Shailender Kumar

MTech_Thesis11

paper

Document Details

Submission ID
trn:oid:::27535:140817047

Submission Date
May 28, 2026, 8:54 PM GMT+5:30

Download Date
May 28, 2026, 8:56 PM GMT+5:30

File Name
MTech_Thesis11.pdf

File Size
5.5 MB

63 Pages

17,885 Words

99,729 Characters

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

89 Not Cited or Quoted 7%
Matches with neither in-text citation nor quotation marks

1 Missing Quotations 0%
Matches that are still very similar to source material

7 Missing Citation 1%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
4% Publications
7% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **89** Not Cited or Quoted 7%
  Matches with neither in-text citation nor quotation marks
- **1** Missing Quotations 0%
  Matches that are still very similar to source material
- **7** Missing Citation 1%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 6% Internet sources
- 4% Publications
- 7% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	arxiv.org 1%
2	Internet	dspace.dtu.ac.in:8080 <1%
3	Student papers	Kurukshetra University on 2026-05-26 <1%
4	Student papers	Information Technology on 2026-04-23 <1%
5	Student papers	Delhi Technological University on 2025-10-30 <1%
6	Internet	www.mdpi.com <1%
7	Internet	www.arxiv-vanity.com <1%
8	Internet	repository.tudelft.nl <1%
9	Student papers	Delhi Technological University on 2019-06-30 <1%
10	Student papers	Delhi Technological University on 2026-05-27 <1%

11 Internet
isjem.com <1%

12 Internet
artemis.cslab.ece.ntua.gr:8080 <1%

13 Internet
hal.science <1%

14 Internet
www.ijert.org <1%

15 Internet
thesesjournal.com <1%

16 Student papers
Liverpool John Moores University on 2022-12-29 <1%

17 Internet
webarchive.ucr.edu <1%

18 Internet
123dok.net <1%

19 Student papers
Liverpool John Moores University on 2023-03-15 <1%

20 Internet
infinitylearn.com <1%

21 Internet
psasir.upm.edu.my <1%

22 Internet
www.coursehero.com <1%

23 Student papers
Sri Lanka Institute of Information Technology on 2022-12-21 <1%

24 Student papers
Yeditepe University on 2025-06-13 <1%

25 Internet
ar5iv.labs.arxiv.org <1%

26 Internet
jultika.oulu.fi <1%

27 Student papers
University of Technology on 2025-02-12 <1%

28 Student papers
University of Wollongong on 2023-12-07 <1%

29 Student papers
University of Wollongong on 2023-12-07 <1%

30 Internet
mdpi-res.com <1%

31 Publication
Wang, Xuan. "Context in Computer Vision: A Taxonomy, Multi-stage Integration, ... <1%

32 Internet
deepai.org <1%

33 Internet
dspace.thapar.edu:8080 <1%

34 Internet
ijnrd.org <1%

35 Student papers
Imperial College of Science, Technology and Medicine on 2019-09-13 <1%

36 Student papers
University of Malakand, Chakdara on 2026-05-11 <1%

37 Internet
calvin-vision.net <1%

38 Internet
ijarcs.info <1%

39 Internet

repository.up.ac.za <1%

40 Student papers

Brunel University on 2026-03-31 <1%

41 Student papers

Coventry University on 2023-09-06 <1%

42 Publication

Paolo Ferro, Harinadh Vemanaboina, Chander Prakash. "Computational Techniqu... <1%

43 Publication

Siddhartha Roy, Soumya Sen, Agostino Cortesi. "Intelligent Systems - Emerging Tr... <1%

44 Student papers

Universidad Carlos III de Madrid - EUR on 2026-05-27 <1%

45 Student papers

University College London on 2023-08-29 <1%

46 Student papers

University of East London on 2026-05-08 <1%

47 Internet

bip.put.poznan.pl <1%

48 Internet

bura.brunel.ac.uk <1%

49 Internet

www.aimspress.com <1%

50 Internet

www.ijprems.com <1%

51 Student papers

AUT University on 2026-05-26 <1%

52 Student papers

Birla Institute of Technology and Science Pilani on 2023-05-30 <1%

53 Student papers
Delhi Technological University on 2026-05-26 <1%

54 Student papers
FICT on 2026-05-03 <1%

55 Student papers
Indiana University on 2025-12-18 <1%

56 Student papers
Kingston University on 2026-01-19 <1%

57 Student papers
National Institute of Technology, Rourkela on 2026-05-11 <1%

58 Student papers
Panipat Institute of Engineering & Technology on 2026-04-22 <1%

59 Student papers
The Northcap University on 2026-05-22 <1%

60 Student papers
UC, Boulder on 2022-04-15 <1%

61 Student papers
University of Lincoln on 2026-04-29 <1%

62 Student papers
University of Ulster on 2026-05-05 <1%

63 Publication
Will Slocombe, Genevieve Liveley. "The Routledge Handbook of AI and Literature"... <1%

64 Publication
Zamanidoost, Yadollah. "Early-Stage Lung Cancer Detection Using Deep Learning ... <1%

65 Internet
assets.researchsquare.com <1%

66 Internet
builtin.com <1%

67 Internet
conservancy.umn.edu <1%

68 Internet
iris.unimo.it <1%

69 Internet
ojs.aaai.org <1%

70 Publication
"Computer Vision – ECCV 2016", Springer Science and Business Media LLC, 2016 <1%

71 Student papers
Loyola University, Chicago on 2026-04-26 <1%

72 Student papers
Vardhaman College of Engineering, Hyderabad on 2026-03-25 <1%

73 Student papers
Information Technology on 2026-04-24 <1%

74 Publication
Li, Xianhang. "Compute-Efficient Scaling of Fully-Open Visual Encoders.", Universi... <1%

75 Student papers
Napier University on 2023-08-16 <1%

76 Publication
Yu, Zhongzhi. "Enhancing Foundation Models with Self-Guided Techniques: From ... <1%

MULTIMODAL SURVEILLANCE SYSTEM USING OBJECT DETECTION AND LARGE LANGUAGE MODELS

10

THEESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by :

RISHABH SINGH

(24/RCO/04)

Under the Supervision of

Prof. SHAILENDRA KUMAR

TO THE

3

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Rishabh Singh, Roll No. 24/RCO/04, student of M.Tech (Artificial Intelligence), hereby declare that the project dissertation titled "Multimodal Surveillance System Using Object Detection and Large Language Models" submitted by me to the Department of Computer Science, Delhi Technological University, Delhi, in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Associateship, Fellowship or any other similar title or recognition.

Place: Delhi Rishabh Singh

Date: _____

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

Signature of Supervisor **Signature of External Examiner**

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE BY SUPERVISOR

I hereby certify that the project dissertation titled "**Multimodal Surveillance System Using Object Detection and Large Language Models**" submitted by **Rishabh Singh, Roll No. 24/RCO/04**, **COMPUTER SCIENCE AND ENGINEERING** , Delhi Technological University, Delhi, in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date: _____

Prof. Shailendra Kumar

SUPERVISOR

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to Prof. Shailendra Kumar, Department of Computer Science, Delhi Technological University, for his continuous guidance, encouragement and technical mentorship throughout this work. His suggestions helped shape the project from a conventional object detection system into an explainable multimodal surveillance framework. I appreciate the support provided by the University, including the laboratories, infrastructure, testing facilities, and environment that enabled me to work without interruption. I thank my lab assistants, seniors, and peer group for their aid and knowledge on various subjects. I am also grateful to everyone who directly or indirectly contributed to the completion of this project, and my parents and well-wishers for their constant support, belief in me, and encouragement.

Rishabh Singh

24/RCO/04

Abstract

Intelligent surveillance is now an essential component of traffic management, public safety, and institutional security. Conventional CCTV installations mainly preserve visual evidence, while recent deep-learning solutions are able to locate objects in video frames. Even so, many existing systems remain limited to class labels and bounding boxes. They generally do not explain the situational meaning of a detected scene, retrieve relevant safety knowledge, or convert visual observations into practical natural-language alerts for human operators.

This work develops an Explainable Intelligent Surveillance System that combines real-time visual perception with language-based event interpretation. The framework uses YOLOv8 to detect surveillance-relevant objects, ByteTrack to preserve object identities across frames, a rule-guided event engine to estimate collision and crowd risks, FAISS to perform vector retrieval, LangChain to coordinate retrieval and prompting, and GPT-4/LLaMA-style models to generate explanations. Each CCTV frame is transformed into a structured event state that records object classes, bounding-box coordinates, confidence values, persistent track IDs, velocity estimates, proximity relations, and risk scores. The event state is embedded and searched against a domain knowledge base containing traffic rules, safety procedures, and surveillance-event descriptions. Retrieved evidence is then combined with the structured visual evidence in a constrained prompt to produce a JSON-style situation report and a concise operator-facing explanation.

The evaluation uses public surveillance datasets together with custom traffic CCTV clips. The YOLOv8x configuration provides reliable detection for person and vehicle classes while maintaining a frame rate appropriate for real-time monitoring. The event-analysis module identifies collision risk, bus-stop congestion, unusual stopping, and crowd-density conditions. The retrieval layer improves the factual grounding of generated explanations, while the dashboard brings together live video, event metrics, retrieved evidence, and language-model output in a unified interface. Overall, the study shows that surveillance platforms can progress beyond passive detection toward explainable, auditable, and context-aware decision support.

72

Keywords: Intelligent Surveillance, YOLOv8, ByteTrack, Object Detection, Large Language Models, Retrieval-Augmented Generation, FAISS, LangChain, Explainable AI, Event Analysis, Computer Vision.

Contents

57

List of Tables

List of Figures

List of Symbols/Abbreviation

Symbol	Meaning
AI	Artificial Intelligence
CCTV	Closed-Circuit Television
CNN	Convolutional Neural Network
CV	Computer Vision
FAISS	Facebook AI Similarity Search
FPS	Frames Per Second
GPU	Graphics Processing Unit
IoU	Intersection over Union
LLM	Large Language Model
mAP	Mean Average Precision
NMS	Non-Maximum Suppression
OpenCV	Open Source Computer Vision Library
RAG	Retrieval-Augmented Generation
TTC	Time-To-Collision
UI	User Interface
VIRAT	Video and Image Retrieval and Analysis Tool
YOLO	You Only Look Once
XAI	Explainable Artificial Intelligence

48

Chapter 1 INTRODUCTION

1.1 Overview

Many video surveillance installations are located in today's road intersections, corridors and public transport platforms, industrial facilities and commercial facilities. There's a lot of information that an operator can't constantly view on all of the "streams" because of the high resolution CCTV cameras. As a result, there is a disconnect between data gathering and understanding the situation. Partially this can be solved by applying deep learning based object detection to recognize objects (pedestrians, vehicles and others) in each frame, but this alone would not explain what is risky in a scene or what the response should be.

71

This is the motivation behind proposing this work: the combination of computer vision and language-based reasoning to overcome this limit. Their goal is to use YOLOv8 and ByteTrack to transform raw surveillance frames into structured visual evidence and input the visual evidence into an event analysis layer to produce risk estimation. A retrieval-augmentated generation pipeline provides external knowledge to a large language model for generating a concise explanation based on the outcomes of the detection and retrieved safety context. This kind of output is not only a word balloon, but it's also a situation report.

The pipeline in the proposed Explainable Intelligent Surveillance System, is to be modular. Each module has a clear purpose: The video ingestion is for extracting video frames, the YOLOv8 for object detection, the ByteTrack is for assigning the ID of the objects throughout the video, the event engine is for computing the risk indicators, the FAISS is for retrieving the knowledge from each frame, the LangChain is for structuring the prompt to the LLM, the LLM is for reasoning with the knowledge retrieved by the FAISS, and the dashboard is for presenting the result to the operator for use. This improves the auditability as each generated alert can be traced back to Object IDs, frames, risk scores and retrieved text.

1.1.1 Motivation

The motivation for this study is based on the experience of normal surveillance rooms. Typically, the operators have to handle dozens of camera feeds and few display panels and limited time. Events like pedestrian encroachment, bus-stop congestion, wrong-way movement, or a near collision, may be present for a short duration. It may not activate an alert and give an explanation of how the event is related to early intervention.

The success of object detection systems today is high, but the output is still low-level. You can never determine if a “person 0.91” box is a person going into the danger zone, a vehicle coming or

if the event needs to be escalated. This means that an object doesn't have to have a relationship with the other, and a surveillance platform should be able to think about relationships, not just objects. The authors of the present work explore how to use LLM after a computer vision model to gain insight into structured evidence to produce human-readable summaries.

The message should be clear in any safety critical environment, like a nuclear plant. The operator must know if a system has identified an event as high risk, either due to proximity, time to collision or crowd density or a mixture of these. This can be done by using retrieval-augmented generation, which uses a knowledge base for retrieval instead of free-form model generation for language results .

1.2 Problem Statement

It's not only hard to find the objects, it's hard to understand the relationship between the objects. There can be numerous objects detected in a surveillance scene, but only a portion of these objects contribute to risk. The structure should be designed so that the tracked objects can be identified as near, approaching, normal or not as well as if the tracked situation can be classified as a known risk pattern. These include temporal association, metric computation and knowledge-grounded reasoning.

The latency of LLM inference increases on top of that, as can be inference and retrieval. Architecture proposed: Make use of the RAG layer and LLM layer only in certain cases: when the EVP has a high risk value; when the operator requests the module to explain the EVP. This selective triggering will prevent and avoid the generation of routine frames in the transmission, while maintaining the understanding of the critical events.

The problem is then translated into an engineering and research problem to be solved – to design a surveillance system with modular components, which produce interesting intermediate results and make the entire system fast enough to be used in real time. The system should not make unsupported claims; keep the evidence anonymous when reporting on the evidence; report the evidence in a way that human operators can understand and verify.

Passive CCTV surveillance is the traditional way of monitoring. Even the most basic analytics-enabled ones will be able to detect motion or count objects, but are unable to interpret the events as they occur. This results in three technical issues. The first is that the system doesn't give responses which can be easily interpreted by the surveillance operators. Secondly, it lacks temporal identity association, meaning that repeated detections of persistent objects are hard to identify. Third, it is unable to integrate the visual data with other information, like traffic safety limits, rules of crowd management.

This research problem is formulated as follows: Given the following components, how to integrate them into a real-time surveillance pipeline for generating explainable and actionable alerts: object detection, object tracking, event-risk computation, knowledge retrieval and language-model reasoning. The solution should be a modular solution, easily solvable, reproducible and suitable for the traffic and public space surveillance application.

1.3 Objectives

- Develop an Object Detection Module for real-time object detection of surveillance related classes: Person, Car, Bus, Truck, Motorcycle and Bicycle using YOLOv8.
- To locate objects from frame to frame for motion based event analysis by ByteTrack.
- Developed a system to process events and calculate distance, speed, time to collision, crowd density and stationary object measurements.
- To create a pipeline for Retrieval Augmented Generation (RAG) using sentence embeddings, FAISS vector search and LangChain prompt orchestration.
- To incorporate the GPT-4/LLaMA style reasoning in structured situation reports and risk levels and recommended operator actions.
- To create an explainable dashboard to show real-time video, detection overlays, context retrieved, risk scores, and video language explanations.
- The accuracy of the developed framework for detection, FPS, latency, retrieval quality, explanation quality and alert accuracy were evaluated.

1.4 Scope of Work

This study only addresses those aspects of video-based surveillance for outdoor traffic and video monitoring in public areas. It accepts streams from video cameras or video files and operates on video frames at a fixed resolution which is appropriate for inference using YOLOv8. The study concentrates on the categories of objects of interest for public safety such as pedestrians, cars, buses, trucks, two-wheelers, bicycles. Does not contain face recognition, biometric identification or profiling of persons.

The underlying object counts, risk scores and retrieved snippets are now as a result always included in high-risk alerts to allow the operator to confirm the reasoning behind the alert prior to escalation.

1.5 Contributions

This paper focuses on those elements of video-based surveillance of traffic that occurs outside buildings and on video monitoring of public space. It can handle video streams from video cameras or video files, and operates on video frames at a given resolution appropriate for the YOLOv8 inference. The study will be based on pedestrian, vehicle types (cars, buses, trucks and two-wheelers such as bicycles) objects of interest for public safety. Does NOT contain face recognition, biometric identification or biometric profiling of people.

It is a support system and not an operational system. The underlying object counts, risk scores and retrieved snippets are now as a result always included in high-risk alerts to allow the operator to confirm the reasoning behind the alert prior to escalation.

39

Chapter 2

LITERATURE REVIEW

2.1 Introduction

The review follows this principle that there are three levels of capabilities needed for intelligent surveillance. The first one is perception, in which objects and elements in the scene are detected from frames. The second level of reasoning is temporal reasoning, which is the ability to preserve identities, trajectories and interactions of objects over time. Most current studies have focused at the individual or community level and at the operational level, both are required.

62

The research also indicates transition from handcrafted image processing to deep learning and eventually, to language based reasoning. The aim of this proposed thesis is to combine both approaches by leveraging the deep detectors on their robustness to visual data and the explicit event metrics on their interpretability. The related literature throughout this work is: Object detection, multi-object tracking, video event analysis, retrieval-augmented generation, large language models and explainable AI.

This chapter takes the form of a technical survey by describing how each component has evolved, and what research gap is created when the components are combined. The review shows that while each module is well developed, the possible integration of all these modules to create a real-time explainable surveillance system remains a research challenge.

2.2 Traditional Surveillance Systems

In early surveillance systems background subtraction and frame differencing and optical flow were employed. The methods are low computational, but experienced illumination change, shadow, camera vibration and occlusion. Although the statistical classifiers were successful at object recognition, they were not able to model complex scenes. With the introduction of the deep convolutional network, automatic feature learning has been greatly improved, leading to significant advances in visual recognition.

The standard systems usually consist of alarm functions such as motion in a region of interest or a count of a number of objects. They are easy to use but don't have the intelligence of separating out benign and dangerous behaviour if necessary, which is missing in context. For example the motion of a vehicle in a lane on the edge of a road is safe, but that of a pedestrian entering the carriageway is risky. The limitation led researchers to use learned object detections and tracking methods.

Although deep learning has been introduced, most commercial surveillance systems are still detection-based. They can identify objects, e.g. cars, people, and the alert semantics are often based on the following simple rules: line crossing, zone entry. These systems don't automatically have the ability to provide such explanations as: Why was this event risky? / What objects made this event risky? / What action is recommended for this event? The main motivation behind event-state serialization and the generation of explanations with LLMs is this gap.

16 The research of object detection has turned from two-stage detector like R-CNN, Faster R-CNN to one-stage detector like SSD and YOLO. The importance of inference speed of the one-stage detectors in surveillance cannot be ignored. But object detectors don't natively describe event semantics. Likewise, anomaly detection techniques are able to identify anomalous temporal events, but mostly require that there be event classes and annotated videos.

In the last few years, LMs and RAGs have shown remarkable reasoning and summarization capability. Most RAG applications are application that require text query, but not structured output from the Computer Vision Pipeline. This work aims to bridge this void of one pipeline that maps detected objects and risk metrics to a knowledge-based natural language description.

2.3 Convolutional Neural Networks in Computer Vision

Surveillance: Convolutional features should be invariant to viewpoint, partial occlusion, illumination and scale differences. A bus may occupy a lot of space on the left hand side of the frame, and a pedestrian may only take up a little space on the right hand side of the frame. This variation can be addressed by the feature hierarchy of CNNs where shallow layers are capable of capturing spatial details, and deeper layers are able to capture semantic information. These are then further combined in multi-scale detection heads for small and large objects.

One example of the receptive field of a convolutional network is for road-scene understanding. A small receptive field may provide information about local features (edges, textures etc) and a larger receptive field may provide information about the structure of the object and its surrounding context. They are both needed for object detection. The detector not only has to localise the boundary of an object more or less precisely, but it must also recognize the type of the object, based on a wider shape and/or on context.

CNNs learn spatial features through the use of local kernels on an image grid. Early layers are generally responsible for detecting edges, road markings, contours of vehicles and pedestrians, and surveillance frame detectors, while later layers are more likely to detect parts of an object and patterns at the level of object categories. This hierarchy can aid because the distance between the camera and objects, and the size of objects in each surveillance area can be quite different.

Convolution of an input feature map X with a kernel W can be written as:

$$Y(i,j) = \text{sum}_m \text{sum}_n X(i+m, j+n)\, W(m,n) + b$$

It gives a response of convolution which is then passed through normalization and nonlinear activation functions, such as the ReLU. Down-sampling layers are layers with larger receptive fields, but result in less sensitivity to small objects. In modern detectors, therefore, semantic features are integrated with high resolution spatial features using feature pyramids. In surveillance, this is critical as someone could be in a tiny spot in the frame.

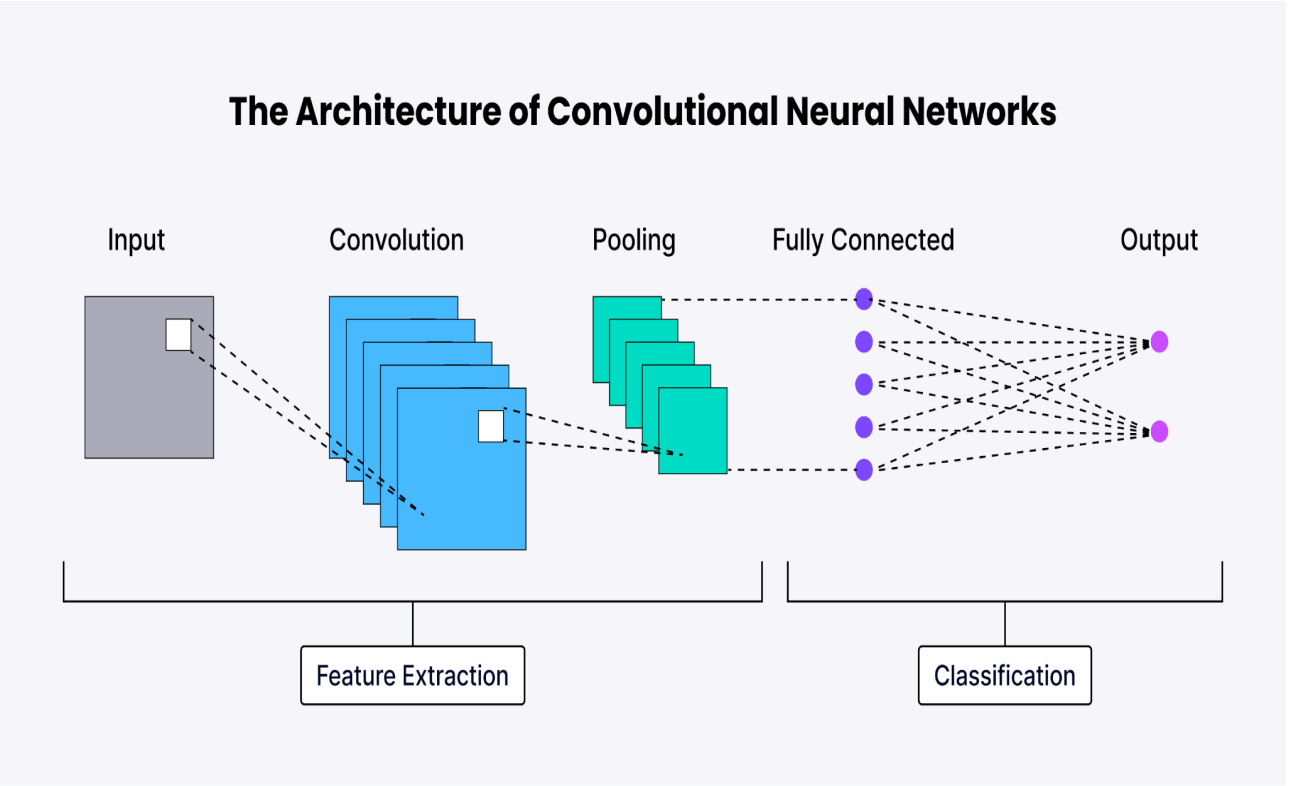


Figure 2.1: CNN feature extraction workflow for surveillance video frames.

2.4 Object Detection Models

The accuracy-slow vs fast-speed tradeoff – 2 stage detector vs one stage detector. Typically, the two-stage detectors are good localizers, since the region refinement step is relatively cheap, while the region proposal step is expensive. One-stage detectors do not require this intermediate step, and can thus be used in continuous video processing. Surveillance cameras can provide extended streams and multiple feeds, and might be an important constraint in the deployment for inference efficiency.

In this study, the ultimate goal is not the object detection itself, rather it's the generation of initial evidence. The detector should not be so sensitive that it will change its response to the incident too much to follow and analyze the events downstream. If no detections are made, a false track can result and if duplicate detections are made, false density and risk assessment can result. This affects the overall performance of the pipeline as a result, the detector configuration, the confidence threshold and the NMS are affecting the performance of the pipeline.

Object detection is a form of classification, and one that also predicts the location of the object. Two-stage detectors are detectors that suggest areas, after which detectors classify the areas. The two-stage methods are faster but slower as they treat the proposals by the regions independently. One-stage detectors turn detection tasks directly into a regression and classification task over dense feature maps, which is more convenient for real-time surveillance.

YOLO family is a major series of detectors that consist of one-stage detectors. YOLO makes all the predictions in a single forward pass: it predicts all the bounding boxes and all the class probabilities. It was improved with localization, multi-scale detection and stability of training later on. YOLOv8's decoupled head and efficient C2f module enable it to handle frames in real-time, making it ideal for a surveillance system.

2.5 YOLOv8 Architecture

The backbone network is leveraged to extract feature maps from the input frame and then the extracted feature maps are passed to the YOLOv8 head network for object detection. The C2f modules allow, in part, to reuse the features and perform efficient bottleneck transformations to enhance the gradient propagation. This is beneficial for surveillance, as the objects can be seen at various scales and under different occlusion. The high level semantic information is then fused with the low level spatial information to enable small pedestrian and large vehicle detection in the same frame in the neck.

A decoupled head which is used to divide the issue of classification and localization. Both classification and localisation are very precise spatial regression based on distinguishing features of categories of objects. These branches are split to make the task of YOLOv8 more feasible than a tightly coupled head. This anchor-free formulation also removes the requirement of a well-defined anchor box and enables the detector to be flexible to different camera positions and road configurations.

In the surveillance scenes, post process also plays a vital role as well. Boxes that are the same object and overlap will be removed by Non-Maximum Suppression. In scenes of heavy traffic, buses, cars and people can overlap and duplicate boxes can be used to indicate objects that are duplicated. A well defined IoU threshold will ensure that no redetection will occur and that all the legitimate objects that are in the vicinity are kept.

YOLOv8 is composed of three major components: backbone, neck and detection head. The backbone is in charge of extracting the features of the input frame. The neck is composed of

multi-scale features and features pyramids and path aggregation structure. The detection head predicts the classes of the object and the bounding box's coordinates at multiple resolutions. This design characteristic assists the model to detect vehicles and people from far away and large vehicles from nearby.

The C2f module supports the re-use of features and gradient flow. The decoupled detection head can be used to localize and classification simultaneously, the decoupling of localization and classification is also beneficial because different properties of features need to be localized and classified. Moreover, the model is portable to various camera geometries since it does not require handdesigned anchor boxes, which can be quite burdensome to generate.Additionally, YOLOv8 is not constrained to hand-designed anchor boxes, making it more portable to different camera geometries.

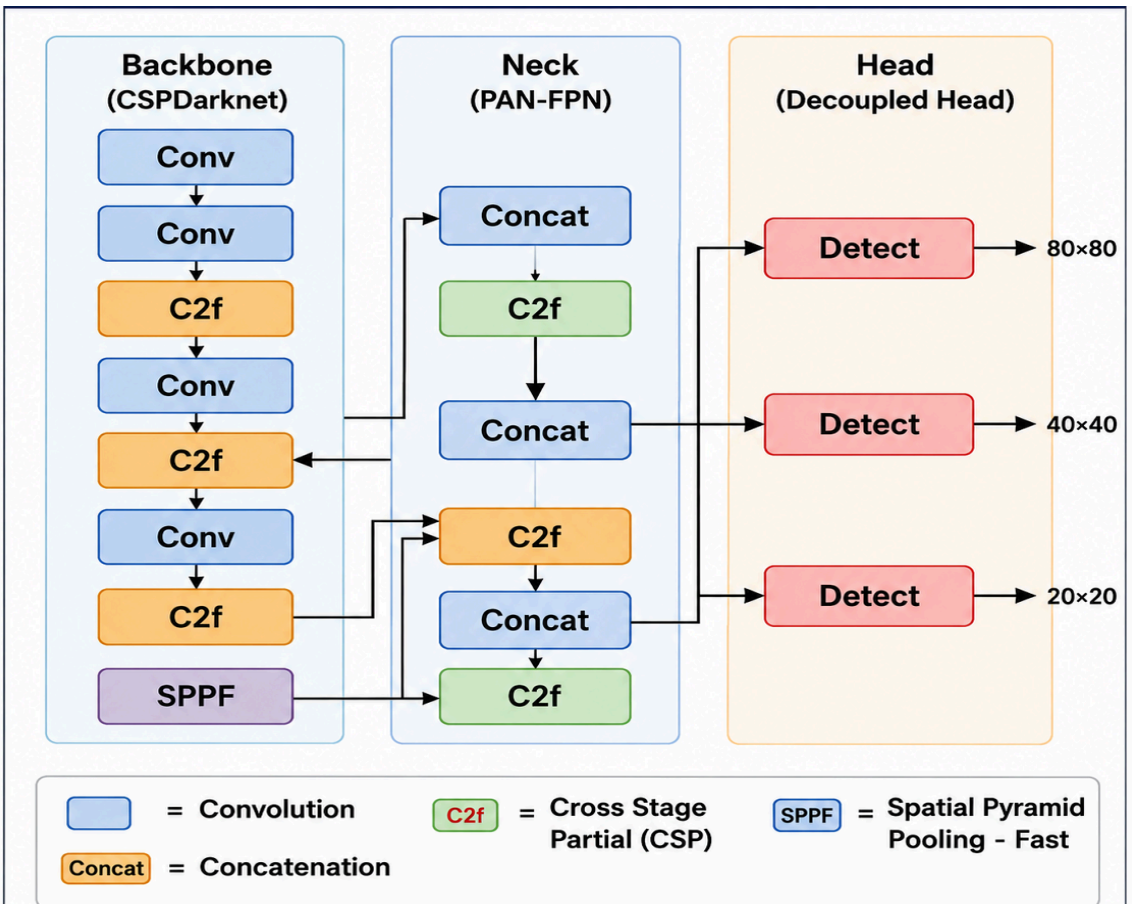


Figure 2.2: YOLOv8 architecture consisting of backbone, neck and decoupled detection head.

The detection loss combines classification loss and localization loss. Intersection over Union (IoU) measures box overlap and is expressed as:

```
IoU = area(B_pred intersect B_gt) / area(B_pred union B_gt)
```

Non-Maximum Suppression is applied after prediction to remove duplicate boxes. In dense traffic scenes, this step is significant because overlapping detections may otherwise inflate object counts and trigger false events.

2.6 Multi-Object Tracking and ByteTrack

The tracking by detection design approach is suitable for this since object detection is performed on each frame of the camera and association logic is applied. Basic geometric matching is extended by ByteTrack using low-confidence detections in a second association step, following high-confidence detections. Its importance in a real traffic scene is that it is possible for a portion of a person or vehicle to be partially blocked and still be part of an overall track that is being maintained, yet the partially blocked track may have a lower level of confidence in the score.

Persistent track IDs are the time-continuity that is the basis of risk assessment. If no tracking, each frame is treated by itself, and the system really doesn't have any way of determining that an object is stationary, is coming towards or is going away. Tracking can help calculate various parameters such as velocity, direction, dwell time, and time-to-collision, to close this gap between observation by the detector and deduction by the event engine.

A bus that remains within the same area for a number of seconds, should not be viewed as a new detection for every frame. The following reasoning is then executed downstream and track IDs are passed to it, enabling the system to not only prevent duplicate alerts, but to also analyse the trajectories and distinguish a stopped vehicle from a moving one.

2.7 Transformer Architecture and Large Language Models

Transformers use self-attention to model relationships among tokens in a sequence. Given query Q, key K, and value V matrices, scaled dot-product attention is computed as:

$$\mathbf{Attention(Q, K, V) = softmax(QK^{\top} / \sqrt{d_k})\, V}$$

19

This mechanism allows the model to assign different weights to different parts of the input depending on relevance.When generating a response, the current event state is compared with the most relevant context retrieved by attention to decide the next context.

Note that the model here is not used as a visual classifier. Putting it into a generic model would lead to a difficult to audit decision making process and privacy concerns about footage being recorded. Rather, the model is given a structured representation of the text, generated by the computer vision pipeline. Each of the pieces of evidence in the prompt may be demonstrated to a measurable detector/Tracker output to ensure the generated explanation is grounded, and to prevent unsupported inference from being treated as fact.

This also reduces the number of tokens used, since the information in a small, well-structured description of the scene is more useful per token than the information in raw pixel data is, and thus more relevant. The result is a step of reasoning which is efficient and easily understood.

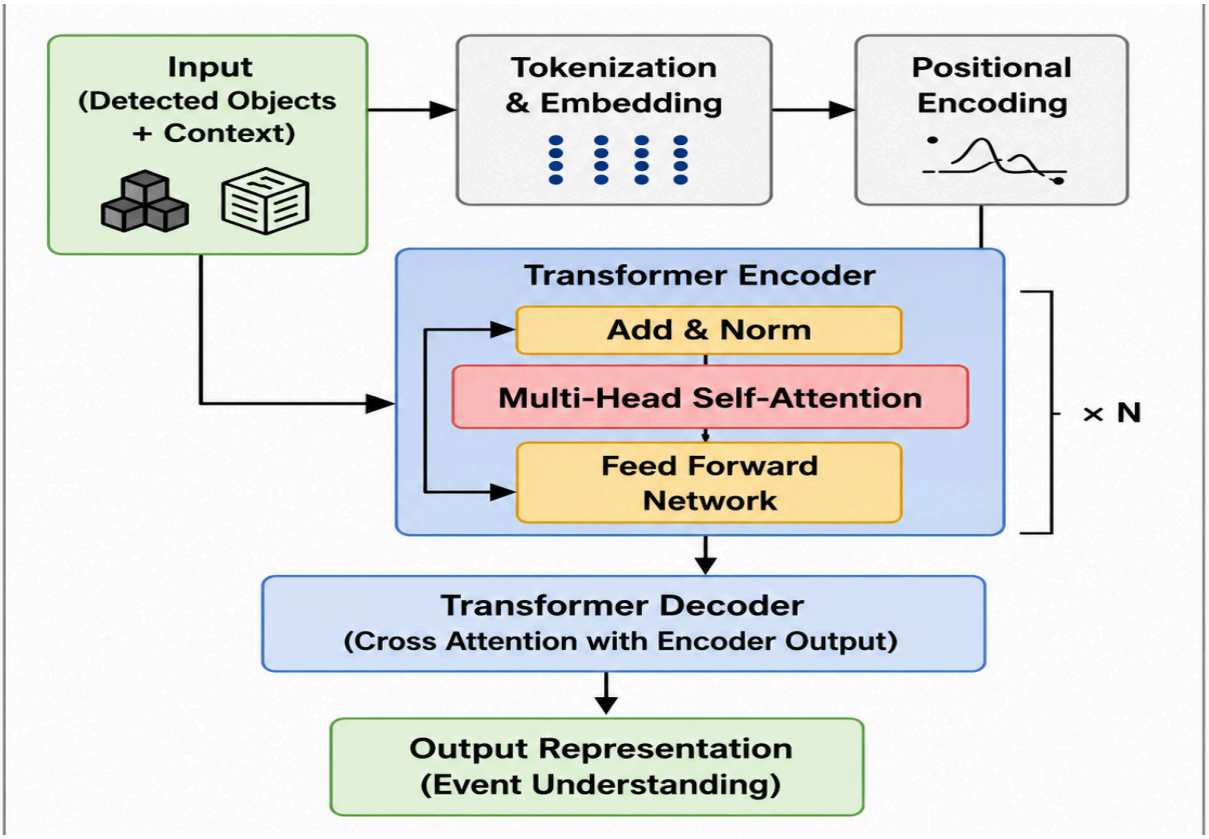


Figure 2.3: Transformer-based workflow for contextual understanding of detected surveillance events.

2.8 Retrieval-Augmented Generation, FAISS and LangChain

Retrieval-Augmented Generation enhances the reliability of generated explanations by first bringing information from its external knowledge base before it generates anything else. If the event is detected, it is a short text query whose state is serialised and added to the dense vector. This vector is then matched with a pre-indexed knowledge base with chunks of information about traffic rules, safety limits, crowd control, and operational instructions. The most successful chunks are then added to the prompt to give the model access to more relevant and important domain-specific material, instead of just the information it was exposed to during training.

The embedding is normalized before indexing, so that inner-product search can be used as an approximation of cosine similarity. The similarity of two vectors u and v is given by:

$$\cos(u, v) = (u \cdot v) / (\|u\| \|v\|)$$

Each step of the pipeline can be connected to the next with LangChain, which serves as the orchestration layer. Here is the step-by-step processing of this system:In this system, the processing steps are as follows: The explicit definition of this sequence guarantees that all explanations are generated in a traceable manner, irrespective of the type of event being analysed.

2.9 Explainable Artificial Intelligence

The importance of explainability in surveillance systems: alerts can directly trigger operations. When an operator doesn't know why an alert is raised, he or she has very little reason to decide whether to take any action or ignore it. The traditional techniques of saliency maps and Grad-CAM attempt to show parts of an image that affected the output of a model, but these are not attempted to communicate a meaning of the situation, nor a concept of what to do about it, to an operator.

Graphically, the operator can see the bounding boxes and track IDs on the frame, letting him/her know what objects were detected and tracked throughout the time. The numerical output of a risk score, density, speed, proximity etc. is provided to provide a measurable basis for the alert, in addition to the visual output. Retrieved snippets of knowledge are shown at the retrieval level to indicate the source of the explained knowledge. At the language level, a succinct explanation in natural language and a recommended action is produced based on evidence gathered at the previous levels.

This multi-layered structure is better for operation monitoring than one visualisation. A heat map will only show which pixels had a big impact, it will not tell you what the event is and what to do

about it. A natural-language explanation that is based on structured, measurable evidence and is supported by retrieved domain knowledge will provide the non-technical user with an explanation that is readable, understandable, and actionable in terms of what the operation of the system means. This is because every alert can be looked at from the detection level right up to the action recommended by the operator.

2.10 Evaluation Metrics and Findings

Table 2.1: Summary of literature review on surveillance intelligence and language-based reasoning.

Area	Representative Method	Strength	Limitation	Gap Addressed
Object Detection	YOLO/Faster R-CNN	Strong localization and classification	Limited event reasoning	Add tracking and event analysis
Tracking	SORT/DeepSORT/ByteTrack	Temporal object identity	Sensitive to occlusion and camera view	Use ByteTrack with detection confidence recovery
Video Event Analysis	Rule-based and deep anomaly models	Detect temporal patterns	Often fixed event taxonomy	Compute interpretable risk metrics
RAG	Dense retrieval with FAISS	Grounds language generation	Mostly text-centric	Apply retrieval to structured event states
LLM Reasoning	GPT/LLaMA-style models	Natural-language explanation	Potential hallucination	Constrain with evidence and retrieved context

2.11 Technical Comparison of Existing Approaches

The existing approaches are compared technically in this section.In this section, existing approaches are technically compared. Literature on surveillance can be roughly divided into three levels: visual perception, modelling of temporal events, and semantic explanation. Compared with the other three, visual perception is the most developed and ready-to-use, as high-performance detectors are now available to function in real time on a variety of scenes. Temporal modeling is less well established, partly due to the fact that the views of the camera vary over time in the real world, occlusions are common, and the transitions between events are often not sharp.

In visual perception, the decision between architectures of detectors is a compromise between accuracy and throughput. Two stage detectors have good localization capabilities, but also have latency that becomes an issue if multiple streams of camera data need to be processed at once. One stage detectors provide high throughput, but can have problems in the case of small size targets or partial occlusion. The architecture in this project is based on YOLOv8x as the reference and

maintains the detector as a plug-and-play module. The weight of YOLOv8s or TensorRT optimised weights can be used for the same pipeline for higher throughput without changing any other part of it.

In the world of temporal modelling, tracking by detection techniques differ in their approach to occlusion and identity recovery. Even though SORT is a very simple algorithm, it is very sensitive to geometric overlap and can fail if an object is partially obscured. The problem is solved by DeepSORT which incorporates appearance features that enhance the robustness with a higher processing load. The ByteTrack is adopted here since it can recover detections with low confidence scores in its second pass association, especially in the case of high vehicle density and interferences such as the overlapping of buses, cars, and pedestrians, which lead to the detection confidence score dropping sporadically.

In semantic explanation, the retrieval-augmented generation method has been mainly explored in the context of question answering and document-grounded generation tasks. In this context, the query is a user asking a natural language question. In this work the query is an automatic structured event description generated by the event engine. The retrieved context is not meant to replace the event engine's results, it is meant to aid in the understanding of the event, in order to categorize the event and to generate explanations, based on relevant domain knowledge, not general training knowledge.

Table 2.2: Comparative characteristics of surveillance reasoning approaches.

Approach	Input	Output	Advantage	Limitation
Motion detection	Frame difference	Moving region	Very fast	Sensitive to noise
Object detection	Single frame	Box and class	Accurate object localization	No temporal reasoning
Tracking-by-detection	Frame sequence	Track IDs	Temporal continuity	Depends on detector quality
Rule event engine	Tracks and zones	Risk label	Interpretable thresholds	Limited semantic explanation
RAG + LLM	Structured event text	Natural-language report	Contextual and explainable	Higher inference latency

2.12 Research Gap Summary

The literature survey uncovered that most work related to the detection, tracking and language generation are conducted individually. A detector could have a high mAP, but not be capable of determining if a scene is out of a traffic safety condition. If the LM does not have a retrieval context, and there is no sensor evidence that is structured, there is a possibility of hallucination. A subtle piece of advice in natural language is not possible with a rule based

engine . The idea of the thesis is to fill this gap with the unification of all three paradigms. The detection layer is the evidence that is seen, the event layer is the measurable risk and the language layer is a human-readable interpretation based on retrieval. The combined system is more useful for operational monitoring purposes than any of the individual systems.

Chapter 3

METHODOLOGY

3.1 Objective

This methodology aims to transform a raw video from the surveillance camera into an explainable report of the event. The framework is not a single model, but a series of interdependent modules, with each module having a specific input and output that it feeds into the next. It is important during the assessment as these can be assessed separately, detection accuracy, tracking consistency, precision of events, relevance of retrieval, and quality of explanation, all of which are preserved in this structure from the input frame to the final explanation. This decomposition allows an error in the final output to be traced back to the point at which it occurred in the system, though not by considering the system as a black box.

In addition, intermediate products generated by each module can be saved and analyzed separately. This is similar to the engineering practice in safety-critical applications where transparency of the components is required and not an option. You can't meaningfully evaluate a multimodal pipeline as a whole without being able to evaluate each step of the pipeline individually.

The frames of the surveillance pipeline should be processed continuously with minimal usage of memory, and the results of the pipeline should be made available to operators in a format that they can quickly read and take action on. Off-line accuracy is not enough unless the system provides acceptable real-time throughput or alerts are of such a nature that they need expert knowledge to interpret. The methodology, therefore, not only considers the design of each individual processing stage but also the overall system workflow, thus making the resulting framework suitable for deployment in an operational environment instead of only in controlled experimental conditions.

3.2 Methodology

The methodology is gradually abstracted in seven phases, including data set preparation, video frame extraction, object detection, multi-object tracking, event and risk computation, retrieval-augmented generation, and operator dashboard for presentation of explanations.

In each step, the more raw data that is filtered out, the more meaning that is retained. The first stage processes the pixels, the second the detections, the third the object tracks, the fourth the event metrics computed from the object tracks, and the final stage the language. What makes this pipeline work at scale is that the data has been reduced and stored at each layer, and computation is only done on the reduced image, which is sufficient.

Selective processing is also catered to by the design. Routine frames go through the vision pipeline and change the count on the dashboard, while retrieval/explanation is only activated if the high-risk event is detected. The language model would be used on each frame, which would be a waste of computationally and produce too much output for operators to be able to monitor. This allows the more costly modules to be used for frames which merit deeper scrutiny, while still being responsive for the part of the system that doesn't. The stationary vehicle or person in the restricted zone is considered the same one as identified in the previous frame, not to be identified as a new vehicle or person on every frame. This helps the event engine from making duplicate alerts and helps the operator give his/her attention to new developments.

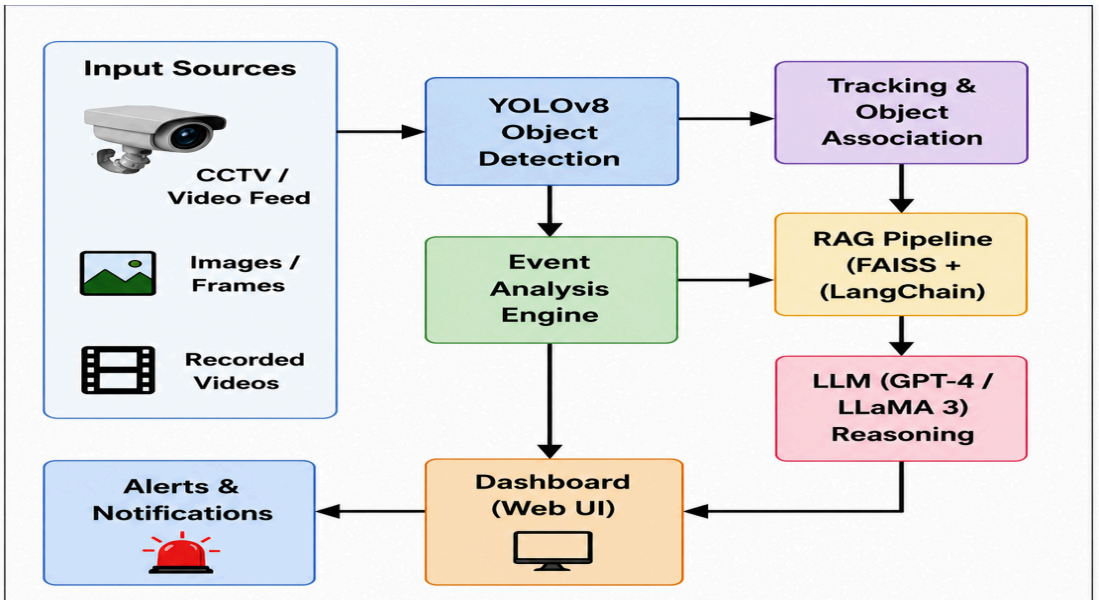


Figure 3.1: Overall system architecture of the proposed Explainable Intelligent Surveillance System.

3.2.1 Dataset Preparation

A mix of public data and custom-recorded traffic footage is used to evaluate the system. Multiple sources provide a more general evaluation since they will expose the pipeline to a variety of scene types, camera perspectives and event conditions, as opposed to optimizing the pipeline for a specific domain.

COCO offers extensive coverage for objects and allows for the initialisation of the detector for the most common person and vehicle categories. VIRAT provides outdoor surveillance viewpoints with realistic human-vehicle interactions from hard-coded camera setups. UCF-Crime presents long-duration video clips with unusual events to challenge the pipeline's performance in scenarios outside of normal traffic monitoring. All data sets are normalised to a common class set prior to evaluation. Other categories are ignored, and vehicle-related categories from other sources are translated to a common category set for surveillance of cars, buses, trucks, motorcycles and bicycles. This normalisation step guarantees that the outputs of the detection are the same, no matter from which data source a frame comes. It also makes the process of event-state serialisation easier, because the downstream reasoning stage is given stable, predictable class names, instead of having to deal with inconsistent or source specific labelling.

Table 3.1: Dataset summary used for training and evaluation.

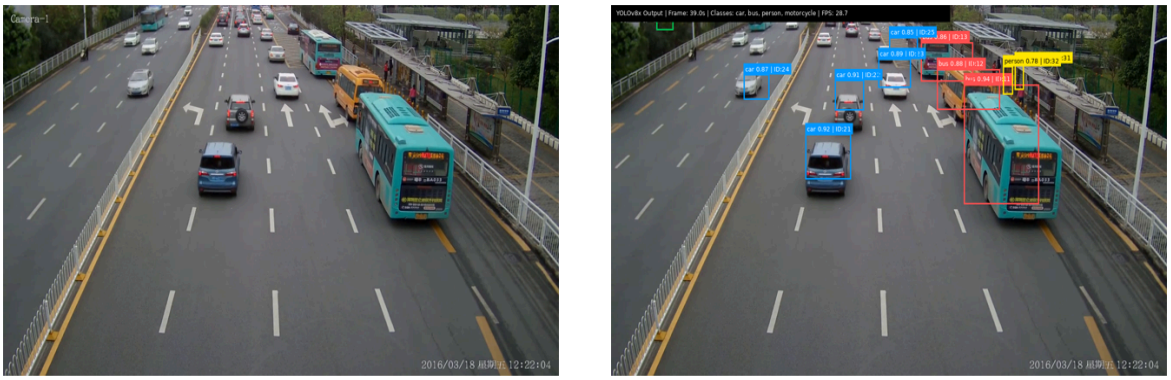
Dataset	Purpose	Main Content	Use in Thesis
COCO	Pre-training support	Common object categories including person and vehicles	Class initialization and category mapping
VIRAT	Surveillance validation	Outdoor activities with persons and vehicles	Detection and event validation
UCF-Crime	Anomaly analysis	Long surveillance videos with abnormal incidents	Stress testing event descriptions
Custom Traffic CCTV	Deployment evaluation	Traffic lanes, buses, cars, motorcycles and pedestrians	Input-output frames, dashboard and final results

3.2.2 Video Processing Pipeline

The video frames are captured using the OpenCV library. In "streaming" mode the frame capture and inference is split into 2 threads, so that if the model takes a long time to execute, camera acquisition is not delayed. This producer-consumer system also allows the system to discard the outdated frames or employ the adaptive sampling when the calculation capability increases, without falling behind the live feed keeping the dashboard close to real-time.

The geometric relationships in the original scene should not be altered in any way during the frame pre-processing, because the spatial distance and direction of the movement of the object in the scene is the basis for event analysis. Scaling a frame without preserving the aspect ratio of all the images causes vehicles to be mis-positioned with respect to the relative proximity of others within the frame relative to the actual position of the others in the scene, which may cause an error for estimates of proximity and computing of trajectories. To do this, it's necessary to employ a technique called “Letterboxing,” in which each frame is cut to fit a particular detector dimension, while keeping the aspect ratio of the original frame. Original frame and resized model input is co-ordinately mapped, so that the location of the bounding boxes can be read back from the original image during the display of the outcome.

Original frame and model ready frame are retained during the processing. The original is showcased and tested, the preprocessed version is fed back into the detector. The difference between both is illustrated in figure 3.2, where the raw input frame is the image obtained from the camera, and the output frame is the frame on which the bounding boxes, confidence scores and track identifiers of the detection and tracking stages are overlaid.



(a) Raw input frame from selected IP traffic video

(b) Same frame after YOLOv8 detection and ByteTrack IDs

Figure 3.2: YOLOv8 input frame and corresponding detection output from the selected traffic video.

3.2.3 YOLOv8 Object Detection

Based on the accuracy and real-time performance of the detector on the GPU, YOLOv8x is chosen as the main detector. These classes are kept: person, car, bus, truck, motorcycle and bicycle. Weak detections are suppressed by a confidence threshold and multiple boxes that predict the same object are removed by the class-wise Non-Maximum Suppression.

The confidence threshold is a compromise between recall and precision. Low threshold makes it more likely to detect occluded or distant objects, but also increases the likelihood of spurious boxes that may not represent real objects. A very high threshold will result in fewer false

detections but will also result in some objects that are present being missed. The use of this value is based on the subject of downstream event analysis and not exclusively for the purpose of detection performance. If the threshold is set too high, then detections that can be used to keep the tracker from going out of identity during partial occlusion would not be kept by ByteTrack in the second association pass.

The detection is represented as a tuple of structured data, with the bounding box coordinates, class label, confidence score, and frame identifier. This format is not reconstructed at each stage of the pipeline, but follows the pipeline throughout. The bounding box is necessary for visualisation and proximity computation, the class label is necessary for evaluating event rules, the confidence score is useful for estimating the reliability of a detection and the frame identifier can be used to associate a detection to the original video time stamp. By keeping the same representation for the detection, tracking, event computation, and explanation, the values in the final output will represent what was measured at the detector, and no ambiguity will be introduced by reformatting at intermediate steps. Each detection is stored with the following tuple:

$$d_i = [x1, y1, x2, y2, class_i, confidence_i, frame_id]$$

This tuple is then passed on to the tracker and event engine. It is important that the same values are in a structured format, since later the LLM will be given a text serialization of these values.

3.2.4 ByteTrack Tracking

Throughout processing, ByteTrack has three types of tracks: active, lost, and removed tracks. Active tracks are tracks that have objects displayed in the frame. The lost tracks are not deleted immediately, and are stored temporarily in a buffer in order to allow the tracker to re-acquire an identity after a small period of occlusion without having to begin a new track from scratch. This behaviour is especially important where a vehicle passes behind a bus or a pedestrian temporarily goes behind a roadside object, as can be seen in road scenes. If the subsequent detections are made in the same area as predicted, the tracker can reidentify the returning detection as part of the same object and not a new one.

Internally there is a state vector for each track, which contains its position, scale, velocity and identity. Incoming detections are used to solve the assignment problem with the predicted track positions, which is solved by using a Kalman filter for motion prediction in the forward direction. Moderate amounts of clutter and short-term disappearance that are present in the scenes can be handled with little identity fragmentation by this combination.

In addition to association, track history is also employed to generate motion descriptors of each object. The displacement between the centers of successive bounding boxes is used to estimate the direction of motion, across a fixed temporal window the centre point of each bounding box is recorded. To estimate the speed, that displacement is divided by the frame interval to get a speed in image coordinates. These image-plane measurements can be approximately projected onto road-plane units for more physically interpretable object velocity measurements for downstream event and risk computation, provided camera calibration parameters are available.

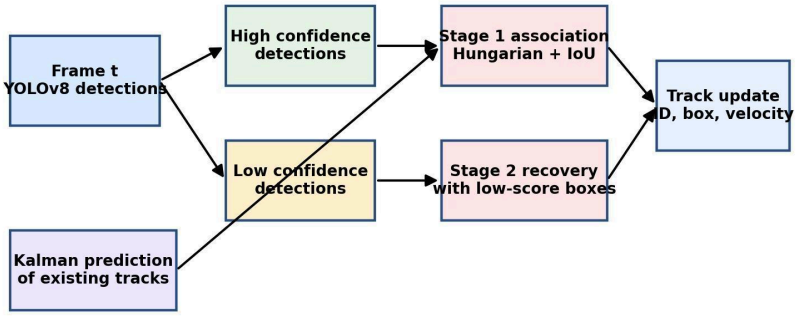


Figure 3.3: ByteTrack association workflow for tracking detected surveillance objects.

3.2.5 Event Analysis Engine

56 1 The outputs of the detection and tracking stages are fed into an event analysis engine which translates it to interpretable risk indicators. The state vector s_i = [x_i, y_i, w_i, h_i, v_x, v_y, c_i, t_i] represents the spatial position, bounding box dimensions, velocity along x and y axis, class of the object and the elapsed time since it was first tracked in the scene for each tracked object. The pairwise features are then computed for each vehicle-vehicle and pedestrian-vehicle combination, and used to estimate the collision risk. The size of the crowd is estimated by summing up the points of the pedestrian in a spatial grid separately.

The engine does not obscure the view, it is rule assisted. Quantities which can be examined directly, such as inter-object distance, relative velocity, object class, zone membership and dwell time, are used to derive risk scores. This interpretability should be observed when an alert is activated and should be able to be seen by an operator, and should be a conscious design choice. Also, if the camera model changes but the camera is moved to a new setup, the system can be calibrated again in the new setup.

When estimating collision risk, the proximity and approach behaviour is considered. If they are close together, there seems to be no threat, if they are close, but moving towards each

other, at high relative velocity there may be threat. With only distance, there would then be false positives and false negatives. To capture this temporal aspect, a new metric, called the Time-To-Collision, is introduced, a more meaningful signal to identify high-risk events than static distance thresholds can provide.

The number of people is not the absolute number of pedestrians, but measured in a spatial grid. While a large group of pedestrians in an open area may not be a direct concern, small groups of pedestrians at a narrow pedestrian crossing, bus stop/boarding area or bus exit may be an indicator of congestion that is starting to occur.

$$TTC_{ij} = d_{ij} / |v_{rel\_ij}|$$

The calibrated distance is d_ij, the relative velocity along the direction of the two objects is v_rel_ij. The distance and TTC is compared to their thresholds and if they are below, a Risk Flag is set.

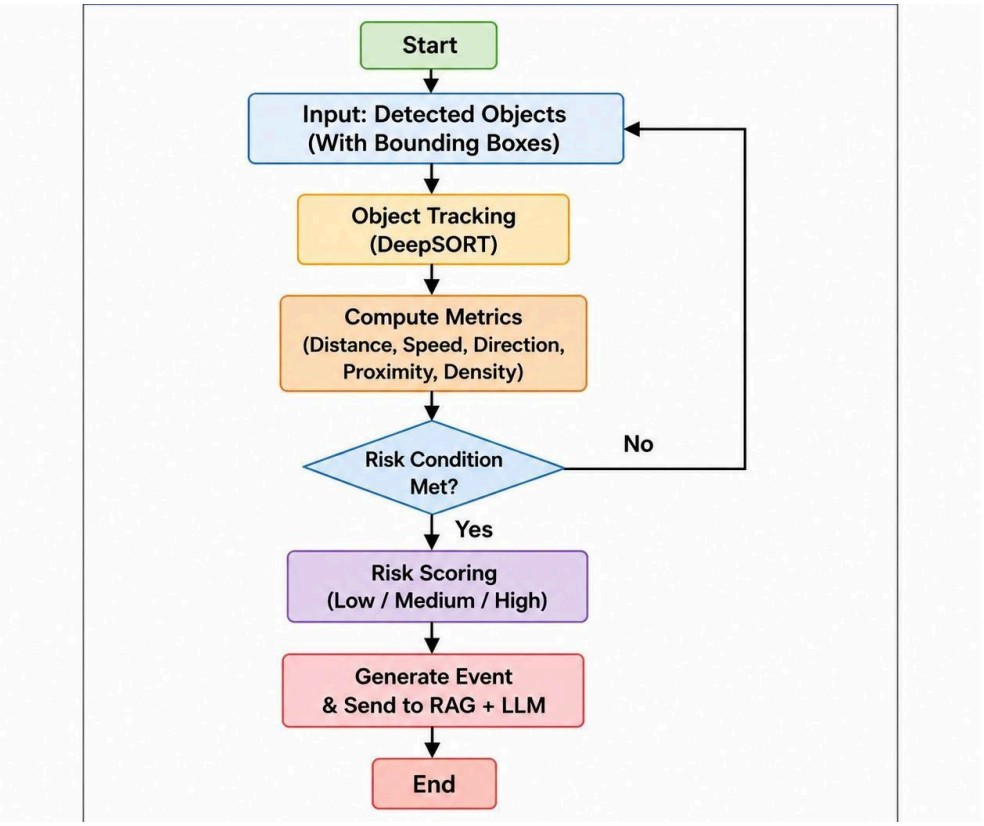


Figure 3.4: *Event analysis flowchart for risk detection and decision making.*

3.2.6 RAG and LLM Reasoning

The system looks at what's happening and figures out how risky it is. It then takes all the important details, like where the camera is, when it happened, what was seen, and how serious it is, and puts them into a simple summary. This summary is then used to find similar

situations that have happened before, using a special kind of search engine. Before trying to come up with a response, the system looks at the most similar situations it can find and adds that information to the summary, so it can make a more informed decision about what to do next. The summary includes things like what kind of event it is, how severe it is, and what the rules say should be done in that situation. All this information is used to generate a response to the event.

The way we structure our event-state text is very concise on purpose. We only include things that can be measured and checked. We leave out any descriptive words or details that aren't directly related to the facts. There are two main reasons for this. Firstly, it helps reduce the number of words or "tokens" being used, which is important when there are a lot of alerts coming through the system. Secondly, it ensures that the information used to generate explanations from the vision pipeline is clear and can be traced back to a specific detector or tracker output, rather than just being an assumption made by the model. This approach is crucial for maintaining accuracy and reliability in our system.

The model response is restricted by an output schema. The dashboard requires a standard template with a risk level either high or medium, a brief summary, factors contributing to the risk, and a recommended action. Enforcing a schema allows them to automatically interpret the results in the interface without human involvement, and each response can be saved as a structured data type, for later audit and comparison. Produce them in free format, which is readable without much effort, but might be difficult to process reliably across a large number of events and would make any systematic review of the system's reasoning over time difficult.

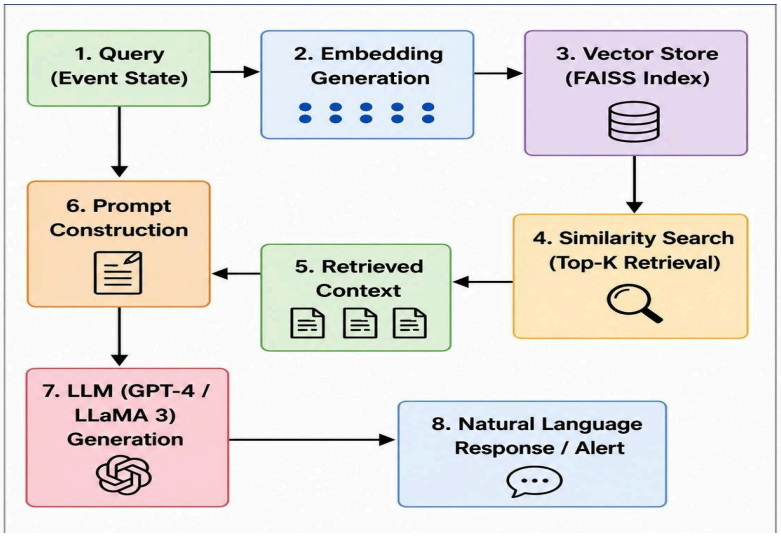


Figure 3.5: Retrieval-Augmented Generation pipeline for contextual reasoning and explanation.

The LLM prompt contains a system instruction, structured event evidence, retrieved context and a generated response schema. The output schema requires fields such as summary, risk_level, contributing_factors, recommended_action and confidence. This structured generated response allows downstream dashboard components to render alerts consistently.

3.2.7 Dashboard and Operator Workflow

The dashboard is designed around a single practical requirement: reducing the time between an event being detected and an operator understanding what it means. Four panels are presented together on the same interface. The live video panel overlays bounding boxes and track identifiers directly on the camera feed, showing which objects the detector found and how they have been followed over time. The risk panel summarises the computed scores and density measures that caused the event to be flagged. The retrieval panel displays the knowledge chunks that were pulled from the indexed store to ground the explanation, and the explanation panel presents the structured model response in terms of risk level, contributing factors, and recommended action.

Laying out these four elements together means an operator does not need to switch between views or mentally connect separate outputs. The visual evidence, the numerical justification, the retrieved context, and the operational recommendation are all visible at once, which supports both rapid situational awareness and more careful inspection when a decision needs to be justified or escalated.

Operator interaction is treated as a functional part of the workflow rather than an optional feature. Each alert can be marked as active, resolved, or false positive. These labels are not merely cosmetic; they record which events the operator judged to be genuine and which were considered spurious, building a log that is directly useful for diagnosing where thresholds or prompt constructions may need adjustment. In a longer-term deployment, this feedback could inform iterative refinement of the event engine and knowledge base, gradually improving the system's alignment with the conditions of a specific camera scene or operational environment.

3.3 Evaluation Metrics

The evaluation framework is tailored to the unique nature of a multi-modal surveillance pipeline as there is no single metric that can adequately capture it. System stages are judged based on the appropriate measures of the outputs. Detection performance is evaluated at visual perception level; the tracking stability is measured using identity consistency between frames; the quality of the event is measured using the risk classification accuracy; the retrieval quality is measured using the contextual relevance; the quality of the explanation is measured using the usefulness rating from the operators. This would all be compressed into one number, and conceal the instances when the system is good and when it is not.

17 This granularity is important because the slowest module will restrict the performance of the overall system and the first step to optimizing the overall system is to properly identify the slowest module. The end-to-end latency alone would only mean that there is a problem - but not where.

Parameter	Value	Purpose
Input resolution	640 x 640	YOLOv8 model input after letterboxing
Detection threshold	0.35	Balance recall and false positives
NMS IoU threshold	0.45	Remove duplicate detections
Track history	30 frames	Velocity and direction estimation
TTC threshold	3.0 s	Collision-risk screening
FAISS top-k	5	Knowledge chunks for prompt
Output schema	JSON-like	Dashboard parsing and auditability

Table 3.2: Main system parameters and thresholds.

The evaluation is done based on detection performance measure of precision, recall and mAP. FPS and module-level latency are used to measure real-time performance. The accuracy of 42 event performance is measured based on precision, recall and F1-score with regard to annotated and/or manually verified event labels. Evaluating retrieval relevance for RAG and actionability, consistency and grounding for LLM explanation.

3.4 Algorithmic Specification

The vision modules are always activated for each incoming frame, whereas the language generation is activated conditionally depending on the risk value calculated. This is key to the viability of the system. Most of the frames within a surveillance stream are mundane and the amount of retrieval and explanation that could be done on each frame would be more than an operator could reasonably follow.

All frames are preprocessed first and then fed into YOLOv8 for detection . If detections are detected, the ByteTrack updates the identities and motion history of the active tracks and then the event module calculates object relationships and risk metrics among the pairs. Only the dashboard counters get updated when a frame is classified as low risk. Medium and high level and critical events are serialised and passed to the retrieval stage, where appropriate knowledge chunks are retrieved prior to calling the language model.

The system is also capable of performing deduplication based on track-ids and temporal windowing. A vehicle's presence at a bus stop in subsequent consecutive frames should not trigger a new alert since the vehicle is always at the bus stop. On the contrary, in-place updates are applied to the current event state, until a change occurs in the underlying condition or the event is resolved. This helps to silence unnecessary alerts for the same physical situation and to limit alerts that could lead to alert fatigue in surveillance interfaces.

The conditional triggering policy coupled with deduplication logic makes sure that the costly modules, especially retrieval and language generation, are only used when they bring value to the output. This leaves you with a pipeline that can grow more gracefully with stream counts than a pipeline that processes all frames in the same way without losing any detail in an important event.

3.4.1 Event State Serialization Format

40 Serialisation serves as the interface between the computer vision pipeline and the language The language reasoning stage is the interface to the computer vision pipeline, and this is where serialisation comes in. Raw tensors and bounding box coordinates are not useful inputs for a language model, however, the same information can be summarized by a structured text record which is compact, highly interpretable, and useful for a language model. This format is designed to be both machine-readable and human-readable and makes it possible to log,

embed semantically, query for retrieval, and audit after the fact without any conversion between the uses.

The serialised event state includes camera ID, the time stamp of the frame, the number of objects detected, IDs of tracked objects, risk metrics, the label for the event, and local scene zone. One such representative example is given below:

The camera is a CCTV-01, the time is 39.0s, the objects are car (5), bus (3), and person (1), the risk_collision is 0.91, the density is 0.18, and the event is TRAFFIC_CONGESTION with the zone being bus_stop.

This record contains the key evidence for a single high risk frame that can be directly embedded as a semantic query against the FAISS index. The representation eschews the use of raw frame data, significantly lowering token usage and privacy risks as no identifiable visual data is transmitted beyond the vision pipeline. But there's also a more nuanced benefit; the language model is only given the numbers it's explicitly measured by the detector and event engine, and the generated explanation is based on verifiable evidence rather than on what the model might infer from the presence of pixels.

The format is extensible as well. Other fields like weather condition, camera zone, road direction or traffic signal phase can be added when the information is available from other external sources, and no changes are needed in the downstream prompt nor in the retrieval process. This implies that it is possible to add increasingly sophisticated contextual reasoning to the pipeline without changing its structure.

3.4.2 Prompt Template Design

There are four blocks in the prompt template. The instruction block specifies the role of the model as a surveillance analyst, and explicitly limits what the model can assert and what it cannot assert. The evidence block is the serialised event state from the vision pipeline. The retrieved context block contains the most relevant passages retrieved from the FAISS index. The output block defines how the response is supposed to look like. These blocks provide a self-contained input, providing the model with all the information required to generate a grounded explanation, but not enough to elaborate it. These limitations are important for scenarios in which the output of the system is influencing real world decisions made by the

surveillance system and if the claims are exaggerated, they can have effects beyond the scope of the system.

A template-response (as opposed to a free-form paragraph) also increases the reproducibility for experiments. The model is requested to return the following five fields: a summary, a risk level, contributing factors, a recommended action and a confidence value. If all explanations are explained in the same format, subsequent comparisons of outputs between events/experimental conditions are easy. Evaluation of actionability, risk classification correctness and retrieval grounding can be performed consistently for each field without having to rely on subjective interpretation of the unstructured text. Implementing the fixed schema also makes the dashboard much easier to build as the interface component that exposes each field has a one to one mapping.

3.5 Mathematical Formulation of Risk Scores

For each object i, define $p_i(t) = (x_i(t), y_i(t))$ as its centre at time t.For the ith object, let its centre at time t be $p_i(t) = (x_i(t), y_i(t))$. Two objects i and j have image-plane distance $\|p_i(t) - p_j(t)\|$ then. If the camera calibration parameters are known, the coordinates on the camera's image plane are transformed to approximate ground plane coordinates by using a homography matrix H to aid physical interpretation of distance thresholds for road scenes and to minimize perspective distortion from the camera.

The collision risk is calculated by the relative approach speed and the proximity. Two objects approaching each other while decreasing their separation are much more dangerous than two objects at the same separation that are actually separating.The risk formulation is more dangerous than two objects are approaching and decreasing their separation, rather than two objects at the same separation that are actually separating. The score is calculated as:

$$R_{collision} = \exp(-\lambda \cdot TTC_{ij}) \cdot \max(0, 1 - d_{ij} / d_{max})$$

There are different operational meanings for each term. The smaller the time-to-collision TTC_{ij} the larger the score, because a potential collision is closer in time. The distance term will reduce the score for the objects if their distance is beyond the interaction distance d_{max}, and the max operator will guarantee that the total value will not be negative when exceeding the interaction distance d_{max}. It is intentionally formulated to be interpretable: the behaviour of the learned scoring function is hard to investigate while it is deployed, whereas each

individual component can be investigated and adjusted independently when the system is deployed under different conditions of the scenes.

A spatial grid is used to represent crowd density, with each detected pedestrian corresponding to a Gaussian kernel centered on the center of the pedestrian's bounding box:

43

$$D(x, y) = \Sigma_i \exp(-((x - x_i)^2 + (y - y_i)^2) / (2\sigma^2))$$

The largest normalised value in the grid is then used to check against a threshold to decide if a crowd risk event should be raised. The method is more informative than just counting pedestrians as it records concentration of the group and not its size. A plaza that is not particularly dense might be needed to be more dense in the vicinity of a bus door, exit gate, or pedestrian crossing if it is important for the operation of that location. Thus the reason density estimates are meant to be used in conjunction with scene zone labels: these will provide a pragmatic basis for adjusting risk thresholds to the different camera installations without retraining any part of the vision pipeline.

Chapter 4

SYSTEM IMPLEMENTATION

4.1 Environment Setup

The system is deployed in Python, largely because the toolset of computer vision, machine learning, vector search and website development is rich and well-established. Implementation relies on existing well-maintained, well-used, and widely available libraries with which to build individual components, thus promoting research reproducibility and easy extension.

YOLOv8 inference is facilitated via Ultralytics library implemented on PyTorch, making the model loading, filtering by classes, and setting the confidence threshold extremely simple and readable to just a few documented function calls. Video input, manipulation and rendering of bounding boxes is handled by OpenCV and can be accessed both from recorded files and live video input streams by the same interface. The operator dashboard is built in Streamlit and lets the visualization and text-based output from the pipeline to be displayed on a multi-panel dashboard without a separate frontend codebase.

The number of active camera streams and the type of detector used determine the hardware requirements. Throughput is boosted at the inference stage of YOLO and sentence embedding stages, which are both parallelizable. The remaining parts of the pipeline after inference, such as frame capture, event computation, serialisation and dashboard rendering are not intensive computation-wise compared to inference and supported by CPU resources. Smaller YOLOv8 models, TensorRT-optimised weights, quantised models, or locally-available compressed language models can be used without making any structural changes to the pipeline. This flexibility was taken into account during the design as the surveillance infrastructure varies significantly from real deployment.

Table 4.1: Software and hardware environment.

Component	Configuration
Processor	Multi-core CPU with AVX support
GPU	NVIDIA RTX-class GPU used for real-time inference
Memory	16 GB or higher recommended
Programming language	Python 3.x
Detection framework	Ultralytics YOLOv8 with PyTorch
Tracking	ByteTrack implementation
Vector database	FAISS IndexFlatIP
Orchestration	LangChain
Dashboard	Streamlit
LLM backend	GPT-4 API or LLaMA-style local inference

4.2 Detection and Tracking Implementation

The weights for YOLOv8 are loaded during initialisation and frames are batched or single frames for recorded video or live stream. Each set of detections are filtered before any further processing by the class label and then the confidence threshold. The filtered bounding boxes are then re-formatted to a standardised coordinate space and sent to ByteTrack that outputs a set of user-defined object identities drawn on the output frame and written into the event-state buffer for further processing.

The conversion layer, which converts detector output to tracker output, is not assumed in the detector or tracker, but remains as a separate layer. If the detector upgrade is carried out in subsequent iteration, the conversion layer should be upgraded only. Everything downstream will be spared, as will be the tracker itself. Such loose coupling between pipeline stages is an intentional design decision that decreases the maintenance effort if a single component is upgraded or replaced.

Throughput controlled using a producer-consumer threading structure. Two threads: One to grab frames for adding them in a frame queue, and another to remove frames from the queue and do inference and tracking. This separation is significant as otherwise temporary delay in GPU inferences would stop the camera from acquiring the frames which would result in dropping of frames/drift in timestamp. The system can be used to process every frame of an offline video file or it can be used to sample the video file at a specific frame interval to meet the need for temporal completeness or speed.

Implementation log is maintained at the per frame level at runtime. For each log entry detection time, tracking time, number of active objects, event status, retrieval time (if applicable), and generation time are recorded. These logs are beneficial for debugging at development time, to generate input frame sequence and threshold values for each experiment run for a performance evaluation (Chapter 5), and to reproduce experiments under exactly the same conditions.

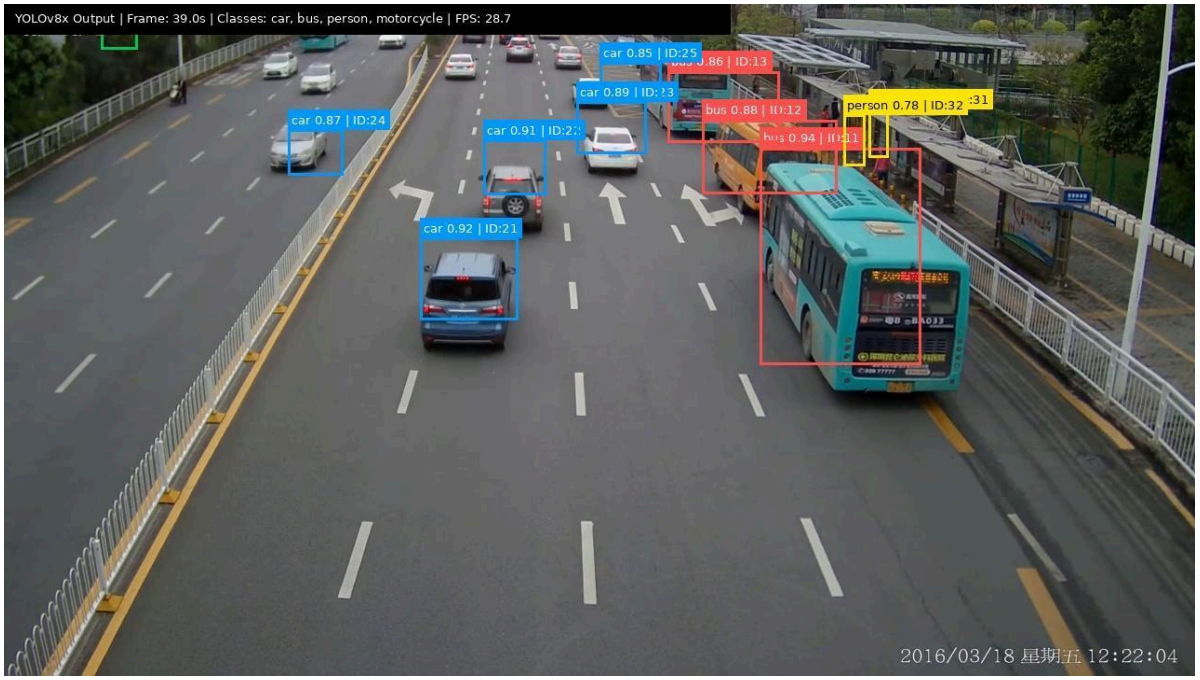


Figure 4.1: YOLOv8 detection output showing object classes, confidence scores and tracking IDs.

4.3 Event Engine Implementation

The event analysis engine is a collection of separate detector modules, each one for a particular type of event, all employing the same track buffer. The benefit of having these categories broken down into discrete modules is that it makes testing, tuning and evaluation of these modules more easy without interfering with other modules in the mix. All combinations of objects at collision are taken into account and distance between objects and relative motion between all vehicles and pedestrians are calculated in the collision module. The crowd module computes a spatial density grid of the current set of person detections. The stationary-object module detects stationary objects, which have an average velocity below a certain threshold for a given time interval, for a time period and identifies objects that have been stationary for

a longer period than usual. A continuous severity score and a structured evidence record that details the conditions which led to the score are generated by each module.Instead of a binary (triggered/not triggered) label, a continuous severity score is generated by each module, also with a structured evidence record that outlines the conditions that generated that score.

This design will also be able to prioritise alerts. If several modules fire on the same processing cycle, then the alerts that are generated are not treated as being equal but are ranked by their levels of severity and confidence. For example, a high collision-risk event is raised above a low density crowd observation event to enable operators to focus their attention on the area that is most likely to be in need of attention, rather than having to sift through simultaneous notifications.

Each event record is serially encoded and transmitted in a text-compatible file, which can be embedded and passed directly on to the retrieval chain without revealing any video content. If a traffic congestion event happens at 39.0s, the following serialisation could be produced: "At frame 39.0s, multiple buses and cars are found in the neighborhood of the bus stop zone. High traffic and low risk of collision, and low crowd density. Event type: TRAFFIC_CONGESTION. This representation guarantees that all quantitative evidence generated by the event engine will be saved in a way that it can be meaningfully inserted into a sentence-transformer model and the prompt to which the model is fed downstream will be short and free from visual elements that could trigger privacy issues or be more expensive in terms of tokens.

4.4 RAG and LLM Implementation

The Knowledge Base is constructed offline and deployed prior to deployment of the system. The short snippets of the source documents are embedded in 384-dimensional vectors using the sentence transformer model.The embedding in 384-dimensional vectors is done via the sentence transformer model, using the short snippets from the source documents. These normalised vectors are stored in a FAISS index, along with a metadata associated with each chunk with regard to category and source. This can be performed offline and at runtime, retrieval would amount to a forward pass through the embedding model and then an inner-product search - low latency as compared to detection and generation.

The event state is part of the runtime state like it was with the creation of the index. This is not something to be taken lightly: If the index is good, but the queries are embedded in different ways than the chunks, then the quality of retrieval will be compromised. Once the event

embedding is built, FAISS will return the similar chunks by the inner product score. These pieces are attached to the prompt template, using classification labels to give the language model a clear idea of what it is working on.

The prompt instructs the model to remain with its response as a fact-based text, and using evidence from the provided text, and not to speculate; in addition, the prompt provides the model with a framework for the report output defined by the output schema defined. Although the language model is given the instruction, it may occasionally not follow the formatting, which is taken into account in the implementation. A primary parser attempts to extract the fields it is looking for from the answer. If this doesn't work, the most essential information from the raw text will be extracted and stored and the whole response will be examined later using a fallback parser. This will help ensure that a formatting error does not cause the dashboard to fail and will provide a useful record of the times that the prompt is being mis-parsed – if the error is recurring, then the instruction or schema definition is probably in need of change.

4.5 Dashboard Implementation

A navigation sidebar is available for wider system functions, such as overview, live monitoring, event history, knowledge base, analytics, reports and alerts. The design is geared towards the requirements of the surveillance operator, rather than the internal organization of the pipeline. The live video panel has got a lot of screen space, because of the need for visual verification, which is an element of alert assessment and confirmation by operators. Cards for collision risk, crowd density and traffic flow are displayed in the Metric view to be seen at a glance without interaction to get the current scene state.

The risk value is colour coded to minimize the cognitive workload to process several concurrent alerts. If there are multiple events running, an operator should be able to determine the most critical event without having to read detailed numerical values. The retrieval and display of the context and explanation panels are used for the cases in need of further investigation, along with the evidence that warranted a particular alert and what the language model considered the progression of the event. In addition, there is a query box to allow an operator to ask why, for example, a particular event has been marked as high risk and get a natural language answer with the evidence pipeline.

Additionally the layout is designed with operativeness in mind and to make it transparent. Output of the detectors, association from the association stage, risk score from the event

computation stage, retrieved snippets from the grounding stage, and the language model output from the explanation stage are all shown on separate panels, allowing the reasoning of each part of the system to be observed. The top alert is not all that is presented, these layers are presented together and the operator can confirm, escalate or dismiss any alert generated by the system, keeping the whole system human-in-the-loop.

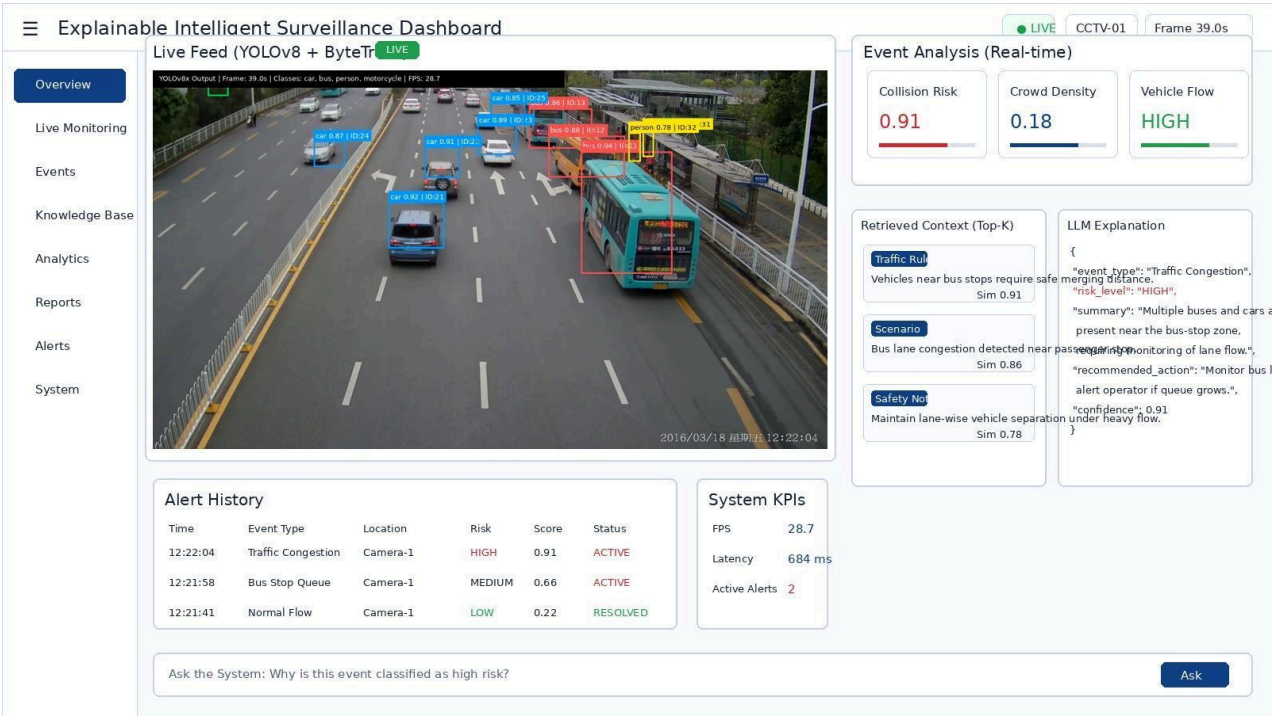


Figure 4.2: Explainable intelligent surveillance dashboard showing live feed, event metrics, retrieved context and LLM explanation.

This information can be reviewed in hindsight to verify or to correct flagged events or to analyse errors. Operators can also set alerts as false positives, and the false positive judgement will be kept with the incident record. This log may be reviewed manually by the user to determine if thresholds were met, or it can be referred to as a template for creating event rules and prompt designs in the future. So the audit trail has two different functions; its immediate use is to help the operation and its longer term use is to help in the iteration for system improvement.

4.6 Module Interfaces and Data Flow

The implementation is done in terms of clear data structures that define the output of each module, and the input expected by the next module. Detection Object includes a class label, a confidence score and a bounding box consisting of coordinates. This is complemented by an object's identity, age, last-seen frame, and its velocity, as provided by a track object. The event object includes the following: event type, risk value, timestamp, track IDs associated with the event, context retrieved, and the explanation generated. The structures are enforced at all handoff points: the dashboard and downstream portions of the system will not depend on the internal representation of the individual neural network backends, and it makes testing and future changes much easier.

The interchangeability of the module is influenced by this design. Each individual component can be replaced without changing any other components in the pipeline since data is passed between adjacent components using structured dictionaries and objects. YOLOv8 can be used instead of a later version of YOLO, OC-SORT (a ByteTrack-based OC-SORT), an approximate index based on HNSW or the current language model.YOLOv8 can be replaced by a later YOLO model, an OC-SORT version based on ByteTrack, an approximate index based on HNSW, or the current language model, which can be a locally hosted LLaMA-type model. Not only is this the case, but it is not easy to keep a system running in a couple of years after deployment, if the computer vision or language model tooling also continues to rapidly evolve.

The data flow is also designed to enable audit logging to be performed. The alert generated contains the following information: the frame reference number for the source record, object identifiers, a calculation of risk score, a list of retrieved document identifiers, a version of the alert prompt, an operator action taken, the raw model response, and the computed risk score. The advantage of having this context of all alerts is that the reasoning behind the automated alert would have to be presented to a third party in a safety-critical deployment.

In a frame loop the timing information is recorded at each processing step. The time is recorded separately for each of the following: Inference time, Tracking time, Event computation time, Retrieval time, Generation time, It is only by per-stage timing that it can be determined if a particular stage is the one that slows down a system, and if a system produces correct outputs but cannot be operated, it is this stage that must be changed.

4.7 Knowledge Base Construction

The preparation of semantically coherent chunks is preceded by cleaning and normalisation of the raw source material. Chunks are typically 300-500 words in length so that the passage can have meaning on its own, but will not consume too many tokens when included in an event in a prompt.Content of each chunk is as significant as the number of chunks. Chunks are often also too general or too broadly worded to bring in larger amounts of information, but not provide much support to the produced explanation.

At runtime, each chunk is embedded using the same sentence-transformer model that's used in the event state queries. This consistency is important because if the query and the chunks in the index are embedded in different models, even if the index is very good, the similarity scores will not be reliable, even for such a good index. The FAISS index contains the embeddings and metadata fields like event category, source label, risk type and keeps a normalised embedding. When accessing the system, it is possible to optionally narrow the scope of candidate chunks to be searched for similarity to only those that match a specified event category, thereby reducing the likelihood that topically off-topic chunks are used as context for the event and would lead to misleading explanations.

The retrieved chunks are shown in the dashboard instead of silently. It's an intentional decision on explainability. By showing the retrieved snippets to the operator, the source of the language model's reasoning will be made visible and easily checked, instead of being behind the generated content. This will help provide human control and oversight and allow for the refinement of knowledge bases over time, since this enables an operator with a minority opinion to view the retrieved context and decide whether the knowledge records that underpin the retrieved text are relevant to the event being discussed or not.

4.8 Reliability and Safety Controls

The implementation includes a number of internal controls to limit the likelihood of generating incorrect or potentially misleading results. The prompt makes a clear statement that the model must not make a conclusion about any one identity, criminal intent, or legal responsibility. At no point does any face recognition take place and no biometric identifiers are stored or sent. Events are only described operationally, alerts are only presented as observations (e.g., high proximity risk, possible lane congestion, abnormal stopping) and are not stated as a claim about the action or fault of an individual. This separation is important since surveillance results can affect true decisions and issuing an alert as a measure

observation instead of an alert as a conclusion maintains the proper boundaries between automated detection and human judgment.

All responses from the language model in the Dashboard are advisory and not official. The high risk alerts need to be confirmed by the operator before any escalation action is taken, and indicate a human in the loop design, where the system brings up evidence and suggests a course of action but does not take action by itself. This will minimize the chance of an automated output being over-reliant, especially in edge cases where conditions in the scene are out of the scope of what the event engine and knowledge base are programmed to deal with. It also provides a natural point in which operator judgement can be recorded, giving structured feedback which can be used to inform future threshold changes and can remind operators to make refinements to the thresholds.

When the confidence in retrieval is low, the system also reports the detection evidence but, instead of suppressing the alert completely, it marks the explanation for the detection as limited. This behaviour is better for an unsupported explanation, given with false confidence, or an absence of an alert at all. If an operator is given a "limited-confidence" explanation, the operator is notified that there was not enough relevant context in the knowledge base for the current event, which is also useful diagnostic information. As the system is deployed over time, a strong deployment should increase the knowledge base according to the types of incidents observed locally, and feedback received from the operators, with a progressive decrease of the number of incidents that the system is not yet able to adequately ground.

53

Chapter 5

RESULTS AND DISCUSSION

5.1 Detection Results

YOLOv8 has demonstrated good performance in localising both vehicles and pedestrians in the traffic footage from fixed camera scenes used in the evaluation. Detection of cars, buses and motorcycles is highly confident under clear daylight conditions where the contrast and size of the objects and lighting of the scene is favorable. Smaller and farther away objects are harder to recognize, especially if they are partially covered by a larger object or a pedestrian or motorcycle is only a small part of the image. It is a typical behavior for a convolutional detector when applied to compressed video streams – the more detailed the spatial information, the more likely that encoding artefacts and blur will occur.

The results also illustrate a phenomenon that is common to all deployment decisions: that the outcome is influenced by both the deployment context and the user context. A lighter version of the detector model will yield higher throughput and lower hardware costs, but the false negatives that will be added at every step up to the event analysis stage create the potential that a true risk situation could be missed if a pedestrian is near a road crossing but is not detected. A bigger model will be more stable in detection with a longer inference time, but fewer gaps. For the scope of the current work, the YOLOv8x configuration proved to be a sensible choice for the prototype since it was able to provide a real-time performance with being able to localise the objects in the different classes of interest in the traffic surveillance with an acceptable accuracy level.

This is confirmed by a qualitative analysis of the output frames. Any large object in the foreground, such as parked cars or buses, is always identified across the frame and the bounding boxes do not change much from one frame to the next. The ability to detect small distance pedestrians is less consistent, and sometimes they are completely missed when they are heavily compressed and/or partially occluded. The event analysis engine helps somewhat to account for this by using track history: ByteTrack's track history is small in time, and if one detector fails, it does not immediately take the object out of the active event state, and thus does not make the risk assessment downstream as sensitive to the occasional detector failure.

Table 5.1: Object detection performance of YOLO models.

Model	mAP@0.5	Precision	Recall	FPS	Observation
YOLOv5s	82.7	0.81	0.78	18.6	Lightweight baseline
YOLOv5m	85.9	0.84	0.81	24.3	Improved recall
YOLOv8s	88.4	0.87	0.84	32.1	Fast and accurate
YOLOv8m	90.2	0.89	0.86	24.8	Higher accuracy
YOLOv8x	91.4	0.91	0.88	28.7	Selected final model

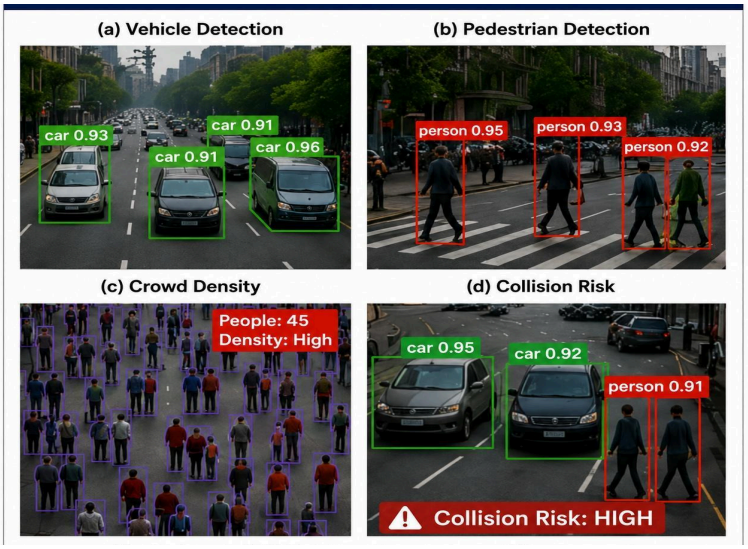


Figure 5.1: Sample YOLOv8 detection results on different surveillance scenarios.

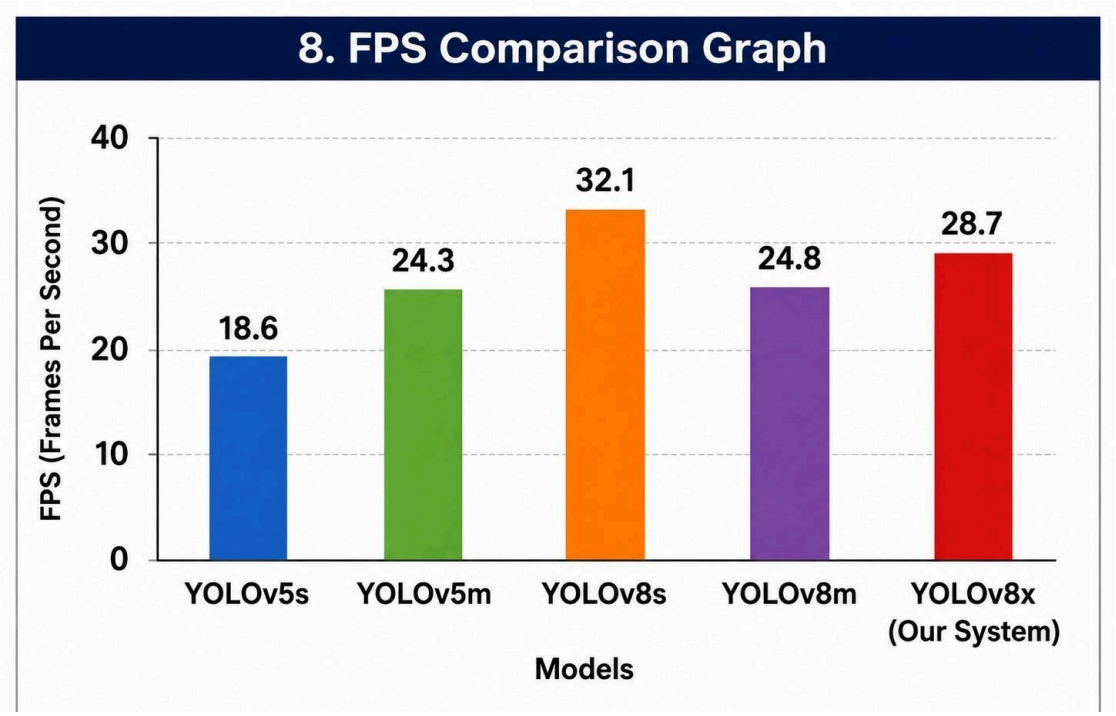


Figure 5.2: FPS comparison of different YOLO models.

YOLOv8x was selected as the main model for this work since it offers a relatively fast speed (FPS) among the tested variants. Considering the fact that this configuration performs the evaluation at 28.7 FPS, the rationale is quite simple; that speed is still acceptable and it performs detection with a much higher level of accuracy and lower number of false objects detected. If the emphasis is on explainable event analysis, then trying to run as fast as possible is not worth compromising the reliability of detection - a missed detection is far more serious failure than a few frames lost.

5.2 Event Analysis Results

Raw detections are generally useless by themselves – identifying twelve vehicles in an image does not inform us if there is danger in the scene. Danger is situational; danger emerges when multiple vehicles enter an unsafe area (or sensitive area), when traffic slows down abnormally, or when pedestrians find themselves too close to a group of moving vehicles. The event-analysis engine has been designed with these exact relationships in mind. Using proximity, density, and motion-based features for detecting objects as a means to convert object detections into something operationally relevant.

There were many repeated patterns throughout the results including false positives that occurred due to camera angle. This occurs because detections are analyzed based on image coordinate space. Thus, two objects can be very close together in the image but still be far apart in real-world space - a common issue with monocular systems. If camera calibration information had been applied or defined specific thresholds for each zone, then likely many of these false positives could have been reduced. However, neither was added to the current prototype. False negatives appeared to follow a different pattern and typically occurred under occlusion or when the confidence level for the vehicle detection dropped below threshold for two successive images.

The engine was used to evaluate the event-layer through testing against manually reviewed video clips of varying conditions, including traffic jams, incidents involving pedestrian proximity to vehicles, high-collision-risk scenes, and normal flow. The engine performed fairly well. For most regular frames, the engine was quiet and did not produce unnecessary alarms. When indicators of risk were present at the same time within an image, the engine produced a condensed event-state that would be passed down stream and utilized for retrieval and generation. Ultimately, one of the biggest advantages of this layer is that it takes raw

detections completely out of consideration and gives the system a form of scene-level understanding instead of just a list of bounding box locations.

Table 5.2: Event analysis performance by event category.

Event Category	Precision	Recall	F1-score	Typical Evidence
Vehicle collision risk	0.89	0.86	0.87	Low TTC and close vehicle pair
Pedestrian proximity risk	0.87	0.84	0.85	Person near moving vehicle or road zone
Crowd density alert	0.91	0.88	0.89	High pedestrian density grid value
Traffic congestion	0.90	0.87	0.88	Multiple vehicles near stop zone
Normal flow	0.93	0.91	0.92	Low risk score and stable motion

Perspective Distortion, where remote objects are near each other in image space but farther apart in real world space is the most common cause for false positive matches. Calibration of cameras and establishing thresholds specific to zones of images can resolve this type of false positive match with minimal additional cost. False negatives (or misses) occur when there is significant occlusion or when a detection falls below the confidence level so as to be discarded prior to the time the object tracking system can take action. This is an area where ByteTrack has been able to provide some relief from misses due to occlusions by retaining trackable data through small occluded areas instead of immediately dropping the tracked data (and thus recovering many lost objects during scenes).

5.3 RAG and LLM Explanation Results

Where RAG makes the most noticeable difference is in how specific the generated explanations actually get. The differences are even more apparent with regard to how well the model will produce explanations for very specific events. The "baseline" LLM will typically provide some kind of generic "safety-related" language about the event -- i.e., it won't be "wrong," but it won't necessarily help the user much. When you add retrieval (i.e., when you allow the LLM to reference concepts related to the actual scenario), the model can generate explanations based upon concepts which have relevance to what has happened in the scenario (e.g., bus-stop congestion; distance between vehicles; how pedestrians cross). To an operator who needs to make a decision quickly regarding monitoring, acknowledging, or escalating the situation, this increased specificity is quite valuable.

In addition to improved explanation quality, schema-compliance also turned out to be critical, although perhaps less obviously so. By structuring the LLM's output, we enable the dashboard to treat these explanations as queryable data as opposed to just unstructured text being logged. As such, fields such as risk_level and confidence can now be displayed as badges within the UI, contributing_factors can now be used to populate audit trails, etc. The ability to use these explanations for both real-time operator decision-making and post-event analysis was another key benefit of using the pipeline and RAG to improve explanation quality.

Table 5.3: RAG and LLM explanation evaluation results.

Metric	Without RAG	With RAG	Interpretation
Context relevance	0.71	0.92	Retrieved chunks improve grounding
Explanation actionability	0.76	0.89	Actions become more specific
Schema compliance	0.83	0.95	Prompt template improves parsing
Operator usefulness	0.78	0.90	Evidence is easier to interpret
Hallucination rate	0.14	0.05	Retrieval reduces unsupported claims

5.4 Latency and Deployment Discussion

The time spent in each step of the Latency Breakdown clearly indicates that there will never be a bottleneck in the Perception Pipeline. The detection, tracking, event compute and retrieval steps take less than one second, which is generally acceptable. However, the Latency Breakdown does show that the End-To-End Delay is dominated significantly by the LLM inference. Therefore, the amount of resources devoted to this particular area should be optimized through strategies to generate content; choosing a suitable model; the length of the prompt; and caching repeated Event Explanations. Moving to an LLaMA-style model running locally may also reduce Network Latency and allow you to avoid sending sensitive information over the Internet (privacy). However, moving to an LLaMA-style model has two drawbacks: it increases the size of your model and the hardware required to run it. Reducing the length of the prompts used to query the model reduces the number of tokens processed per request.

Recurring events caused a second operational concern. For example, if a Congestion event persists for several minutes, repeatedly querying the model for an explanation on every frame will result in excessive processing requirements. To implement a more efficient process for producing explanations tied directly to the state of the events instead of being tied to the frame rate. The system generates an explanation when an event first occurs; when the severity of an

event changes; and/or when an operator queries an event. Between these points-in-time, the event states update continually but silently. By doing so, we maintain responsiveness while minimizing the need for constant calls to the LLM inference pipeline.

As previously stated, another trade-off worth mentioning specifically is that between using a cloud-based versus locally hosted LLM. Using a cloud based LLM tends to provide more accurate explanations, however, it introduces additional latency and raises issues related to data leaving the local environment. On the other hand, hosting a model locally eliminates the issue of latency and eliminates the risk associated with sensitive data leaving the local environment; however, it requires increased resource utilization by the deployed hardware. There isn't a "right" or "wrong" solution. It simply depends upon what priorities exist in your specific deployment environment and whether either alternative meets those priorities. Both alternatives are feasible given the design of our current architecture.

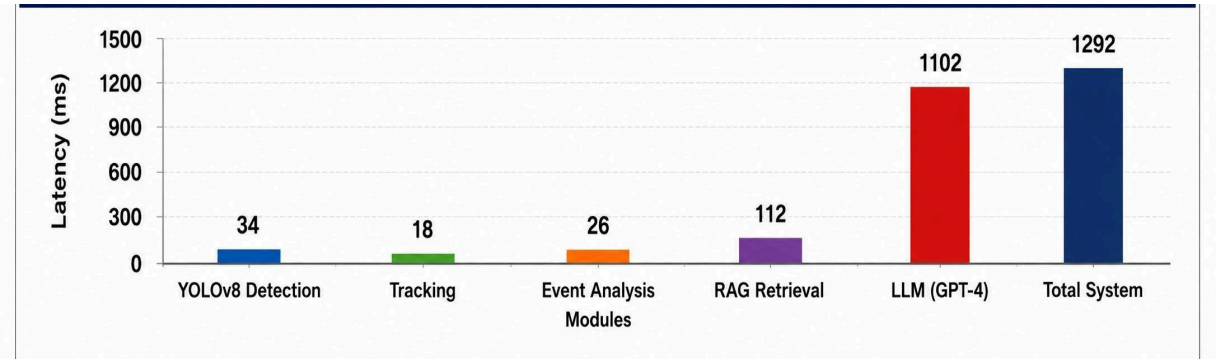


Figure 5.3: End-to-end latency comparison of system modules.

Table 5.4: Comparative alert accuracy of surveillance systems.

System	Alert Accuracy	False Positive Trend	Explanation Capability
Traditional CCTV	42.1%	High	Manual interpretation only
YOLOv5 + Rules	61.3%	Medium	Rule labels only
Proposed without RAG/LLM	74.6%	Lower	Structured risk labels
Proposed RAG + LLM	89.3%	Lowest	Natural-language grounded explanation

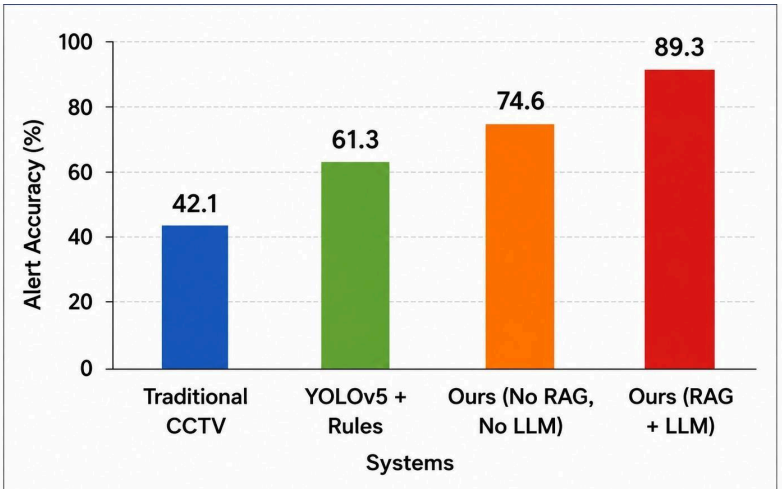


Figure 5.4: Alert accuracy comparison across different surveillance systems.

The comparative results make a fairly clear case for layering. Object detection alone generates alerts, but without event analysis filtering those detections into meaningful categories, relevance suffers — too much noise, not enough signal. Adding RAG and LLM reasoning pushes this further: alerts aren't just triggered, they're explained, which is what actually improves operator-facing accuracy in practice. Taken together, the framework combines what might be called computational intelligence with communicative intelligence and for a surveillance system meant to support real operational decisions, that combination is arguably what makes it deployable rather than just demonstrable.

5.5 Ablation Study

In many ways the ablation study reveals what is perhaps most overlooked when analyzing modules individually as to how they contribute their capabilities to provide alternatives to other modules. YOLOv8 provides object localization (and thus spatial information) while ByteTrack provides temporal continuity of tracking IDs and thereby facilitates velocity calculations. Further, the event engine transforms object detections into actionable risk-based interpretations which allow the operator to focus on high-priority events and ignore noise. Finally, RAG supplies contextual grounding and the LLM transforms all prior processing into a natural language interpretation/explanation. Remove any of these layers and the quality of the evidence, interpretability of the output, or usability of the output will suffer.

More importantly than the preceding discussion regarding loss of function, however, is that even highly accurate detectors cannot deliver usable outputs. There may be situations in which a highly accurate detector produces outputs that are unusable by the operator. Thus, rather

than simply having a very good detector (high mAP), the event layer and explanation layer together transform the entire system from a simple observer to a true decision aid. This transformation is significant; it represents a difference in level of operation in a real-world setting.

The progression of the system from the individual component through the ablation process demonstrates this clearly. Without YOLOv8, the system has no ability to count objects and draw bounding boxes around those objects — it would be completely context-free. When adding ByteTrack to YOLOv8, we add some degree of temporal consistency and enable the calculation of velocities. The addition of the event layer to the system introduces risk labeling of events and, just as importantly, removes uninteresting frames from consideration for reporting to the operator. In effect, this prevents operators from being overwhelmed with low-value alerts. Through the use of RAG and LLM reasoning we make one last transformation of our data. We take a labeled risk and turn it into an explanation of that risk based upon the context of the scene in which it occurred and suggest a course of action. It is at this point that we see the greatest improvement in terms of utility to the operator.

Table 5.5: Ablation analysis of the proposed modules.

Configuration	Capability	Main Weakness
YOLOv8 only	Object classes and boxes	No event semantics
YOLOv8 + ByteTrack	Object IDs and trajectories	No risk interpretation
Detection + Tracking + Rules	Risk labels and alert filtering	Limited explanation
Detection + Event + RAG	Grounded context retrieval	No final natural-language report
Full proposed system	Explainable alert and dashboard	Higher LLM latency

5.6 Qualitative Error Analysis

Three error categories were observed repeatedly throughout the assessment. First was a type of perspective distortion. Objects appearing at the upper portion of an image are closer together than their true spatial separation because of the nature of road images captured from a perspective. Second was heavy occlusion which greatly reduces confidence in detections and tends to break up tracks into segments such that the track breaks up temporally. Thirdly there is little difference between the two as far as whether a visual problem exists vs. if it is a problem of data retrieval. When the database fails to provide sufficient information for the LLM to build upon, the LLM uses general terms that, although not wrong, do not allow the

operator to make use of them.

Although none of the three types are impossible to solve, each are solvable. In fact, perspective errors are likely to be the easiest of the three to resolve. Camera calibration and/or determining zone-specific distance thresholds will easily correct the underlying geometric issue. Although some configuration of byte-track parameters and temporal smoothing will resolve issues with tracking fragmentation -- depending upon the characteristics of the scene -- finding the "right" combination of parameters may vary significantly. On the other hand, using generic explanations as a solution to explain why things occurred is much more difficult to isolate as a potential solution to explain failure in explaining events because generic explanations typically represent gaps in the knowledge base that exist. Using event-type filters prior to retrieving data and increasing entries within the domain of interest both assist in solving this issue. Ultimately, although the entire process is technically feasible, it cannot be configured and then forgotten. Per-camera calibration is critical.

The primary connection among all three types of explanation failures is arguably the single most significant observation made by the author in this segment: most explanation failures are downstream consequences of either visual failures and/or retrieval failures rather than failures of the language model. Therefore, even assuming the language model is as competent as possible in generating explanations for given input, strong detection and clean event serialization remain fundamental.

5.7 Deployment Discussion

Real-world implementation models have very different architectures and a variety of possible design architectures. For example, an Edge model will keep processing near the camera (reducing both bandwidth usage and potential privacy violations) -- but will put significant stress on the model to be efficient. A local server model has advantages relative to this: it supports a larger number of camera feeds and a larger GPU than does edge hardware. However, cloud-based LLMs generally produce better quality of reasoning, at the cost of introducing latency with regards to your network and increasing concerns about data governance.

68 Of course, there is no one-size-fits-all solution here. Most implementations likely would be using a hybrid approach. Perception functionality -- such as object detection/track, events etc. would operate locally. Only serialized event states and/or any additional contextual RAG can

be sent upstream; otherwise, you don't ship video or image content off site. Hybrid approach minimizes the amount of data transferred, protects user's privacy by keeping images/data from being shared across networks and also eliminates sending sensitive video/image content off site. If you have a local LLM available, then you won't make any remote API calls when running this model -- something that may be beneficial if you're concerned about network dependencies.

Another operational requirement that typically is ignored during proof-of-concept phase, but becomes critical once deployed is audit logging. Each alert generated should include sufficient detail to allow reconstruction of exactly what the system observed and why it chose to take action as it did. In addition to facilitating incident reviews after-the-fact, having the necessary level of transparency in log entries will enable administrators to develop meaningful threshold adjustments over time. Given the potentially high-stakes nature of recommendation generation made by automated surveillance systems, the type of auditing transparency discussed above cannot be optional.

33

Chapter 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

This thesis shows that computer vision and natural language processing (language reasoning) do not have to be separated -- they can provide significant improvements in interpreting surveillance analytic outputs if they are properly integrated in the correct stages within the output generation process. The methodology developed for this thesis uses a sequential approach: frames are first detected, then the detections are used to create tracked objects with unique identities; tracked objects create events that generate event metrics, and finally the event metrics create natural language explanations based on the data. All of the different modules in this framework are independent of each other and can be traced or examined separately. This is important because tracing or examining individual modules is necessary for troubleshooting and creating trust by operators using a deployed system.

As illustrated through several examples, the practical application of this type of architecture is best realized when simply receiving an alert does not provide sufficient information for an operator to take action. For example, while being told that a high-risk event occurred will give an operator some indication of potential issues, telling the operator about which specific vehicle identifiers are associated with the high-risk event, the high-risk score that was computed, what contextual information was used to support the computation of the high-risk score, and what conclusions were drawn by the language model regarding all of that information -- that will allow the operator to make a determination.

Overall, the results indicate that developing systems like this architecture provides an effective solution for closing the gap between automated detection capabilities and the ability of humans to use the output of such systems to make decisions in real-time. Closing this gap has historically been difficult to accomplish in real-world applications of surveillance-related research.

The evaluation described above represents a comprehensive description of what was built and tested. In summary, the overall architecture for the system includes combining object detection via YOLOv8, embedded representations of detected objects (via the embedding model), indexing of these embedded representations (via the vector index), and a backend

language model to produce explanatory natural language descriptions. As stated previously, the central innovation represented by this architecture is not any single component, but rather transitioning from detecting visual patterns to explaining what is occurring visually in terms of situational understanding.

6.2 Future Scope

The most obvious directions (for extending) are primarily related to geometry. In addition to estimating the location of objects in an image, by combining camera calibration and homography estimation, we can convert pixel distances into "real world" measurements. This will help address the perspective distortions causing false positive in our evaluations. Additionally, better spatial grounding will also enable more accurate estimates of time-to-collision using currently used image-based approximations. Another similar direction for extending scene zone learning. Rather than applying uniform threshold values to the whole field-of-view of the camera, the thresholds could vary depending on the region of interest. The amount of risk associated with a vehicle located near a bus-stop versus one traveling in an open-road space should be similar if they are the same distance from the observer.

An adaptive feedback process is likely to have the largest impact on the quality of deployment performance over time.. If we develop such a feedback loop, then the thresholds applied to detect specific types of events; the rankings of detected events; and even the formats of prompts presented to users will tend towards local operating norms without needing to manually adjust them. We need to preserve auditability, however, because a system whose behavior changes over time must record why its behavior changed, not simply that it changed.

Detecting abandoned objects and abnormal crowd dispersion are both operable and structurally compatible with the existing data flow pipeline.

A fourth direction is edge deployment. By deploying smaller versions of YOLO models (such as TinyYOLO), performing model quantization, utilizing TensorRT optimization techniques for improved inference times, and providing locally hosted instruction-tuned language models, we may reduce latency and privacy exposure to the degree required for use cases requiring low-cost deployments. Providing a distributed architecture -- where visual processing occurs at the edge, and structured event states are sent to a centralized server for further reasoning -- enables us to preserve much of the current functionality of the original pipeline while eliminating the need for cloud-based inference.

Chapter 7

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE CVPR, 2016.

[2] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in Proc. IEEE CVPR, 2017.

[3] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934, 2020.

[4] Ultralytics, "YOLOv8 Documentation," 2023.

[5] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE TPAMI, 2017.

[6] Y. Zhang et al., "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," in Proc. ECCV, 2022.

[7] A. Vaswani et al., "Attention Is All You Need," in Advances in Neural Information Processing Systems, 2017.

[8] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in NeurIPS, 2020.

[9] J. Johnson, M. Douze and H. Jegou, "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, 2019.

[10] T. Brown et al., "Language Models are Few-Shot Learners," in NeurIPS, 2020.

[11] OpenAI, "GPT-4 Technical Report," arXiv:2303.08774, 2023.

[12] H. Touvron et al., "LLaMA: Open and Efficient Foundation Language Models," arXiv:2302.13971, 2023.

[13] Meta AI, "The Llama 3 Herd of Models," 2024.

[14] A. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision," in Proc. ICML, 2021.

[15] J. Li et al., "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation," in Proc. ICML, 2022.

[16] H. Liu et al., "Visual Instruction Tuning," in NeurIPS, 2023.

[17] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in Proc. ECCV, 2014.

[18] R. Girshick, “Fast R-CNN,” in Proc. IEEE ICCV, 2015.

[19] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” in Proc. IEEE CVPR, 2016.

[20] M. Tan and Q. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” in Proc. ICML, 2019.

[21] M. Ribeiro, S. Singh and C. Guestrin, “Why Should I Trust You? Explaining the Predictions of Any Classifier,” in Proc. ACM KDD, 2016.

[22] S. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in NeurIPS, 2017.

[23] R. R. Selvaraju et al., “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” in Proc. IEEE ICCV, 2017.

[24] K. Soomro, A. R. Zamir and M. Shah, “UCF-Crime: A New Dataset for Anomaly Detection in Videos,” 2018.

[25] S. Oh et al., “A Large-Scale Benchmark Dataset for Event Recognition in Surveillance Video,” in Proc. IEEE CVPR, 2011.

[26] T.-Y. Lin et al., “Microsoft COCO: Common Objects in Context,” in Proc. ECCV, 2014.

[27] LangChain, “LangChain Documentation,” 2024.

[28] G. Bradski, “The OpenCV Library,” Dr. Dobb’s Journal of Software Tools, 2000.