Deep Learning Based Vehicle Re-Identification Based on Surveillance System

By

Muhammad Ahsan Tayyab

22-ARID-816

Zaheer Ahmed

22-ARID-738

Supervisor

Dr. Noureen Zafer

University Institute of Information Technology,

PMAS-Arid Agriculture University,

Rawalpindi Pakistan

Table of Contents

1.1 Brief.....1

1.2 Relevance to Course Modules.....2

1.3 Project Background.....2

1.4 Literature Review.....3

1.5 Analysis from Literature Review.....3

Table 1.1: Analysis from Literature Review.....4

1.6. Methodology and Software Lifecycle for This Project.....5

1.6.1. Rationale Behind Selected Methodology.....5

2.1. Problem Statement.....6

2.2. Deliverables and Development Requirements.....6

2.3. Current System (if applicable to your project).....8

3.1. Use Cases.....9

3.2. Functional Requirements.....18

3.3. Functional Requirements.....20

List of Figures

Fig 1.1 Block Diagram ……5

Fig 2.1 Use Case Diagram ……9

List of Tables

Table 1.1 Analysis from Literature Review ……4

Table 3.1: Use Case of Detect Vehicles ….……10

Table 3.2: Use Case of Extract Features ..……11

Table 3.3: Use Case of Perform Vehicle Re-Identification …..……12

Table 3.4: Use Case of Generate Trajectories ……13

Table 3.5: Use Case of Store Data in Database ……14

Table 3.6: Use Case of Detect Anomalies ……15

Table 3.7: Use Case of Visualize Results ……16

Table 3.8: Use Case of Evaluate Model Performance ……17

Chapter 1: Introduction

This chapter presents a complete overview of the project. It introduces the purpose and motivation behind the system, explains its relevance to Computer Science courses, outlines the research background, discusses relevant literature, and finally describes the methodology and software development approach adopted for implementation.

1.1 Brief

The rapid advancement of Artificial Intelligence (AI) and Computer Vision (CV) has revolutionized the field of surveillance and public security, transforming traditional closed-circuit television (CCTV) systems from passive video monitoring devices into intelligent analytics platforms capable of automated decision-making. In this era of smart cities and intelligent transportation systems, there is a growing demand for automated solutions that can analyze large volumes of video data efficiently and accurately without human intervention. One such promising technology that has emerged to address this challenge is Vehicle Re-Identification (Re-ID) , an advanced computer vision task that involves recognizing and matching the same vehicle across multiple, non-overlapping camera views distributed in different locations.

Traditional surveillance methods rely heavily on manual inspection by human operators, which is both time-consuming and prone to human error, particularly in large-scale networks where hundreds of cameras continuously capture vast amounts of data. In such environments, manually tracing a vehicle of interest across multiple cameras is almost impossible. This limitation has led to the rise of deep learning-based vehicle re-identification systems, which automate the process by learning unique and discriminative features of each vehicle. These features include color, shape, body type, brand logo, and texture patterns, which collectively enable the model to recognize a vehicle even under drastically changing environmental conditions such as lighting variations, camera angle differences, occlusions, and weather disturbances.

The Deep Learning-Based Vehicle Re-Identification System proposed in this project aims to design, implement, and deploy a robust AI-powered framework capable of automatically detecting vehicles from live or recorded surveillance feeds, extracting both deep visual and semantic features, and matching those features across multiple cameras to accurately re-identify the same vehicle. The system leverages advanced Convolutional Neural Networks (CNNs) and Transformer-based architectures to extract meaningful feature embeddings that are invariant to pose and viewpoint changes. In addition, it utilizes a feature database for efficient storage and retrieval of vehicle embeddings, enabling high-speed similarity comparisons during the matching process.

From a user perspective, the system provides two distinct roles: Surveillance Operator and Administrator. The Surveillance Operator interacts with a user-friendly dashboard interface to upload or stream surveillance footage, query vehicle images, and visualize re-identification

results, including matched vehicles, their similarity scores, and trajectories across camera networks. On the other hand, the Administrator has advanced privileges, allowing them to manage datasets, configure and retrain deep learning models, evaluate performance metrics (such as Rank-1 accuracy and mean Average Precision), and oversee system logs to ensure accuracy, reliability, and consistency of operation.

1.2 Relevance to Course Modules

This project integrates and applies concepts learned throughout our BSCS degree program, including:

- Artificial Intelligence (AI) For understanding and implementing machine learning and deep learning algorithms.
- Machine Learning & Deep Learning Core foundation for vehicle detection, feature learning, and re-identification.
- Computer Vision For image processing, object detection, and recognition using CNN and attention mechanisms.
- Data Structures & Algorithms Used for efficient data handling, feature extraction, and performance optimization.
- Software Engineering Guiding system development, requirement analysis, and SDLC documentation.
- Database Systems Managing image data, model outputs, and results.
- Human–Computer Interaction Designing a simple user interface for visualization of vehicle retrieval results.
- Operating Systems & Networks Understanding deployment constraints in surveillance and networking environments.

1.3 Project Background

In recent years, vehicle surveillance has become a critical component of smart city infrastructure. Traditional surveillance systems are mostly manual or limited to individual cameras, which makes tracking vehicles across multiple viewpoints nearly impossible.

Vehicle Re-Identification (Re-ID) is a computer vision task that aims to match the same vehicle across non-overlapping camera views. However, real-world challenges such as vehicles with similar appearances make the problem complex.

Our system aims to address these limitations by implementing a deep learning–based Re-ID model combined with semantic attribute learning (vehicle type, color, model) and multi-camera trajectory analysis.

Ultimately, this project will contribute to intelligent traffic monitoring, and public safety systems.

1.4 Literature Review

Extensive research in vehicle re-identification highlights the evolution from classical machine-learning methods to modern deep-learning approaches:

1. A Comprehensive Survey on Deep Learning–Based Vehicle Re-Identification (Amiri et al., 2024)
   Provides a complete overview of current datasets (e.g., VeRi-776, VehicleID) and challenges such as occlusion and illumination changes.
2. A Strong Baseline for Vehicle Re-Identification (CVPRW 2021)
   Establishes key baseline architectures and performance metrics (mAP, Rank-1 accuracy) used in evaluating Re-ID systems.
3. Strength in Diversity: Multi-Branch Representation Learning (WACV 2019)
   Introduces multiple feature extraction branches to capture diverse visual cues, improving discriminative capability.
4. Vehicle Re-Identification in Aerial Images and Videos (ICIP 2020)
   Discusses perspective variance and low-resolution problems in aerial surveillance.
5. Learning Part-Based Features for Vehicle Re-Identification with Global Context (Applied Sciences, 2025)
   Suggests using part-based and global features jointly for robust performance against viewpoint changes.

1.5 Analysis from Literature Review

From the reviewed literature, several insights emerge:

1. Gaps Identified:
   Most studies emphasize accuracy but not computational efficiency; few integrate semantic attributes or address real-world constraints (low resolution, bandwidth).
2. Proposed Enhancement:
   Our project combines CNN + attention + semantic fusion, balancing accuracy and efficiency for real surveillance deployment.
3. Novelty:
   Integration of trajectory analysis and semantic attribute learning into a single Re-ID framework.
4. Outcome:
   A robust model that performs well under occlusions and varying environmental conditions while remaining lightweight for deployment.

Table 1.1: Analysis from Literature Review

Feature / Aspect	Strong Baseline (CVPRW 2021) [2]	Multi-Branch Representation Learning [3]	Aerial Images & Videos Approach [4]	IBNT-Net (2025) [5]	CLIP-SE Net (2025)	Proposed System
Global + Local Features	✔	✔	Partial	✔	✔	✔
Multi-branch Network	✔	✔	✔	✔	Partial	✔
Attention Mechanism / Transformer	✘ / limited	Limited	Some?	✔	✔	✔
Semantic Attribute Awareness	✘ / limited	Some attribute branches	Maybe orientation / viewpoint, but limited semantically	Limited semantic attributes	Strong	✔
Trajectory	✘	✘	✘	✘	✘	✔
Efficiency (Parameters / Compute)	Baseline; okay	More complex	heavy	Designed to reduce parameters (group conv etc.)	Overhead due to CLIP etc.	Targeted for reasonable computational cost (for surveillance deployment)
Suitable for Surveillance Camera Network	✔ / baseline	✔ / research	Specialized (aerial)	Yes	Yes	Yes, aimed for deployment in realistic surveillance networks
Dataset Diversity Tested	Standard datasets	Standard plus variations	Aerial datasets + ground	VeRi-776, VehicleID	VeRi, VehicleID, VeRi-Wild	Use standard + possible collected local data

1.6. Methodology and Software Lifecycle for This Project

The system follows the Agile methodology, which promotes iterative design, experimentation, and performance evaluation. Agile allows quick modification of architectures (e.g., testing ResNet, Vision Transformer, or CLIP-based modules) and frequent feedback loops.

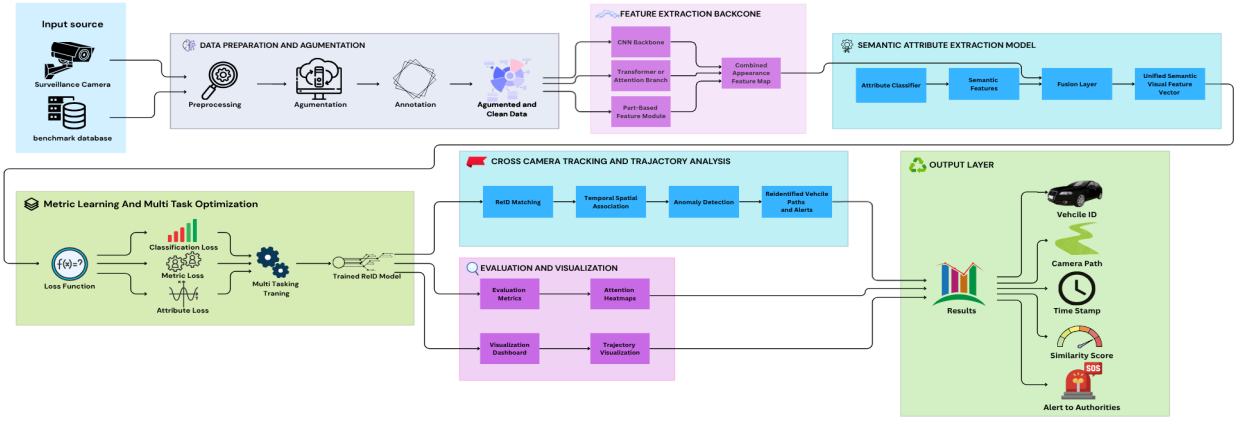

Figure 1.1: block diagram

1.6.1. Rationale Behind Selected Methodology

The choice of Agile is motivated by the following reasons:

- ➔ Flexibility: Supports frequent updates based on model performance and dataset variation.
- ➔ Parallel Development: Different team members can work on data preparation, model design, and testing simultaneously.
- ➔ Rapid Prototyping: Enables early testing of different architectures (baseline, attention-based, attribute fusion).
- ➔ Continuous Evaluation: Allows incremental performance tracking using standard metrics like mAP and Rank-1 accuracy.

Chapter 2 : Problem Definition

This chapter defines the research problem addressed by the project, the objectives, deliverables, and operational environment. It also explains the proposed system architecture and assumptions related to the development and deployment of the project.

2.1. Problem Statement

In traditional CCTV-based monitoring, once a vehicle leaves the field of view of one camera, it becomes nearly impossible for operators to trace its reappearance in other cameras manually. This limitation significantly reduces the effectiveness of current surveillance networks in scenarios such as:

- Locating stolen or suspicious vehicles.
- Investigating hit-and-run cases.
- Monitoring traffic violations and congestion patterns.
- Supporting law enforcement in identifying vehicles across wide camera networks.

Conventional systems rely on human operators, who must manually review hours of video footage from different cameras, a process that is inefficient, error-prone, and impractical for large-scale city deployments.

The proposed system addresses these challenges by integrating deep learning, attention mechanisms, and semantic attribute extraction to achieve robust cross-camera vehicle recognition. By automatically extracting visual and semantic vehicle features (e.g., color, brand, body type, and logo), the system learns to match the same vehicle across multiple views, even under changes in illumination, camera angles, or partial occlusions.

Ultimately, the objective is to provide law enforcement agencies, traffic management authorities, and smart city systems with a scalable, accurate, and deployable AI-based solution for intelligent surveillance and real-time vehicle tracking.

2.2. Deliverables and Development Requirements

The scope of this project encompasses the design, development, and implementation of an end-to-end deep learning-based vehicle re-identification system that automates the entire surveillance analysis pipeline — from vehicle detection to trajectory visualization.

The system will feature two main user roles:

- Administrator: responsible for managing datasets, retraining models, configuring parameters, and monitoring overall system performance.
- Surveillance Operator (User): interacts with the graphical dashboard to upload videos, perform vehicle queries, visualize matches, and analyze results.

1. Project Deliverables

The proposed system will deliver the following functional and tangible outputs:

1. Vehicle Detection Module
    a. Detect vehicles in live or recorded video streams using pre-trained deep learning models such as YOLOv8, Faster R-CNN, or SSD.
    b. Output: Bounding boxes identifying each detected vehicle.
2. Feature Extraction Module
    a. Extract discriminative deep features using CNN and Vision Transformer (ViT) architectures.
    b. Features include color, shape, body type, logo, and texture representations.
    c. Output: High-dimensional feature embeddings for each detected vehicle.
3. Semantic Attribute Recognition
    a. Classify high-level attributes (e.g., color, make, model, and vehicle type) through an auxiliary classification network.
    b. These attributes are later fused with visual embeddings for improved re-identification accuracy.
4. Vehicle Re-Identification Across Cameras
    a. Compare query vehicle embeddings with stored embeddings in the database.
    b. Use distance metrics (e.g., cosine similarity, triplet loss) to find top-k matching vehicles across non-overlapping cameras.
5. Trajectory Analysis Module
    a. Track vehicles across multiple cameras and generate spatio-temporal trajectories.
    b. Identify suspicious movements such as illegal U-turns, restricted zone entries, or abnormal repetitive paths.
6. Anomaly Detection Module
    a. Automatically detect anomalies in vehicle behavior based on trajectory data and predefined movement rules.
    b. Supports proactive alert generation for suspicious activity.
7. Performance Evaluation Module
    a. Administrators can evaluate system performance using metrics such as mean Average Precision (mAP), Rank-1, and CMC curves.
8. Database Integration

    a. All features, metadata, and re-identification results are stored in a MongoDB or PostgreSQL database for quick retrieval and scalability.
9. Documentation and Deployment Guide
    a. A detailed report, model documentation, and deployment guide (for both local and cloud environments) will be included as part of the final deliverables.

2.3. Current System (if applicable to your project)

The overall system architecture of the proposed vehicle re-identification framework is based on modular deep learning components integrated into a unified pipeline.

Key Modules of the Architecture:

1. Input and Preprocessing Module
    a. Accepts video streams or images from multiple surveillance cameras.
    b. Performs preprocessing (resizing, normalization, augmentation) to standardize inputs.
2. Detection and Feature Extraction Module
    a. Uses deep CNN (e.g., ResNet-IBN) or hybrid CNN-Transformer models for visual feature extraction.
    b. Embeds extracted features into a high-dimensional vector space for comparison.
3. Semantic Attribute Module
    a. Classifies vehicle attributes (color, model, type) using an auxiliary branch.
    b. Combines semantic information with visual embeddings to improve matching accuracy.
4. Re-Identification and Matching Module
    a. Matches query vehicle features with gallery database using distance metrics (e.g., cosine similarity, triplet loss).
    b. Returns top-k matching results with similarity scores.
5. Trajectory and Anomaly Analysis Module
    a. Tracks vehicles across camera networks using spatial and temporal association.
    b. Detects abnormal movements such as illegal U-turns or entry into restricted areas.
6. Evaluation and Visualization Module
    a. Displays results in a dashboard: query image, top retrieved images, similarity heatmaps, and statistical performance graphs.

Chapter 3: Requirement Analysis

Software Requirements Specification (SRS) report should be included in this chapter.

3.1. Use Cases

Use cases are a widely used and highly regarded format for capturing requirements. Before writing functional requirement use cases can help you to understand the requirements in the way users expect. The following table presents you not only the template to write use case(s) as well as guides you to write each section with examples.

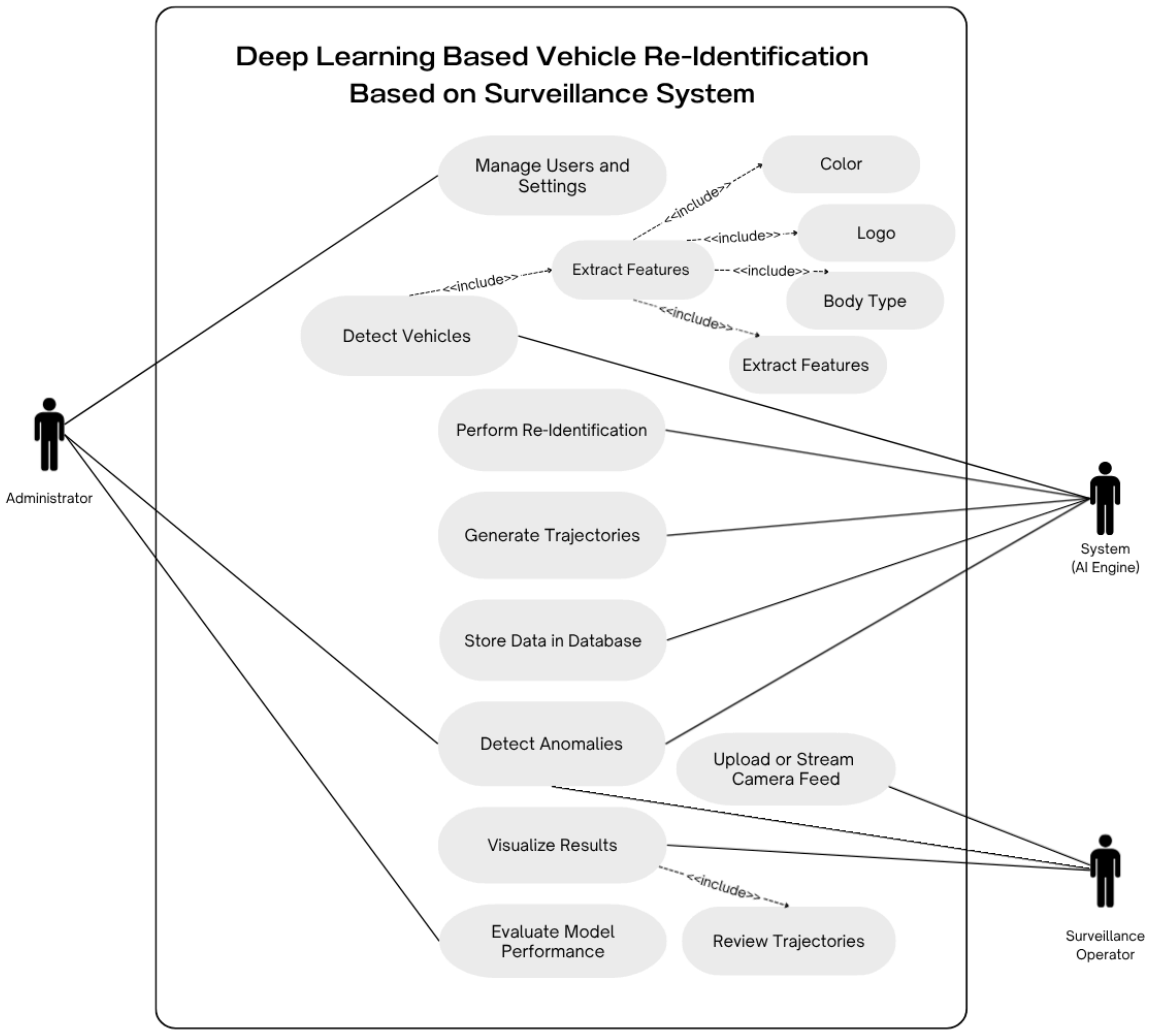

Figure 3.1 Use Case

Table 3.1: Use Case of Detect Vehicles

Use Case ID:	UC-01
Use Case Name:	Detect Vehicles
Actors:	1.System (AI Engine), 2.Administrator
Description:	The system automatically detects vehicles from live surveillance camera feeds or uploaded videos using deep learning-based object detection models.
Trigger:	Camera feed or video input is received by the system.
Preconditions:	Cameras are connected and calibrated; system is running and models are loaded..
Postconditions:	Vehicles in the scene are detected and cropped for further processing.
Normal Flow:	1. System receives video feed. 2. Frame is processed using detection model. 3. Bounding boxes are drawn around detected vehicles. 4. Cropped images are sent to feature extraction module. 5. Detection results stored in database.
Alternative Flows:	A1: If frame quality is low, the system requests next frame. A2: If multiple vehicles overlap, detection confidence is recalculated.
Exceptions:	E1: Camera feed loss. E2: Low light condition causes missed detections..
Includes:	Extract Features (UC-02)
Special Requirements:	Real-time detection at ≥25 FPS; minimal false positives.
Assumptions:	Cameras have stable power and network.
Notes and Issues:	System must adapt to various weather and lighting conditions.

Table 3.2: Use Case of Extract Features

Use Case ID:	UC-02
Use Case Name:	Extract Features
Actors:	System (AI Engine)
Description:	The system extracts distinguishing visual features of vehicles, such as color, logo, and body type, using a deep CNN or transformer-based model.
Trigger:	A detected vehicle image is available.
Preconditions:	Detection completed; feature extraction model loaded.
Postconditions:	Vehicle features (embeddings) stored in feature database.
Normal Flow:	1. System receives cropped vehicle image. 2. Model extracts visual features. 3. Features converted into embeddings. 4. Data stored in database for re-identification.
Alternative Flows:	A1: Low-quality images lead to partial feature extraction.
Exceptions:	E1: Model inference error. E2: Missing input frame.
Includes:	Color, Logo, Body Type
Special Requirements:	Feature extraction model must maintain consistency under different lighting.
Assumptions:	Trained model is available.
Notes and Issues:	Accuracy directly impacts re-identification performance.

Table 3.3: Use Case of Perform Vehicle Re-Identification

Use Case ID:	UC-03
Use Case Name:	Perform Vehicle Re-Identification
Actors:	System (AI Engine) Administrator Surveillance Operator
Description:	The system re-identifies vehicles across multiple camera views using similarity matching of extracted features.
Trigger:	New vehicle features are available for comparison.
Preconditions:	Feature extraction complete; database accessible.
Postconditions:	Vehicle ID assigned or updated; re-identification results saved.
Normal Flow:	1. System compares new features with existing ones. 2. Computes similarity scores. 3. Determines match or new ID. 4. Updates trajectory and database. 5. Sends results for visualization.
Alternative Flows:	A1: If multiple candidates found, ranked list displayed for review.
Exceptions:	E1: Database query timeout. E2: Low similarity threshold.
Includes:	Detect Vehicles (UC-01), Extract Features (UC-02)
Special Requirements:	None
Assumptions:	Database indexed and optimized for similarity search.
Notes and Issues:	Model retraining may be needed periodically.

Table 3.4: Use Case of Generate Trajectories

Use Case ID:	UC-04
Use Case Name:	Generate Trajectories
Actors:	System (AI Engine) Surveillance Operator
Description:	The system generates movement trajectories of vehicles based on camera timestamps and geolocation data.
Trigger:	Re-identification results are updated.
Preconditions:	Re-identification completed; time and location data available.
Postconditions:	Vehicle trajectories stored and visualized.
Normal Flow:	1. The system collects vehicle IDs and timestamps. 2. Merges sequential camera views. 3. Plots trajectory paths. 4. Stores in the database and sends for visualization.
Alternative Flows:	A1: Missing timestamps → system interpolates trajectory.
Exceptions:	E1: GPS or camera sync error.
Includes:	Store Data in Database (UC-05)
Special Requirements:	Accurate time synchronization across cameras.
Assumptions:	Cameras have synchronized clocks.
Notes and Issues:	Inaccurate timestamps may distort trajectories.

Table 3.4: Use Case of Store Data in Database

Use Case ID:	UC-05
Use Case Name:	Store Data in Database
Actors:	System (AI Engine)
Description:	The system securely stores detected vehicle data, features, and trajectories into a centralized database.
Trigger:	New vehicle or trajectory information available.
Preconditions:	Database connection established.
Postconditions:	Data Successfully saved and indexed
Normal Flow:	1. System packages vehicle metadata. 2. Sends data to database. 3. Confirms write operation. 4. Makes data retrievable for queries.
Alternative Flows:	A1: Duplicate records merged.
Exceptions:	E1: Database write failure.
Includes:	None
Special Requirements:	Database encryption and indexing for performance.
Assumptions:	Database capacity sufficient.
Notes and Issues:	Data backup and security essential.

Table 3.6: Use Case of Detect Anomalies

Use Case ID:	UC-06
Use Case Name:	Detect Anomalies
Actors:	System (AI Engine), Administrator
Description:	The system identifies unusual vehicle behaviors or routes using trajectory and re-identification data.
Trigger:	Anomalous data pattern detected.
Preconditions:	Historical data available for comparison.
Postconditions:	Anomaly alerts generated.
Normal Flow:	1. System analyzes trajectories. 2. Detects abnormal speed or direction. 3. Generates alert for operator. 4. Logs anomaly for further review.
Alternative Flows:	A1: Operator manually marks anomaly as false positive.
Exceptions:	E1: Insufficient data for comparison.
Includes:	Generate Trajectories (UC-04)
Special Requirements:	Real-time alerting system.
Assumptions:	Anomaly detection model trained and deployed.
Notes and Issues:	Threshold tuning affects false alarm rate.

Table 3.7: Use Case of Visualize Results

Use Case ID:	UC-07
Use Case Name:	Visualize Results
Actors:	Surveillance Operator, Administrator
Description:	The system displays re-identification results, trajectories, and anomalies through an interactive dashboard.
Trigger:	Operator opens dashboard or new results are available.
Preconditions:	Re-identification and trajectory data available.
Postconditions:	Results displayed and available for user analysis.
Normal Flow:	1. User logs in to dashboard. 2. System retrieves data from database. 3. Displays vehicle info, path, and similarity scores. 4. User reviews and exports data.
Alternative Flows:	A1: User filters results by date or camera
Exceptions:	E1: Dashboard connection timeout.
Includes:	Review Trajectories
Special Requirements:	Intuitive UI with minimal latency.
Assumptions:	User permissions properly set.
Notes and Issues:	Dashboard updates should be near real-time.

Table 3.8: Use Case of Evaluate Model Performance

Use Case ID:	Enter a unique numeric identifier for the Use Case. e.g. UC-1.2.1
Use Case Name:	Evaluate Model Performance
Actors:	Administrator
Description:	The administrator evaluates detection and re-identification models using performance metrics such as accuracy, precision, and recall.
Trigger:	Model evaluation initiated by administrator.
Preconditions:	Test dataset and ground truth labels available.
Postconditions:	Evaluation report generated and stored.
Normal Flow:	1. Admin selects evaluation dataset. 2. System runs inference. 3. Calculates metrics. 4. Displays performance summary. 5. Logs results for record-keeping.
Alternative Flows:	A1: Admin adjusts thresholds and re-runs evaluation.
Exceptions:	E1: Missing test data.
Includes:	None
Special Requirements:	Support for multiple evaluation metrics.
Assumptions:	System resources available for batch processing.
Notes and Issues:	Periodic evaluations maintain model accuracy over time.

3.2. Functional Requirements

FR-01: Vehicle Detection

The system shall automatically detect vehicles from live surveillance streams or uploaded video files using a state-of-the-art deep learning detection model such as YOLOv8 or Faster R-CNN.
The detection model should be capable of accurately identifying vehicles of various types (cars, trucks, vans, motorcycles, etc.) under diverse environmental conditions including low light, rain, and shadows. This functionality ensures that every relevant vehicle in the camera frame is identified and tracked for further processing.

FR-02: Feature Extraction

After vehicle detection, the system shall extract visual and semantic features such as color, logo, body type, brand, and texture using deep CNN and Transformer-based architectures.
These extracted features are used to create a unique feature vector representation for each vehicle, allowing the system to differentiate between vehicles even with similar appearances. This module forms the core of the re-identification process, as it ensures discriminative and robust feature learning.

FR-03: Vehicle Re-Identification

The system shall perform cross-camera vehicle matching by comparing extracted feature embeddings with previously stored vehicle data.
Using similarity measurement techniques such as cosine similarity or triplet loss, the system identifies whether a vehicle appearing in one camera has been seen before in another. This function enables operators to trace the movement of a particular vehicle across multiple non-overlapping camera views, supporting crime investigations and surveillance tracking.

FR-04: Data Storage

The system shall maintain a centralized database for storing all detected vehicle information, including extracted features, timestamps, and trajectory metadata.
The database must be optimized for fast retrieval and support queries based on various attributes such as vehicle ID, time, and camera location. This ensures that data is preserved for long-term analysis and comparison.

FR-05: Trajectory Generation

The system shall automatically generate vehicle movement trajectories by linking detections across multiple cameras and timestamps using spatio-temporal association algorithms.
This allows the system to map the exact path a vehicle takes through different camera networks, enabling tracking and behavior analysis across large surveillance zones.

FR-06: Anomaly Detection

The system shall detect abnormal or suspicious vehicle behavior using trajectory data and learned motion patterns.
Examples of anomalies include wrong-way driving, entry into restricted zones, unusual stopping behavior, or repetitive movement patterns. The system must trigger alerts when such deviations are observed to assist authorities in real-time decision-making.

FR-07: Camera Feed Handling

The system shall support both real-time streaming from live IP cameras and manual video uploads for offline analysis.
It should handle multiple concurrent video feeds, automatically manage buffering, and recover from network interruptions without losing important footage.

FR-08: Visualization Dashboard

The system shall include an interactive web-based dashboard for both administrators and surveillance operators.
This dashboard will display detected vehicles, re-identification results, similarity scores, and trajectory visualizations in an intuitive interface.
The visualization panel should include filter and search features to allow users to analyze data by time, location, or vehicle type.

FR-09: User Management

The system shall implement a role-based access control mechanism.
Administrators will be able to create, update, or remove user accounts, while surveillance operators will have limited access based on assigned privileges.
This ensures data security and prevents unauthorized usage of sensitive features or data.

FR-10: Model Evaluation

The system shall compute and display performance metrics such as accuracy, precision, recall, F1-score, mean Average Precision (mAP), and Rank-1 identification rate to evaluate model efficiency.
This feature allows administrators to analyze system performance and make informed decisions about retraining or optimizing the model.

FR-11: Alert and Notification System

The system shall generate real-time alerts and notifications when anomalies or suspicious vehicles are detected.
These alerts should be displayed on the dashboard and optionally sent via SMS, email, or push notifications. This feature enhances situational awareness for law enforcement or surveillance staff.

FR-12: Data Retrieval and Filtering

The system shall allow users to query and retrieve stored vehicle data based on multiple parameters such as vehicle ID, timestamp, location, or camera source.
Advanced filtering options should also allow searches based on attributes (e.g., color = red, type = SUV) for quick access to relevant footage.

FR-13: Logging and Auditing

The system shall maintain a detailed activity log of all user actions, system operations, and re-identification results.
Logs will help in auditing user activities, diagnosing issues, and tracking security breaches if they occur.
This ensures accountability and supports post-event investigations.

FR-14: System Configuration

The system shall allow the Administrator to configure operational parameters, including similarity thresholds, alert sensitivity levels, logging preferences, and data retention settings.
These adjustable parameters ensure flexibility and optimal performance under various deployment environments.

3.3. Functional Requirements

1. Performance

The system should provide real-time or near-real-time vehicle re-identification with minimal delay.The average processing time for one vehicle query should not exceed one second when executed on GPU-enabled hardware.
Optimized models, caching mechanisms, and efficient database queries should be used to maintain high throughput even under heavy workloads.

2. Reliability

The system must operate continuously and reliably for 24/7 surveillance without performance degradation.It should automatically handle unexpected camera disconnections, frame losses, or hardware failures gracefully, ensuring uptime above 99%.
All critical components must have error recovery and restart mechanisms.

3. Usability

The user interface should be intuitive, visually appealing, and accessible to non-technical users such as surveillance officers or police staff.The dashboard must include simple navigation, well-labeled buttons, and consistent color schemes for functions like Upload, Query, and Visualize Results.Proper feedback messages and tooltips should guide the user throughout each process.

4. Scalability

The architecture should be modular and scalable, capable of integrating additional cameras, data sources, and processing nodes as the surveillance network expands.
It must be able to handle at least 50 or more camera streams simultaneously without requiring significant architectural redesign.

5. Security

The system must ensure data confidentiality, integrity, and access control.All sensitive information, including login credentials, should be encrypted.Access should be restricted based on user roles (Administrator or Operator), and all API endpoints must be secured against unauthorized access.

6. Maintainability

The software must be designed using modular programming principles and managed through a version control system (Git/GitHub).Each module (detection, re-identification, dashboard, database) should be independently testable, maintainable, and upgradable without affecting the entire system.Clear documentation and comments must accompany all code for future maintenance.

7. Portability

The system must be easily deployable on multiple environments such as Windows, Linux, or cloud-based servers (AWS, Google Cloud, etc.).It should not depend on platform-specific configurations.All deployment scripts and setup documentation must be provided for seamless installation on different operating systems.

8. Supportability

The system should support model retraining and dataset updates without needing to rebuild the entire pipeline.Administrators should be able to load new datasets, fine-tune the model, and redeploy updates while preserving existing configurations.

9. Efficiency

The system must efficiently use available hardware resources like GPU, CPU, and RAM.
Techniques such as model pruning, quantization, and optimized inference should be applied to minimize computational overhead.Memory management must ensure stable operation even during high-load processing.

10. Availability

The system should be available for use at all times during surveillance operations.Automated service monitoring and restart mechanisms should ensure continuous operation even after unexpected crashes.Database backups and redundancy measures must be implemented to prevent data loss.

11. Accuracy

The system's vehicle re-identification model should achieve at least 90% Rank-1 accuracy on benchmark datasets such as VeRi-776 or VehicleID.The model should also maintain consistent accuracy across real-world camera environments, ensuring dependable identification for operational use.