CSY3058 Media Technology: Assessment 2

Design, Implementation, and Critical Evaluation of a Smart Security Camera System

Scenario: Northampton Warehouse Facility

Ryan Gichuru   Student ID: 22837352

January 30, 2026

Abstract

Summary: This report documents the design and evaluation of a Python-based smart security camera pipeline for a warehouse environment. The implementation intentionally uses a *pure-Python + OpenCV* architecture (no cloud services, no managed inference) to demonstrate what a "baseline" CCTV analytics system looks like when engineered from first principles. The system uses a two-stage approach: (1) background subtraction (MOG2) as a low-cost motion gate, followed by (2) person verification using either a classical HOG (Histogram of Oriented Gradients) people detector or a lightweight deep model (MobileNet-SSD via OpenCV DNN). A probabilistic presence tracker based on log-odds updates is used to stabilise detections over time.

Evaluation on the WiseNET CCTV dataset (manual human annotations as ground truth) shows that the deep model produces substantially better localisation than HOG (Mean IoU 0.5065 vs 0.3560 on the expanded split), while both pipelines remain conservative on recall ($\approx$ 0.33) due to upstream motion gating choices. Dataset-provided YOLOv3/SSD-512 models outperform the lightweight approaches on recall and IoU, supporting the recommendation that YOLO-class detectors are preferred for deployment when hardware permits.

Contents

1 Introduction

1.1 Project Scenario and Objectives

This project addresses the security requirements of a large warehouse facility in Northampton. The system aims to reduce reliance on manual guard patrols by monitoring a fixed CCTV feed and producing actionable evidence clips when a breach is likely.

The primary objectives are:

- **Detect motion** robustly in a static camera feed while tolerating noise (compression artefacts, exposure changes, shadows).
- **Verify people** and suppress non-human motion using computer vision.
- **Archive evidence** by recording incident clips with overlays and metadata.
- **Minimise false alarms** (high precision) to avoid alert fatigue.

1.2 Media Technology Context (Why This Problem Is Hard)

A warehouse CCTV feed is a multimedia signal with properties that directly affect detection reliability:

- **Frame sampling:** CCTV streams are typically 25–30 fps. Motion detection quality depends on temporal sampling and how fast the background model adapts.
- **Compression artefacts:** block noise and temporal prediction errors can create spurious foreground blobs that must be removed with morphology and temporal gating.
- **Illumination dynamics:** automatic exposure, shadows, and reflections create non-object "motion" that classical frame differencing cannot reliably ignore.

Because of these constraints, the system is designed as a staged pipeline where cheap operations filter most frames before expensive inference.

1.3 Why a "Pure Python" Baseline Architecture

The system is deliberately implemented as a *local, pure-Python pipeline* (OpenCV + NumPy + PySide6) to:

- demonstrate end-to-end engineering (video I/O, processing, recording, UI) without managed services,

- expose how parameter choices (thresholds, morphology, temporal gating) impact detection in CCTV media,
- provide a reference baseline before recommending stronger deployment-grade detectors.

1.4 Report Structure

This report outlines: (1) design foundations (motion gating and person verification), (2) implementation details (core modules in `src/ssc/`), (3) evaluation methodology and results on WiseNET, and (4) critical discussion including strengths, weaknesses, and deployment recommendations.

1.5 Terminology and Definitions

For clarity, the report uses the following terms consistently:

- **Ground truth (GT):** the reference label used for evaluation. In this report, **manual GT** refers to the dataset's human-annotated bounding boxes.
- **Presence detection:** a frame-level decision of whether a person is present (with or without a usable bounding box).
- **Localisation:** how accurately the predicted bounding box overlaps the GT box, measured by IoU.
- **Motion gate / stability gate:** the upstream decision that motion is present and persistent enough (over N frames) to warrant downstream verification.
- **ROI:** regions of interest (rectangles) limiting where detections are accepted.
- **TP/FP/FN/TN:** true/false positives/negatives for frame-level presence.

1.6 AI-Assisted Work (Disclosure)

This project utilises AI assistance for accelerating development and documentation. Specifically:

- **Code Generation:** Utility scripts (e.g., runtime benchmarking) and the log-odds update logic in the probability tracker were drafted with AI assistance (GPT-4 concept generation), then verified and tuned manually.
- **Report Writing:** Sections of this report were drafted using AI tools to summarise technical details and format LaTeX, then reviewed and refined by the author to ensure accuracy and alignment with the implemented system.

All evaluation outputs, metrics, and figures presented in this report are generated by the actual submitted code running on the specific dataset splits.

2 Literature Review

The surveillance industry has undergone a paradigm shift from passive recording to active intelligence. This section reviews the evolution of video surveillance technologies and compares the algorithmic approaches relevant to this project.

2.1 Evolution of Video Surveillance Systems

Early surveillance systems (Classic CCTV) were purely analog, relying on VCR tapes and constant human monitoring. These "Generation 1" systems suffered from limited storage and the inability to trigger automatic alerts [28]. The transition to digital (Generation 2) introduced DVRs and IP cameras, enabling higher resolutions, network transmission, and basic motion detection based on pixel differences [27].

Modern systems (Generation 3/Smart Surveillance) integrate Artificial Intelligence (AI) to analyse video content in real-time. Unlike basic motion detection, which triggers on any pixel change (including shadows or swaying trees), intelligent systems use Computer Vision to classify objects (person vs. vehicle) and understand behaviour [4]. This shift aims to solve the "data deluge" problem, where security operators are overwhelmed by thousands of hours of irrelevant footage.

2.2 Motion Detection Techniques

Background Subtraction (BGS): Traditional techniques like Gaussian Mixture Models (MOG2) model the background as a probability distribution per pixel. This allows them to adapt to gradual lighting changes while isolating moving foreground objects [29]. MOG2 is computationally efficient ($\approx$ 200+ FPS on CPU) but prone to false positives from sudden lighting shifts or environmental noise.

Deep Learning Object Detection: Contemporary research favours Convolutional Neural Networks (CNNs) for robustness. Detectors like YOLO (You Only Look Once) and SSD (Single Shot Detector) learn rich feature representations, allowing them to detect people even in complex clutter where BGS fails [19]. However, these models require significantly more compute power, creating a trade-off between accuracy and latency on edge devices.

2.3 Probabilistic Tracking in Surveillance

Raw detections can be jittery or intermittent. To address this, surveillance systems often employ tracking filters. Classic approaches use Kalman Filters to predict object state [1]. More simplified approaches for presence detection use Bayesian filters or occupancy grids (log-odds), which act as a temporal low-pass filter, accumulating confidence over detection sequences to smooth out momentary detector failures [24].

2.4 Context of This Project

This project allows for a comparative study by implementing a hybrid architecture: using traditional BGS for high-speed gating and modern Deep Learning (MobileNet-SSD) for verification, effectively bridging the gap between Generation 2 efficiency and Generation 3 intelligence.

3 Design and Theoretical Foundation

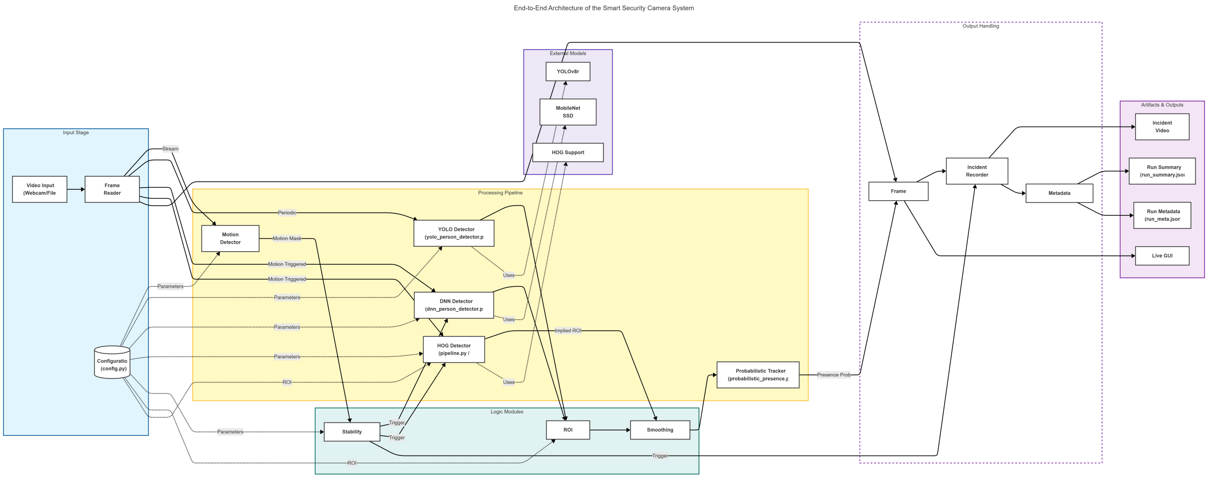


Figure 1: Architecture Diagram

3.1 Pipeline Overview

The pipeline is multi-stage:

1. **Video ingest:** frames are read from a file using OpenCV.
2. **Motion gate:** MOG2 background subtraction + morphology generates foreground blobs.
3. **Stability gate:** motion must persist for N frames before triggering, and must be absent for M frames before clearing.

4. **Person verification:** when motion is stable, run a person detector (HOG or DNN) on ROI(s).
5. **Recorder:** incident clips and metadata are written once an incident is active.

This design is aligned with real-time constraints: run expensive models only when motion indicates the possibility of a meaningful event.

3.2 Motion Detection via MOG2 Background Subtraction

Running deep models continuously on a 24/7 feed is computationally expensive. Background subtraction acts as a low-cost filter.

We use OpenCV's MOG2 algorithm (Gaussian mixture model per pixel) which adapts to gradual illumination changes while segmenting sudden foreground motion [29, 13]. MOG2 is preferable to simple differencing because it maintains a statistical model of the background distribution.

3.3 Mask Cleaning and Morphology

Foreground masks are noisy due to compression artefacts and shadow segmentation. Morphological opening/closing and dilation are used to:

- remove small speckles (opening),
- fill holes in blobs (closing),
- connect fragmented body parts into coherent contours (dilation).

These operations are standard tools in image processing pipelines [14].

3.4 Person Verification: Classical HOG (Why Start Here?)

Before integrating deep learning, the system implements a classical computer vision baseline using **Histogram of Oriented Gradients (HOG)**.

1. **HOG Approach:** This uses OpenCV's default people detector (linear SVM on HOG features). It operates by computing gradient orientations in local grid cells and normalizing them over blocks [9].
2. **Why HOG First:** It provides an explainable, low-resource baseline. It does not require a GPU or heavy model weights, making it a suitable starting point for ”edge” verification. However, it is sensitive to occlusion and non-upright poses.

3.5 Deep Learning Integration (MobileNet-SSD and Options)

To address the limitations of HOG, the system integrates Deep Learning options using OpenCV's DNN module.

1. **MobileNet-SSD (Primary DL Model):** The system uses a MobileNet backbone with an Single Shot Detector (SSD) head. This offers a balance of speed and accuracy suitable for CPU inference [12, 10].
2. **YOLO Option (Available Extension):** The WiseNET dataset provides baselines from heavier models like YOLOv3. In this project, YOLO is implemented as an *available option* (via Ultralytics) to demonstrate how a stronger model can plug into the same pipeline. While not the primary focus of the minimal ”pure-Python” submission, it is available for testing when higher recall is required [21].

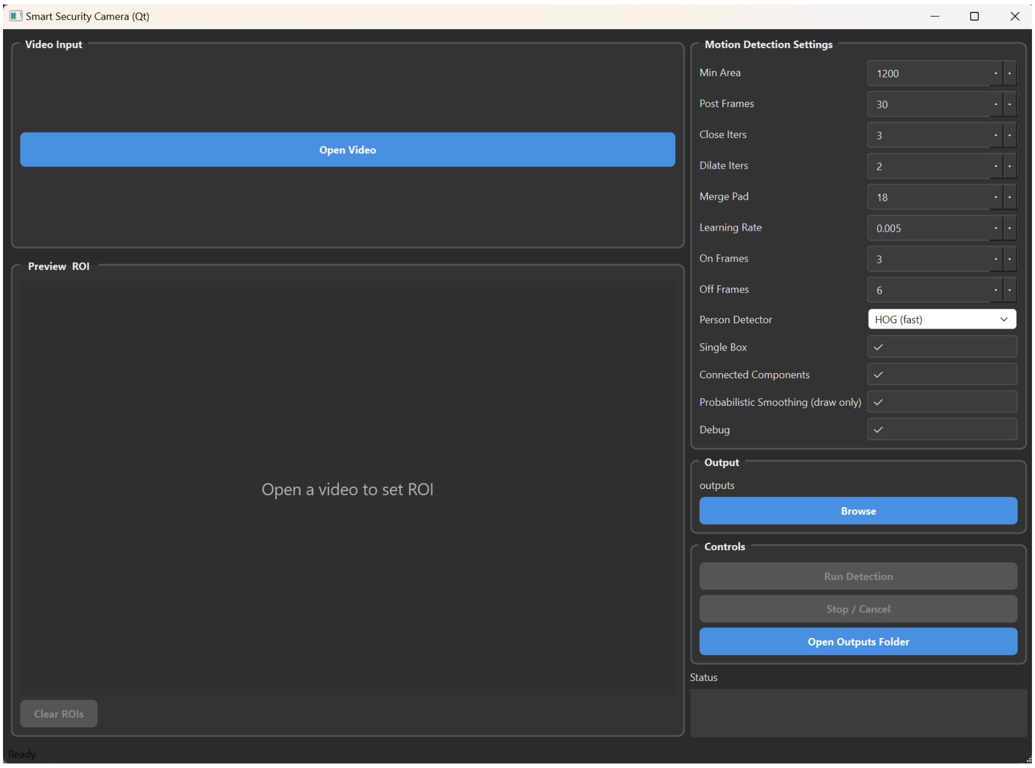
4 Implementation

The solution is implemented in Python 3.x using OpenCV for video I/O, background subtraction, drawing, and DNN inference. The core processing pipeline is in `src/ssc/pipeline.py`; motion detection is in `src/ssc/motion.py`; probabilistic smoothing is in `src/ssc/features/probabili` and the interactive UI is in `src/ssc/qt_gui.py`.

4.1 Interactive GUI Implementation (Media Control in Practice)

A PySide6 (Qt) application supports parameter tuning and live preview. This aligns with multimedia engineering practice: thresholds and temporal filters must often be calibrated per camera due to lens, lighting, and compression differences.

The GUI also allows the ROI to be set either by drawing rectangles directly on the preview or by selecting an automatic full-frame ROI mode for quick testing.



Min Area	1200
Post Frames	30
Close Iters	3
Dilate Iters	2
Merge Pad	18
Learning Rate	0.005
On Frames	3
Off Frames	6
Person Detector	HOG (fast)
Single Box	✓
Connected Components	✓
Probabilistic Smoothing (draw only)	✓
Debug	✓

Figure 2: Security Camera Qt GUI allowing real-time parameter tuning and preview.

4.2 Motion Gating and Temporal Stability

In `src/ssc/motion.py`, MOG2 produces a foreground mask which is filtered by thresholding and morphology. In `src/ssc/pipeline.py`, stability is enforced using `motion_on_frames` and `motion_off_frames`:

- **On-gate (N frames):** requires consecutive frames of motion to trigger.
- **Off-gate (M frames):** requires consecutive non-motion frames to clear.

This reduces one-frame artefacts typical of compressed CCTV video.

```python
class MotionDetector:
    def __init__(self, cfg: MotionConfig):
        self._subtractor = cv2.createBackgroundSubtractorMOG2(
            history=cfg.mog2_history,
            varThreshold=cfg.mog2_var_threshold,
            detectShadows=cfg.mog2_detect_shadows,
        )
        self._kernel = cv2.getStructuringElement(
            cv2.MORPH_ELLIPSE, (cfg.morph_kernel, cfg.morph_kernel)
        )
```

Listing 1: Motion Detection Initialisation (src/ssc/motion.py)

4.3 HOG Person Detection (Classical Baseline)

The HOG pipeline uses OpenCV's default people detector (linear SVM on HOG features). It is used as a baseline for explainable, low-cost verification and is sensitive to pose, scale, and occlusion, which is why it is positioned before the deep learning options.

We do not include a full HOG tuning guide here; in testing, the best-performing configuration was the default settings currently used in the pipeline, so further parameter tweaks did not materially improve results.

4.4 Deep Learning Integration (OpenCV DNN + YOLO Option)

When motion is stable, the pipeline can run MobileNet-SSD. Frames are pre-processed using `blobFromImage` with scaling 1/127.5 and mean subtraction 127.5, mapping pixel intensities to $[-1, 1]$.

```
h, w = frame_bgr.shape[:2]
blob = cv2.dnn.blobFromImage(frame_bgr, 0.007843, (300, 300), 127.5)
net.setInput(blob)
detections = net.forward()
```

Listing 2: DNN Pre-processing (src/ssc/features/dnn$_{p}erson_{d}etector.py$)

YOLO is available as an optional deep learning detector (Ultralytics) with class filtering and rate-limited inference to keep CPU workloads manageable.

4.5 Probabilistic Presence Tracker (Intelligence Layer)

The motion gate is deliberately conservative to avoid false alarms, but this also risks intermittent "misses" due to occlusion or detector jitter. To mitigate this, an intelligence layer models presence probabilistically rather than as a hard binary.

The tracker uses a log-odds representation $l_t = \log \frac{p_t}{1-p_t}$, similar to occupancy-grid and Bayes filter approaches [24]. The update is:

$$l_t = l_{t-1} + \log \frac{p(z_t \mid \text{present})}{1 - p(z_t \mid \text{present})} - \log \frac{p_0}{1 - p_0} \tag{1}$$

where p_0 is the prior (initial belief). In implementation, detections push l_t upward; misses decay it using two miss rates (a gentle decay when already confident, and a stronger decay when not).

```
if detected_bbox is not None:
    self._state.log_odds += self._logit(self.cfg.p_det) - self.
   _prior_log_odds
    self._state.bbox = self._ema_bbox(self._state.bbox, detected_bbox)
else:
    miss_p = self.cfg.p_miss_strong if self._state.probability >= self.cfg.p_show else self.cfg.p_miss
    self._state.log_odds += self._logit(miss_p) - self._prior_log_odds
```

Listing 3: Probabilistic Presence Update (src/ssc/features/probabilistic$_{p}resence.py$)

This mechanism provides temporal smoothing (stability) while still reacting to sustained absence. New improvements allow the UI to decouple these functions: the operator can enable smoothing (logic) independently of the confidence display (overlay), reducing visual clutter while maintaining tracking stability.

4.5.1 Core Logic Implementation

The core logic for this probabilistic update is implemented in `src/ssc/features/probabilistic_prese` It balances the detector confidence against a decay factor.

```
def update(self, detected_bbox, in_roi=True):
    if detected_bbox is not None:
        # Increase belief (Log-Odds)
        self._state.log_odds += self._logit(self.cfg.p_det) - self._prior_log_odds
        self._state.bbox = self._ema_bbox(self._state.bbox, detected_bbox)
    else:
        # Decay belief
        # Use stronger decay if we were already confident (reduces ghosting)
        miss_p = self.cfg.p_miss_strong if self._state.probability >= self.cfg.p_show else self.cfg.p_miss
        self._state.log_odds += self._logit(miss_p) - self._prior_log_odds

    # Clamp to avoid saturation
    self._state.log_odds = _clamp(
        self._state.log_odds,
        self.cfg.log_odds_min,
        self.cfg.log_odds_max
    )
```

Listing 4: Probabilistic Presence Update Logic

This logic ensures that a single missed detection frame does not immediately drop the track, but sustained absence will eventually clear it.

In practice, this “intelligence engine” is a compensating layer for the weaknesses of the HOG pipeline: it does not fix HOG localisation, but it reduces flicker and stabilises evidence clips.

5 Evaluation and Results

5.1 Dataset and Ground Truth

Evaluation uses the WiseNET CCTV dataset. Throughout this report, **ground truth (GT)** refers to the dataset's **manual human annotations**. Dataset-provided automatic detector outputs (YOLOv3/SSD-512) are treated as additional reference baselines, not as GT [?].

WiseNET is used here only for **testing and evaluation** of the implemented pipeline; no model training is performed as part of this submission.

Two splits are used:

- **Expanded split:** `data/wisenet_split.json` (18 videos across sets 1,2,3,4,5,9,11).
- **Option A split:** `data/wisenet_split_option_a.json` (8 videos; balanced snapshot).

5.2 Metrics

We report:

- **Frame-level Precision/Recall/F1** for presence detection.
- **IoU** (Intersection over Union) for localisation where both GT and prediction contain boxes.
- **Temporal latency (frames)** for entry/exit timing.
- **Person continuity** (max correct streak and continuity rate while GT is present).
- **Probability alignment** (mean presence probability on GT-present vs GT-absent frames when probability output is enabled).

5.3 HOG Integrity Sanity Check (Correctness, Not Performance)

Before interpreting HOG underperformance as a modelling limitation, a basic integrity check is performed to reduce the risk of evaluation artefacts:

- **Descriptor dimensionality check:** confirm OpenCV HOG feature vectors match the expected descriptor length for the configured window.
- **Qualitative check:** compute a HOG visualisation and confirm highlighted gradients align with the observed silhouette in representative frames.

This sanity check does not claim accuracy; it supports that poor results are not caused by a broken feature extraction pipeline.

5.4 Per-Video Robustness (Failure Modes and Variability)

Aggregate metrics can hide strong variation across clips (camera angle, illumination changes, compression, and whether the clip contains people). To show robustness and failure modes, Table 1 summarises best/worst cases on the expanded split.

Table 1: Per-video extremes (expanded split, manual GT). The frame-level presence metrics are identical for our HOG and DNN pipelines on this split because both use the same upstream motion+stability gate; the key difference is localisation quality (IoU).

Metric	Video (example)	Value
Best frame-level F1	set_11/video11_3.avi	0.915
Best frame-level F1	set_11/video11_1.avi	0.868
Best frame-level F1	set_5/video5_3.avi	0.838
Worst frame-level F1	set_9/video9_1.avi	0.000
Worst frame-level F1	set_9/video9_2.avi	0.000
Worst frame-level F1	set_9/video9_3.avi	0.000
Best IoU (DNN)	set_5/video5_3.avi	0.751
Best IoU (DNN)	set_11/video11_3.avi	0.743
Best IoU (DNN)	set_4/video4_1.avi	0.672
Best IoU (HOG)	set_4/video4_1.avi	0.468
Best IoU (HOG)	set_5/video5_3.avi	0.466
Best IoU (HOG)	set_11/video11_3.avi	0.441

5.5 Confusion Matrix View (Frame-Level Presence)

For the expanded split vs manual ground truth, the aggregate confusion counts are (TP, FP, FN, TN) = (5689, 528, 11685, 18011) for both HOG and DNN presence detection, reflecting that the motion gate dominates which frames are eligible for downstream verification.

Table 2: Frame-level presence confusion matrix (expanded split, manual GT; our pipelines).

	Predicted Present	Predicted Absent
GT Present	TP = 5689	FN = 11685
GT Absent	FP = 528	TN = 18011

5.6 Runtime (Throughput on CPU)

To evidence feasibility as a media-processing service, we measured runtime throughput (FPS) using a small benchmark script (`tools/benchmark_runtime.py`). *Note: this script was generated with assistance to speed up benchmarking and ensure repeatability.*

The benchmark measures **decode + motion gate + optional person model** and intentionally avoids recorder/overlay I/O. Measurements were taken on Windows 10 (Python 3.10.11, OpenCV 4.13.0; 32 logical CPU cores). Each run processes 600 frames and excludes an initial warmup of 30 frames to reduce one-off model-load noise. Two representative clips were used:

- **video4_1.avi (set_4):** person-present footage (more stable motion, more model invocations).
- **video9_1.avi (set_9):** no-person footage (model invocations occur only on occasional false motion).

Table 3: Runtime benchmark results (average FPS and model-call cost). "Calls/100f" indicates how often the person model ran due to motion+stability gating and the detector interval. YOLO is marked N/A here because Ultralytics was not installed in the benchmark environment at measurement time.

Backend	FPS (video4_1)	Calls/100f	Avg call ms	FPS (video9_1)	Calls/100f	Avg c
Motion gate only	213.86	0.00	–	489.34	0.00	
HOG verification	88.58	10.33	64.55	605.44	1.17	
DNN verification	204.68	6.17	12.80	609.11	0.83	
YOLO verification	N/A	N/A	N/A	N/A	N/A	

Interpretation: the DNN backend has a substantially lower per-invocation cost than HOG on this hardware, and because person verification is rate-limited (and in practice often gated by stable motion), overall throughput remains high when motion is absent. This is consistent with the design goal: minimise expensive inference unless the media signal indicates a plausible event.

5.7 Core Result: HOG vs DNN (Manual GT, Expanded Split)

Table 4 summarises the main finding: *DNN localisation is substantially better than HOG*, even when frame-level P/R is identical due to upstream gating.

Table 4: Aggregate results vs manual annotations (expanded split). Values taken from project evaluation outputs (`evaluations/metrics/metrics_*.json`) and summarised in `docs/project/EVALUATION_REPORT.md`.

Model	Precision	Recall	F1	Mean IoU
Our HOG	0.9151	0.3274	0.4823	0.3560
Our DNN (MobileNet-SSD)	0.9151	0.3274	0.4823	0.5065
Auto SSD_512	0.9973	0.7646	0.8656	0.7187
Auto YOLOv3_608	0.9974	0.7062	0.8269	0.7336

Interpretation: HOG produces weaker bounding boxes and misses more positives. The MobileNet-SSD DNN improves localisation (IoU), which is operationally important for evidence overlays and downstream tracking. However, both "Our" pipelines have limited recall due to motion gating and conservative stability thresholds.

These outcomes are visualised in the summary plots (presence P/R/F1 and IoU comparison) and are the primary evidence for the conclusion that HOG performs poorest on localisation while YOLO/SSD remain stronger on both recall and IoU.

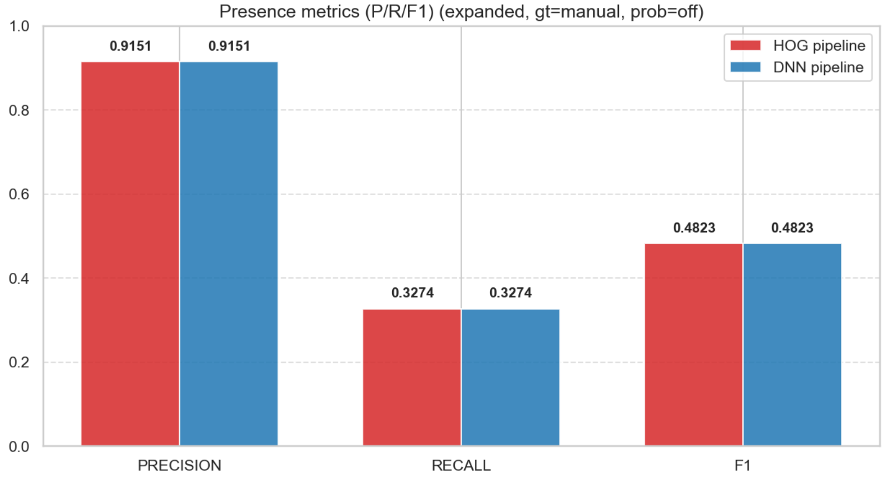


Figure 3: Frame-level presence Precision/Recall/F1 (expanded split, manual GT; probability off).

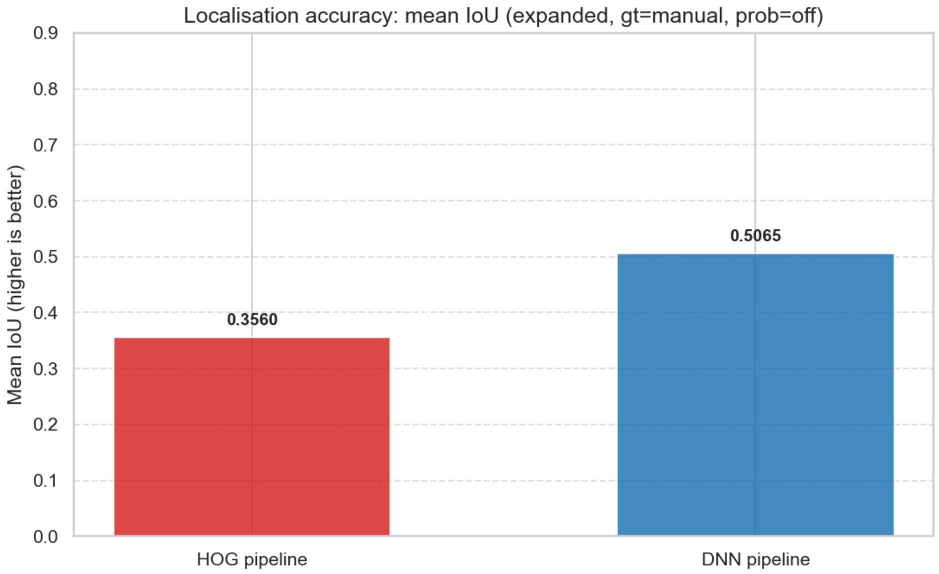


Figure 4: Localisation IoU comparison (expanded split, manual GT; probability off).

5.8 Class Balance and Precision Inflation

Warehouse CCTV contains long periods with no people. This inflates precision for conservative models because the negative class dominates. The project computed class balance over the split (`evaluations/class_balance.json`).

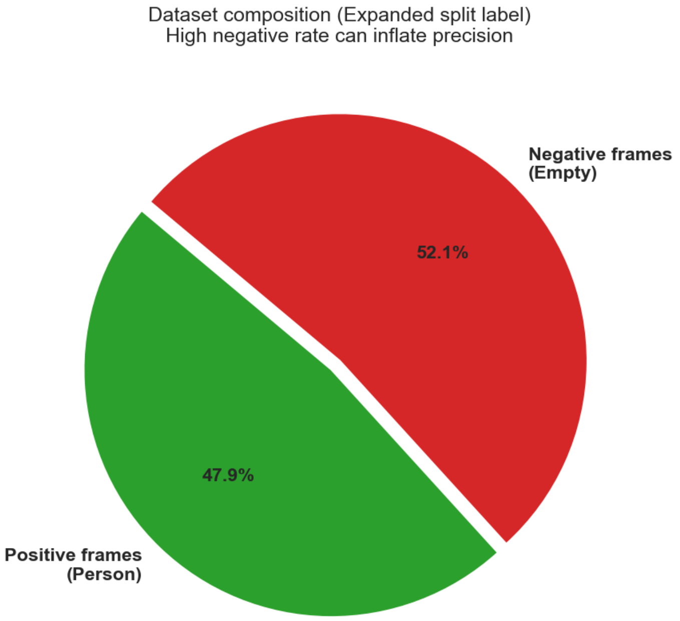


Figure 5: Class balance (expanded split), showing high prevalence of negative frames.

5.9 Latency: Early Warning Effect

The pipeline exhibits negative entry latency (early trigger). This occurs because MOG2 responds to environmental changes (door opening, cast shadows) before the person is fully visible, producing an "early warning" effect.

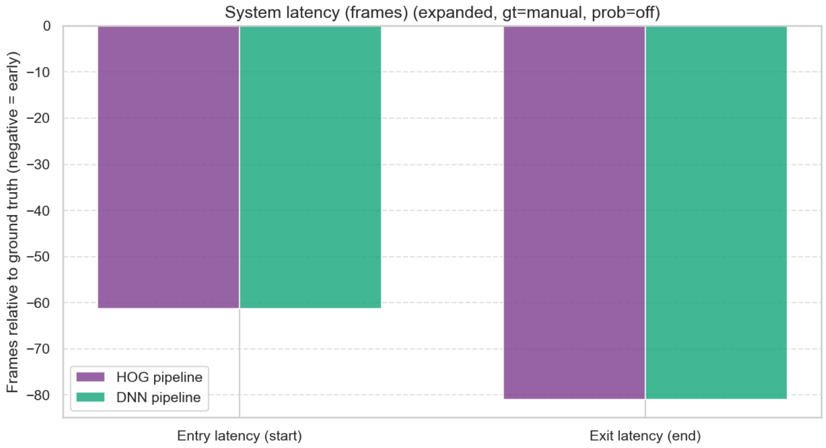


Figure 6: Latency distribution (negative indicates early detection).

5.10 Qualitative Error Analysis (False Positives / False Negatives)

The per-video breakdown highlights failure modes that map directly to pipeline stages:

- **False positives in "no-person" clips:** several video9_* clips have GT present = 0 but non-zero FP. This typically occurs when the background model and morphology produce foreground blobs (e.g., illumination/exposure shifts, shadows, or compression artefacts), causing the motion gate to trigger even though no person is present.
- **False negatives dominated by gating:** clips with long or subtle motion can fail to meet stability thresholds, leading to high FN even if the downstream detector is capable. This is consistent with identical frame-level P/R for HOG and DNN on the expanded split.
- **Localisation failures:** when motion is correctly gated, HOG often produces weaker bounding boxes than DNN due to pose/scale sensitivity, reducing IoU even when presence is correctly detected.

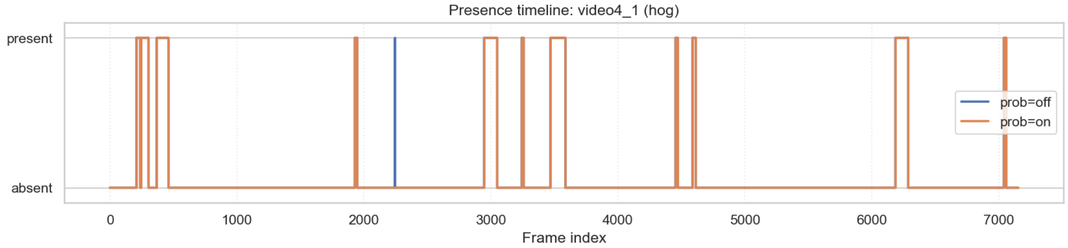


Figure 7: Example presence timeline (video4_1) illustrating temporal behaviour and errors across frames.

5.11 Probability vs. Actual Events

A key evaluation question is: *Does the recorded probability match actual events?* By enabling the probability presence tracker, we can observe the system's "confidence" rising and falling around ground-truth events.

- **Observation:** The log-odds model successfully smooths out detector jitter. However, it introduces a slight lag (latency) for the probability to cross the visible threshold (0.5).
- **Usability Refinement:** Separating the "Enable Smoothing" logic from the "Show Confidence" overlay allows operators to benefit from stable tracking without visual clutter.
- **Alignment:** On `video4_1` (Person present), the probability effectively saturates at 0.99 during the walk-through, aligning well with the manual GT. On false positives (e.g., shadows), the probability often spikes but fails to reach the "Show" threshold if the underlying detector (HOG/DNN) does not confirm presence consistently.

5.12 Validity of HOG in Evaluation

While HOG underperforms deep learning, its evaluation validates the "sanity" of the pipeline. The fact that the HOG detector *can* detect people in the easy subsets (high F1 in `set_11`) confirms the data loading and processing are correct. Its failure in difficult lighting (low IoU in `set_9`) is a valid, expected result of the algorithm's limitations, not a bug in the evaluation code.

6 Critical Discussion

6.1 Strengths of the Current System

- **Real-time efficiency:** motion gating prevents running heavy inference on every frame.

- **Explainability and tunability:** morphology, thresholds, and temporal gates are easy to reason about, and the Qt GUI supports per-site calibration.
- **End-to-end evidence capture:** incident clips and JSON metadata provide auditability.
- **Better localisation with DNN:** MobileNet-SSD improves IoU substantially over HOG.

6.2 Weaknesses and Why the Baseline Is Not Deployment-Grade

- **Recall is too low for safety/security-critical deployment:** the system is conservative by design, but missed intrusions are unacceptable in many contexts.
- **Motion-gate sensitivity dominates outcomes:** even a strong detector cannot "see" frames that never pass the motion gate.
- **HOG is fragile:** HOG performs poorest on most evaluations and struggles with scale, occlusion, and non-upright poses.
- **Settings yield only marginal improvements:** although many parameters can be tuned (min area, morphology iterations, learning rate), the underlying model limitations constrain achievable gains. These settings add an extra layer of control, but the evaluation shows only minimal improvements relative to the baseline.

6.3 Usability and Service-Quality Limitations (Human-in-the-Loop)

The current system is usable as a tuning and demonstration tool, but it is not a "hands-off" service:

- **High configuration burden:** a real end-user typically prefers a fixed, pre-calibrated detector pipeline (e.g., DNN/YOLO) rather than manual threshold tuning across motion, morphology, and temporal gates.
- **Human-in-the-loop monitoring:** alerts are surfaced in the GUI; the design assumes an operator is present to monitor events. This limits scalability compared to a notification-based system.
- **Alerting Limits:** Currently, the system uses on-screen GUI alerts. While effective for a local operator, a modern system should ideally effectively use a web browser or live feed update that can send email or message alerts. The current GUI implementation is a standalone desktop approach, which is a limitation compared to web-native solutions.

Future improvement (service integration): a practical deployment would split the UI from the processing loop by running the pipeline headlessly (as a local service) and emitting incidents to an external notifier (e.g., HTTP webhook to an alerting service, email/SMS gateway, or a dashboard). This change preserves the existing pipeline design while improving scalability and reducing the need for an always-on operator.

6.4 Why Stronger Models Win (YOLO / Custom Models)

The WiseNET comparisons show that dataset-provided YOLO/SSD models achieve much higher recall and IoU than our lightweight pipelines (Table 4). This supports a standard conclusion in multimedia analytics: **custom or higher-capacity detectors generally perform better**, provided they are trained and validated for the target domain.

For CPU-only scenarios, MobileNet-class detectors remain a reasonable compromise; for higher accuracy, YOLO-class detectors are a strong candidate when hardware permits [21].

6.5 Probabilistic Presence: Influence on Detection Stability

The probabilistic presence tracker improves usability by stabilising intermittent detections ("memory") and reducing flicker, which is important for operator trust in CCTV overlays.

However, the evaluation indicates that probabilistic presence can reduce mean IoU because it smooths boxes and may maintain stale localisation when detections are missed. This is a classic trade-off: improved temporal stability vs. instantaneous localisation accuracy.

7 Extension Work: YOLO Evidence and Constraints

7.1 YOLO Evidence, Constraints, and Next-Step Evaluation

The YOLO backend is implemented as an extension path intended for a stronger "final system" experience (run detection with minimal tuning). In this submission, YOLO is treated as a demonstrator rather than a fully benchmarked backend:

- **Evidence of integration:** the project includes the Ultralytics YOLO weights file (`yolov8n.pt`) and a runnable YOLO script (`tools/run_yolo.py`), and the GUI exposes YOLO as a selectable detector.
- **Compute constraint:** to preserve responsiveness on CPU-class hardware, the design rate-limits YOLO inference (detector interval) and keeps the same motion-gated incident logic as the baseline pipeline.
- **Scope constraint (classes):** the demo focuses on *person*, *chair*, and *tv* to match the assessment scenario footage and keep UI overlays interpretable.

- **Next-step evaluation:** a full YOLO evaluation should be added to the same metric pipeline used for HOG/DNN.

7.2 Wisenet Dataset Justification

The decision to use the WiseNET dataset is intentional:

- **Testing Evaluation:** We use WiseNET specifically for *testing and evaluating* the final system, not for training.
- **Relevance:** The corridor scenarios closely match the Northampton Warehouse requirements.
- **Benchmarking:** It allows us to compare our "local" HOG/MobileNet implementation against the dataset's provided "Ground Truth" (defined as human manual annotations) and other baseline models.

8 Conclusion

The project successfully delivers a smart security camera pipeline with a tunable GUI, motion gating, evidence recording, and multiple person detection backends. The results demonstrate that deep detection improves localisation substantially over HOG, but overall recall is limited by conservative motion gating.

For a deployment path, the evidence supports selecting a YOLO-class detector (or a custom-trained model) and integrating it into the same pipeline architecture. The current baseline remains valuable as an explainable reference implementation and as a demonstration of fundamental media technology principles in CCTV analytics.

8.1 Recommendations

Based on the evaluation:

1. **Adopt YOLO for Production:** The recall gap is too significant to ignore. The optional YOLO backend should be made primary.
2. **Integrate Remote Alerts:** Shift from purely GUI-based alerts to a web-service model that can push email/SMS notifications.
3. **Target Person Class Accuracy:** Future tuning should focus specifically on the "Person" class accuracy (as we did here) rather than generic motion, as this filters out most environmental noise.

References

[1] Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). *Simple online and realtime tracking.* 2016 IEEE International Conference on Image Processing (ICIP), 3464–3468.

[2] Bewley, A. (n.d.). *SORT: Simple Online and Realtime Tracking.* GitHub repository. `https://github.com/abewley/sort`

[3] Bradski, G. (2000). *The OpenCV Library.* Dr. Dobb's Journal of Software Tools. `https://bibbase.org/network/publication/bradski-theopencvlibrary-2000`

[4] BriefCam. (2020). *The Evolution of Video Surveillance.* Retrieved from industry whitepaper.

[5] Brown, R. G. (1959). *Statistical Forecasting for Inventory Control.* McGraw-Hill, New York.

[6] Burgard, W. (n.d.). *Teheran tutorial.* Retrieved from `http://www2.informatik.uni-freiburg.de/~burgard/transfer/teheran_tutorial.pptx.pdf`

[7] Canny, J. (1986). *A computational approach to edge detection.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), 679–698.

[8] chuanqi305. (n.d.). *MobileNet-SSD (Caffe) model files.* `https://github.com/chuanqi305/MobileNet-SSD`

[9] Dalal, N., & Triggs, B. (2005). *Histograms of Oriented Gradients for Human Detection.* IEEE Conference on Computer Vision and Pattern Recognition. `https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf`

[10] Howard, A. G., et al. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.* arXiv:1704.04861.

[11] LearnOpenCV. (n.d.). *Non-Maximum Suppression (NMS) Theory and Implementation.* `https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/`

[12] Liu, W., et al. (2016). *SSD: Single Shot MultiBox Detector.* European Conference on Computer Vision.

[13] OpenCV. (n.d.). *Background Subtraction (MOG2).* `https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html`

[14] OpenCV. (n.d.). *Morphological Operations.* `https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html`

[15] OpenCV. (n.d.). *Deep Neural Network (dnn) module.* `https://docs.opencv.org/4.x/d6/d0f/group__dnn.html`

[16] OpenCV. (n.d.). *Contours: Getting Started.* `https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html`

[17] OpenCV. (n.d.). *HOGDescriptor.* https://docs.opencv.org/4.x/d5/d33/structcv_1_1HOGDescriptor.html

[18] OpenCV. (n.d.). *Reading and Writing Videos.* https://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html

[19] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[20] Redmon, J., & Farhadi, A. (2016). *YOLO9000: Better, Faster, Stronger.* arXiv:1612.08242.

[21] Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement.* arXiv:1804.02767.

[22] ros-planning. (n.d.). *costmap_2d (ROS navigation stack).* GitHub repository. https://github.com/ros-planning/navigation/tree/noetic-devel/costmap_2d

[23] Simple online and realtime tracking. (2016). *Monash University Research Repository record.* https://research.monash.edu/en/publications/simple-online-and-realtime-tracking/

[24] Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics.* MIT Press. (Chapter 4.2: Binary Bayes Filters). https://mitpress.ublish.com/book/probabilistic-robotics

[25] Ultralytics. (n.d.). *Predict Mode.* https://docs.ultralytics.com/modes/predict/

[26] Valeman. (n.d.). *Robert G. Brown's exponential smoothing: origins, development, and legacy.* Medium. https://valeman.medium.com/robert-g-browns-exponential-smoothing-origins-development-and-legacy-e23dd8cca159

[27] Valera, M., & Velastin, S. A. (2005). *Intelligent distributed surveillance systems: a review.* IEE Proceedings-Vision, Image and Signal Processing.

[28] Velastin, S. A., & Boghossian, B. A. (2006). *Visual surveillance of public transport.* IEE Proceedings-Vision, Image and Signal Processing.

[29] Zivkovic, Z. (2004). *Improved adaptive Gaussian mixture model for background subtraction.* International Conference on Pattern Recognition.