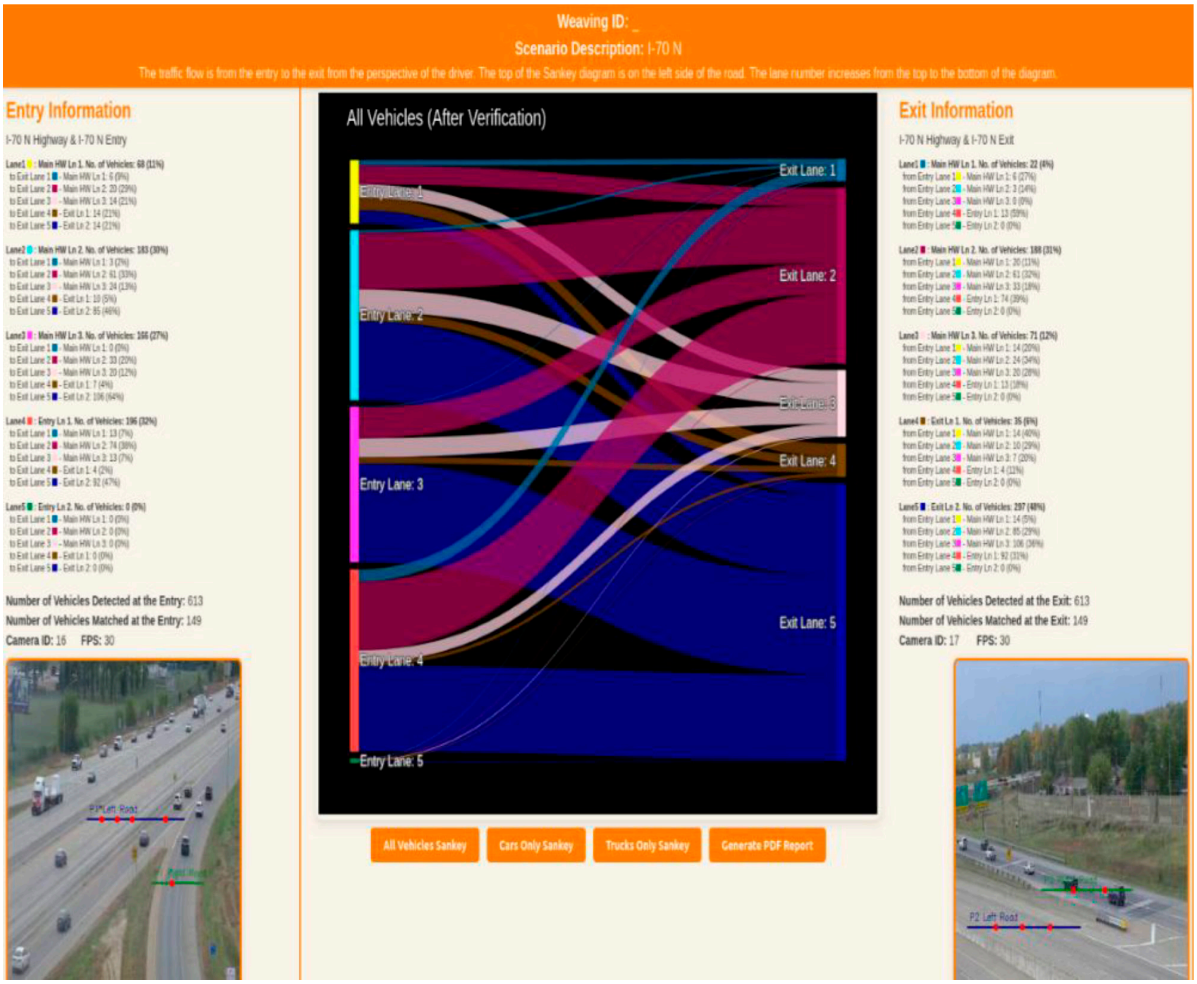


Figure 4.21 The Sankey Diagram Page of the Online Weaving Analysis Report.

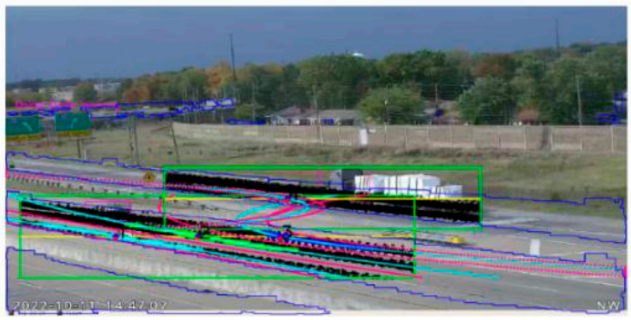


Figure 4.22 A Page From A Debug Report Shows the Detected Lane Boundaries.

future updates and maintenance. The upcoming subsections will provide an overview of the specific tools and methodologies used to manage the program's source code, prepare it for distribution, and deploy it within the designated server environments. This documentation acts as a guide for developers and IT staff responsible for maintaining and upgrading the Highway Weaving Analysis Program.

4.6.1 Version Control with GitHub

The integrity and history of the program's source code are managed using the Git version control system. All software components, including the main application logic and the web interface, are stored in a central repository hosted on Purdue's GitHub instance. This repository is structured with Git submodules, which allows for the modular management of different program components while ensuring that a deployment pulls the correct, compatible versions of all necessary codebases. The deployment process begins by cloning this central repository onto the target servers. A dedicated system user performs this to ensure process ownership and appropriate permissions. The use of submodules ensures that the entire, verified codebase is retrieved in a single, atomic operation, providing a reliable foundation for subsequent build and deployment steps.

4.6.2 Environment Standardization

To ensure seamless operation and eliminate environment-related discrepancies, a standardized configuration is applied to

all servers involved in the deployment. This includes creating a consistent user account, establishing a shared file system, and installing the required software runtimes.

- **Dedicated User and Permissions:** A dedicated user, say tasi_prod, is created on each server with a consistent User ID (UID 4000) and Group ID (GID 984 for the Docker group). This uniformity is critical for preventing file permission conflicts, particularly across the shared network storage.
- **Network File System:** A Network File System (NFS) is set up to enable data sharing between the application and web UI servers. The application server functions as the NFS server, exporting a designated directory where all shared files, such as video inputs and analysis results, are stored. The web UI server is configured as an NFS client, mounting the shared directory to provide the web interface with access to the necessary data.
- **Software Runtimes:** The program requires Python version 3.7 or higher to be installed on the servers. The environment setup includes verification of the Python version and, if necessary, installation.

4.6.3 Application Packaging and Deployment With Docker

The Highway Weaving Analysis Program is packaged and deployed using Docker containers. This approach encapsulates the application and its dependencies into isolated, portable units, guaranteeing that the software runs identically regardless of the underlying host environment.

- **Containerization and GPU Acceleration:** Due to the resource-intensive nature of video processing, the application is built to utilize NVIDIA GPUs. The Docker environment is configured with the NVIDIA Container Toolkit, enabling Docker containers to access the host machine's GPU hardware directly. This step is essential for achieving the performance needed for real-time traffic analysis. The setup is carefully managed to ensure the Docker runtime is properly connected to the NVIDIA drivers.
- **Configuration and Execution:** Before launching the application, configuration files within the cloned repositories are updated to specify the IP addresses of the application and database servers. Once configured, the deployment is finalized by executing a series of shell scripts (build.sh and run_docker.sh). These scripts automate the process of building Docker images and launching containers for each component of the system, including the core analysis engine and the web UI. A central Container Management service orchestrates the various application containers.

A PostgreSQL database is used in the backend to track the status and metadata of each weaving analysis scenario initiated by the user. The Postgres server is part of the Unified system, which is shared with other projects.

4.6.4 A Summary of the Highway Weaving Analysis Program Execution Process

The Highway Weaving Analysis Program has been successfully developed and deployed to a cloud environment that simulates the intended INDOT system architecture. This section offers a summary of the complete analysis workflow, illustrating how the components described in this report work together to produce a weaving analysis report.

The process starts when a user accesses the web-based interface hosted on a web server. To initiate a new analysis, the user assigns a unique name to the weaving scenario and uploads two video files: one for the entry point of the weaving section and one for the exit point. Using the Weaving Area Specification GUI, the user visually defines the roadway layout on a frame from each video. This involves specifying the number of lanes on each road, marking the center of each lane, and providing descriptive names for the roads and lanes. This user-provided information is essential for ensuring both the accuracy of the analysis and the clarity of the final report.

Once the user submits the scenario details, the system starts the Vehicle Detection and Matching Program. This resource-intensive task runs on a dedicated application server equipped with a powerful GPU to accelerate video processing. The program analyzes the videos to automatically identify exact lane locations based on vehicle paths, detect and count vehicles in each lane, and extract features from each vehicle. It then performs an initial match of vehicles between the entry and exit videos. This backend process has been optimized to enhance performance and user experience, including the automatic determination of the correct width of the vehicle counting reference lines based on user input.

Due to the technical challenges of vehicle re-identification from CCTV footage, the automated matching process can produce false positives. To address this, the system includes a human-in-the-loop verification step. After backend processing is complete, the user is directed to the Web-Based Vehicle Matching Verification and Result Generation GUI. This interface displays pairs of matched vehicle images from entry and exit points. The user confirms whether each pair is a true or false match. Additionally, the user can verify the vehicle counts for each lane.

Based on this verified data, the system calculates the final lane-to-lane weaving percentages. The results are shown in a detailed, user-friendly report that features Sankey diagrams illustrating traffic flow for all vehicles, as well as for cars and trucks separately. This final report, available on the webpage or as a downloadable PDF, includes user-provided descriptions to deliver a clear and understandable analysis of the traffic weaving patterns. The entire workflow, from user input to final report, is managed within a distributed architecture and deployed using standardized version control and containerization practices to ensure reliability and ease of maintenance.

5. CONCLUSION

All three tasks listed in this project have been completed successfully. A unified, web-based user interface has been established and hosted on the Purdue University Cloud. This interface provides a standard process for connecting all current and future JTRP project software. It now includes access to the weaving analysis program, the anomaly detection program, and the program from SPR-4937 (Ding et al., 2025)—which focuses on Enhanced Traffic Pattern Knowledge Abstraction and Presentation from the 511 Database—through a single web portal. The interface is available to both the public and INDOT

users with login credentials. The anomaly detection program has been expanded to enable lane-based anomaly detection across all cameras. Additionally, the weaving analysis program has been adapted for web access with a more user-friendly output.

REFERENCES

Aguiar-Conraria, L., & Soares, M. J. (2014). The continuous wavelet transforms: Moving beyond uni- and bivariate analysis. *Journal of Economic Surveys*, *28*(2), 344–375. https://ideas.repec.org/a/bla/jecsur/v28y2014i2p344-375.html

Chien, S., Christopher, L., Chen, Y., Qiu, M., & Lin, W. (2024). *Origin-destination vehicle counts in weaving area utilizing existing field data* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2024/02). Purdue University. https://doi.org/10.5703/1288284317719

Choudhary, H., & Hassani, M. (2023). *Enhancing traffic flow prediction using outlier-weighted autoencoders: Handling real-time changes*. arXiv. https://doi.org/10.48550/arXiv.2312.16596

Christopher, L., Chien, S., Chen, Y., Qiu, M., Reindl, W., & Koshy, L. (2024). *Anomaly detection in traffic patterns using the INDOT camera system* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2024/30). Purdue University. https://doi.org/10.5703/1288284317778

Ding, J., Koshy, L., Maheshwari, T., Mathew, A., Zhou, J., Chien, S. Y.-P., Li, T., Tao, C., & Chen, Y. (202X). Enhanced traffic pattern knowledge abstraction and presentation from 511 database (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2025/43). Purdue University. https://doi.org/10.5703/1288284318608

Lee, D., Malacarne, S., & Aune, E. (2024). Explainable time series anomaly detection using masked latent generative modeling. *Pattern Recognition*, *156*, 110826. https://doi.org/10.1016/j.patcog.2024.110826

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *6*(1), 1–39. https://doi.org/10.1145/2133360.2133363

Qiu, M., Christopher, L. A., Chien, S., & Chen, Y. (2022). *Attention mechanism improves YOLOv5x for detecting vehicles on surveillance videos* [Paper presentation]. 2022 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, United States. https://doi.org/10.1109/AIPR57179.2022.10092237

Qiu, M., Christopher, L., Chien, S. Y.-P., & Chen, Y. (2024). Intelligent highway adaptive lane learning system in multiple ROIs of surveillance camera video. *IEEE Transactions on Intelligent Transportation Systems*, *25*(8), 8591–8601. https://doi.org/10.1109/TITS.2024.3358732

Qiu, M., Reindl, W. L., Chen, Y., Chien, S., Shellhamer, N., McCoy, D., & Hu, S. (2025). *Lane-wise highway anomaly detection* [Paper presentation]. IEEE 28th International Conference on Intelligent Transportation Systems (ITSC), Gold Coast, Australia. https://doi.org/10.1109/ITSC60802.2025.11423824

Tuli, S., Casale, G., Jennings, N. R. (2022). *TranAD: Deep transformer networks for anomaly detection in multivariate time series data*. arXiv. https://doi.org/10.48550/arXiv.2201.07284

van den Oord, A., Vinyals, O., & Kavukcuoglu, K. (2017). Neural discrete representation learning [Paper presentation]. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)* (pp. 3645–3649). IEEE Press. https://doi.org/10.1109/ICIP.2017.8296962

About the Joint Transportation Research Program (JTRP)

On March 11, 1937, the Indiana Legislature passed an act which authorized the Indiana State Highway Commission to cooperate with and assist Purdue University in developing the best methods of improving and maintaining the highways of the state and the respective counties thereof. That collaborative effort was called the Joint Highway Research Project (JHRP). In 1997 the collaborative venture was renamed as the Joint Transportation Research Program (JTRP) to reflect the state and national efforts to integrate the management and operation of various transportation modes.

The first studies of JHRP were concerned with Test Road No. 1—evaluation of the weathering characteristics of stabilized materials. After World War II, the JHRP program grew substantially and was regularly producing technical reports. Over 1,600 technical reports are now available, published as part of the JHRP and subsequently JTRP collaborative venture between Purdue University and what is now the Indiana Department of Transportation.

Free online access to all reports is provided through a unique collaboration between JTRP and Purdue Libraries. These are available at docs.lib.purdue.edu/jtrp/.

Further information about JTRP and its current research program is available at engineering.purdue.edu/JTRP.

About This Report

An open access version of this publication is available online. See the URL in the citation below.

Chien, S., Hu, S., Chen, Y., Qiu, M., Reindl, W. L., Meng, N., Koshy, L., & Kumar, S. (2025). *Further refinement and integrated platform for INDOT traffic management and safety toolset* (Joint Transportation Research Program Publication No. FHWA/IN/JTRP-2025/36). West Lafayette, IN: Purdue University. https://doi.org/10.5703/1288284318603