electronics

MDPI

Article

Alert360: A Low-Cost, Interpretable Browser-Based Motion Awareness for Multi-Camera Monitoring

Minh Nguyen *, Trung Nguyen, Xinyu Hu and Bingjie Xiao

School of Engineering, Computer & Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand; trung.nguyen@aut.ac.nz (T.N.); xinyu.hu@aut.ac.nz (X.H.); bingjie.xiao@aut.ac.nz (B.X.)

* Correspondence: minh.nguyen@aut.ac.nz

Abstract

This paper presents Alert360, a low-cost and intrinsically interpretable browser-based multi-camera motion awareness system for local monitoring in security, retail, education, laboratory, and workplace settings, designed to run on commodity cameras and consumer-grade browser-capable computers without dedicated GPUs. Existing video analytics platforms are often cloud-dependent, costly, or designed for complex surveillance tasks that exceed the needs of small sites. Alert360 addresses a narrower but practically important problem: lightweight operational motion awareness using commodity cameras and modern web browsers. Alert360 provides lightweight, rule-based, and privacy-conscious motion awareness using commodity cameras and modern web browsers. The system performs camera capture, frame processing, motion estimation, alerting, and event review entirely on the client side. Its motion pipeline includes frame down-scaling, grayscale conversion, smoothing, adaptive background differencing, temporal differencing, mask refinement, and blob-based activity estimation. The interface supports concurrent camera streams, configurable frame rates, adjustable sensitivity, movement focus, off-screen guidance, optional acoustic alerts, and local snapshot retention. A controlled benchmark across seven multi-camera scenarios showed a mean processing time of 10.62 ms per frame, zero late frames, and a maximum estimated processing duty cycle of 24.43%. An illustrative office-area deployment using a 1080p 360-degree webcam showed that all thirty scripted human-movement scenarios activated the expected motion-triggered recording workflow. These results demonstrate that motion awareness can be delivered through browser-native local processing on consumer-grade computers under the evaluated multi-camera configurations.

Keywords: motion detection; video surveillance; edge analytics; browser vision; retail analytics; low-cost monitoring; background subtraction; multi-camera systems

Check for updates

Academic Editor: Danilo Pau

Received: 16 May 2026
Revised: 3 October 2026
Accepted: 7 October 2026
Published: 12 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Video surveillance and scene monitoring are now used far beyond traditional security settings. Small retailers, convenience stores, cafés, pharmacies, office entrances, reception counters, school spaces, laboratories, and public-facing service points increasingly require lightweight visual awareness for loss prevention, staff safety, occupancy estimation, customer movement observation, and incident review. However, many current surveillance analytics solutions remain expensive, cloud-centric, difficult to configure, or overly dependent on specialised hardware and advanced deep learning pipelines.

This paper introduces Alert360, a practical multi-camera motion awareness system designed to bridge the gap between raw webcam feeds and high-cost intelligent surveillance platforms. The system is intentionally engineered around five principles:

1. Low deployment cost: use of commodity cameras and commodity browsers.
2. Local-first processing: visual analysis occurs on the client side.
3. Operator-oriented usability: motion causes automatic focus, alert visualisation, and optional sound.
4. Scalable lightweight monitoring: support for many simultaneous camera panels with per-camera processing rate control.
5. Privacy-aware event handling: snapshots remain in local browser storage rather than being uploaded by default.

The design target is not to compete directly with high-end cloud video analytics or large-scale deep surveillance infrastructure. Instead, Alert360 focuses on a practically important deployment niche: environments where operators need a simple, explainable, affordable, and rapidly deployable motion awareness layer. In such settings, the most useful question is often not "which object class is present?" but rather "where is activity happening right now, and how can the operator see it immediately?"

Accordingly, the novelty claimed in this paper is not a new background-subtraction algorithm. The underlying motion pipeline deliberately uses established, interpretable operations. The contribution lies in the integration of these operations into a browser-native, local-first, multi-camera monitoring system with operator-centred attention management, configurable per-camera processing, privacy-aware local event handling, and a low-friction deployment model for small sites.

The research problem addressed in this paper is therefore not high-level semantic video understanding, but low-cost operational motion awareness: how can multiple local camera streams be monitored in a privacy-conscious, explainable, and deployable way using ordinary browser technology? This problem is important because many small-site deployments need fast attention redirection and lightweight event review rather than full object classification, identity recognition, or cloud-scale behavioural analytics.

This paper makes the following contributions:

- It proposes a browser-native multi-camera motion awareness architecture that integrates local media capture, canvas-based frame processing, per-camera configuration, and client-side event handling.
- It implements an interpretable, lightweight motion analysis pipeline using frame down-scaling, grayscale processing, local smoothing, adaptive background differencing, temporal differencing, mask refinement, and largest-blob activity estimation. This pipeline is presented as a deliberately simple and explainable implementation choice rather than as an algorithmic novelty claim.
- It introduces an operator-centred attention management interface that supports automatic movement focus, manual focus override, off-screen active-camera guidance, visual alerting, optional sound notification, and local snapshot review.
- It demonstrates a privacy-conscious deployment model in which baseline motion processing and event review occur locally in the browser without continuous cloud upload.
- It reports a controlled browser-side computational evaluation across seven multi-camera configurations, characterising achieved processing rate, per-frame processing time, late-frame behaviour, and estimated processing duty cycle under repeatable generated-frame workloads. The evaluation is positioned as computational feasibility testing and is not intended to replace labelled accuracy evaluation or field validation.

In this paper, "low-cost" refers specifically to the hardware and software requirements of the proposed deployment architecture rather than to a universal total-cost-of-ownership comparison. Alert360 is designed to operate using commodity, off-the-shelf hardware, including consumer-grade cameras and a general-purpose browser-capable computer. It does not require a dedicated GPU, specialised edge-AI appliance, video-processing server, native client installation, or paid cloud-analytics services. The architecture avoids the need for hardware, software, installation, and recurring service costs commonly associated with more complex video analytics deployments. "Interpretable" refers to the explicit and inspectable nature of the rule-based pipeline: an alert can be traced through defined image differences, thresholds, motion masks, blob extraction, motion percentage, and persistence rules.

2. Literature Review

Motion detection and background subtraction have long been core components in video surveillance. Early and influential work by Stauffer and Grimson modelled each pixel as an adaptive mixture of Gaussians, enabling real-time background modelling under slowly changing scene conditions [1]. Subsequent work, such as ViBe, emphasised efficient sample-based background models with low computational cost and strong empirical performance [2]. Comprehensive reviews by Sobral and Vacavant, and by Goyal and Singhai, show that background subtraction remains fundamental because it offers strong efficiency, interpretability, and suitability for stationary-camera settings [3,4].

Benchmark-oriented analysis also demonstrated that video surveillance performance must be judged not only by accuracy but by robustness to shadows, dynamic backgrounds, illumination changes, and computational cost [5]. The CDnet benchmark further illustrates the need to evaluate foreground-detection methods under camera jitter, intermittent object motion, bad weather, dynamic backgrounds, and shadow conditions [6]. This is highly relevant to Alert360, which is designed for practical environments rather than tightly controlled lab scenes. In small-store or local security scenarios, it is often preferable to use transparent and tunable motion pipelines that can run on commodity hardware rather than computationally expensive pipelines that require dedicated accelerators.

At the same time, edge and on-premise analytics have become increasingly attractive. Edge-based video analytics reduce latency, reduce bandwidth use, and can better preserve privacy than cloud-first processing [7,8]. Privacy considerations are especially important in local monitoring settings such as shops, service counters, and workplaces, where indiscriminate cloud upload may be undesirable or non-compliant with local policy. Prior work on privacy-preserving video analytics highlights that even seemingly simple visual pipelines can reveal sensitive information, motivating architectures that keep processing close to the point of capture [9]. From a regulatory and governance perspective, local processing may reduce exposure to third-party data transfer, enhancing compatibility with privacy regulations, such as the GDPR and the California Consumer Privacy Act. [10,11].

Recent retail analytics systems have focused heavily on deep learning, skeleton analysis, behaviour recognition, and customer-state modelling [12,13]. These approaches are valuable for rich behavioural insight, but they are often more complex than what many small sites need. Alert360 addresses a simpler but operationally important problem: rapid identification of active scenes, immediate operator attention redirection, and low-friction review of recent motion events. This makes the system especially suitable as an entry-level or complementary layer for local retail and security analytics.

Finally, browser-based camera capture is now mature enough to support real deployment through web media interfaces such as `getUserMedia`, enabling direct access to local camera streams from modern browsers [14]. This creates an opportunity to build low-cost,

installation-light monitoring systems that are easy to update and accessible from standard computing environments without dedicated native software.

Browser-side visual processing has also become more feasible because modern browsers expose efficient media, graphics, and execution interfaces. JavaScript libraries and toolkits such as TensorFlow.js (version 4.22.0), MediaPipe (version 1.0.0), and OpenCV.js (version 5.0.0) show that visual inference or image processing can be moved into the browser for selected workloads [15–17]. Lower-level technologies such as WebAssembly, WebGL, and WebGPU provide further opportunities for accelerating numerical and graphics-oriented computation in web applications [18–20]. Alert360 differs from these general-purpose browser-vision toolkits because it is not primarily a library for model execution or image processing. It is a complete operator-facing monitoring workflow that combines browser capture, lightweight motion analysis, attention selection, alerting, and local event review.

Open-source monitoring systems such as ZoneMinder (version 1.38.4) and Shinobi (https://shinobi.video/, accessed on 1 October 2026) are also relevant points of comparison because they provide practical video monitoring and recording functions [21,22]. However, these systems are typically organised around server-side video management, network video recording, or installed monitoring infrastructure. Alert360 is positioned differently: it prioritises client-browser execution, minimal installation burden, and local motion awareness for small sites. This does not imply that Alert360 is a substitute for mature video management systems; rather, it clarifies its narrower contribution as a lightweight browser-native alternative for operational motion awareness.

Table 1 therefore summarises relative positioning rather than absolute superiority. "Cost" refers to expected infrastructure and setup burden; "explainability" is the clarity and understandability of a system's decision-making process; "privacy risk" refers to the relative exposure created by continuous cloud transfer, centralised storage, or third-party processing. These categories are necessarily qualitative, but they help clarify the design trade-offs addressed by Alert360. Alert360 is interpretable and thereby inherently explainable, because its processing logic can be inspected and adjusted by non-specialist operators.

Table 1. Architectural positioning of Alert360 against representative monitoring architectures and related browser/open-source solutions. The table compares typical processing locations, infrastructure requirements, decision bases, data locations and functional roles. It is not an empirical comparison of detection accuracy, computational performance, total cost or privacy protection.

Approach	Primary Processing	Cost/Setup	Decision Basis	Data Location	Typical Role
Cloud video analytics	Centralised cloud backend	n/a	Varies from explicit rules to proprietary learned models	External cloud; requires consideration of privacy risks	Enterprise surveillance, centralised monitoring, large-scale multi-site analytics.
Embedded deep edge camera	Dedicated edge hardware or appliance	Medium–High	Varies; commonly learned detection or classification models	Local and/or cloud	Real-time object detection, counting, and appliance-specific intelligent cameras.
Classical DVR/CCTV without analytics	On-premise recorder	Medium	Recordings with limited event rules; normally no analytics	On-premise	Passive recording and post-event review, usually without automatic attention redirection.

Table 1. *Cont.*

Approach	Primary Processing	Cost/Setup	Decision Basis	Data Location	Typical Role
Open-source VMS/NVR systems (e.g., ZoneMinder, Shinobi)	Server-side or installed video management stack	n/a	Varies: configuration, plugins and any analytical models	Primarily on-premise	Mature video management, recording, camera administration, and web dashboards.
Browser-side CV toolkits (e.g., TensorFlow.js, MediaPipe, OpenCV.js)	Browser model inference or image processing libraries	n/a	Varies from image-processing algorithms to learned models	Application dependent	Building blocks for browser-based vision applications rather than complete monitoring workflows.
Alert360 (proposed)	Client browser/local device	Low dedicated-infrastructure requirement	Interpretable rule-based motion analysis; no semantic analytics	Local by default	Local stores, classrooms, laboratories, offices, kiosks, reception areas, and lightweight security/customer-awareness scenarios where attention redirection is more important than semantic recognition.

3. System Design and Architecture

Alert360 is a browser-based, multi-camera, local-first monitoring system. The high-level architecture is shown conceptually in Figure 1. The system has five interacting layers:

1. Capture layer: camera streams are acquired using browser media capture interfaces.
2. Per-camera processing layer: each stream is processed independently at a user-selected frame rate.
3. Attention management layer: motion strength across cameras is compared to determine focus.
4. Interaction layer: the interface highlights the active scene, optionally scrolls to it, and may emit an alert sound.
5. Review layer: if enabled, motion-triggered snapshots are stored locally with time-limited retention.

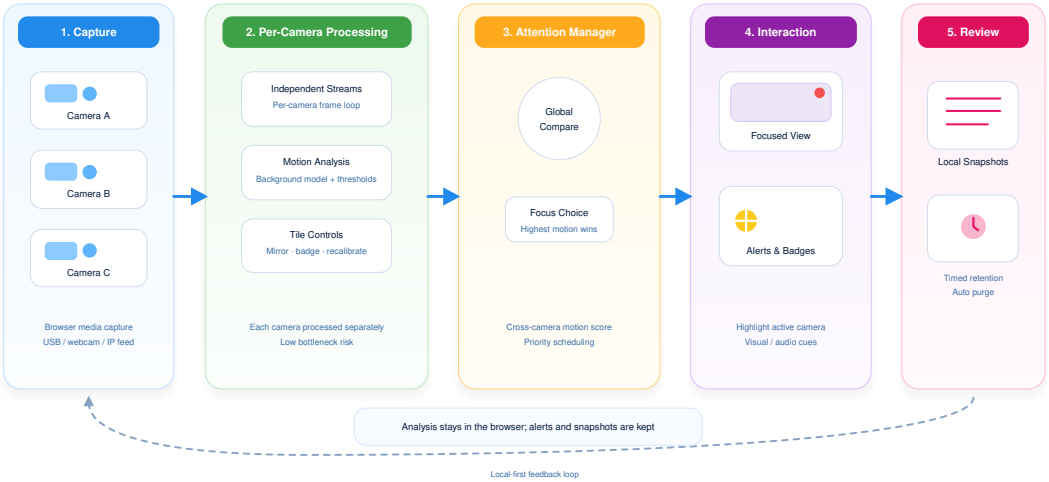


Figure 1. Browser-based privacy-conscious monitoring pipeline. Camera streams are captured in the browser, processed independently per camera, compared by an attention manager, surfaced to the operator through visual or acoustic alerts, and optionally retained as local motion-triggered snapshots for short-term review.

A key architectural choice is that motion analytics remain lightweight and interpretable. Each camera tile is represented as a self-contained processing unit with its own:

- frame-rate setting,
- motion threshold,
- mirror option,
- resettable background model,
- state badge,
- focused/alert visual state.

This modularity makes the system suitable for heterogeneous environments in which camera viewpoints, distances, and activity levels differ significantly.

3.1. Multi-Camera Scheduling and Synchronisation Assumptions

Alert360 treats each camera as an independent processing unit rather than as part of a strictly synchronised multi-view reconstruction system. The purpose is operational attention redirection, not frame-accurate 3D localisation or cross-camera tracking. Per-camera frame-rate settings are therefore used to constrain resource demand and reduce contention across streams. Frames are processed opportunistically at the configured rate for each camera, and the attention manager compares the most recent motion scores available from the active camera tiles. This design avoids the need for global frame synchronisation, but it also means that Alert360 is not intended for applications requiring precise temporal alignment across cameras. Future implementations may use Web Workers and OffscreenCanvas to further reduce main-thread contention and improve scheduling robustness.

3.2. Movement Focus Mode

The interface introduces a Movement Focus Mode to help operators handle many simultaneous cameras. Two behaviours are provided:

- Automatic mode: the system continuously selects the camera with the strongest current motion score.
- Manual mode: the operator explicitly pins one camera and disables automatic focus switching.

This mechanism is valuable when the number of streams exceeds a single screen view. In automatic mode, the system not only highlights the strongest motion tile but also scrolls the page to bring that camera into view when it is off-screen. This improves human response time in long dashboards with 10 or more cameras.

3.3. Low-Cost Deployment Model

The practical attraction of Alert360 is its deployment simplicity:

- no dedicated GPU is required for the baseline lightweight motion pipeline,
- no cloud server is required for baseline use,
- no native application installation is required,
- low-cost USB or built-in cameras can be used,
- the same interface supports local shops, schools, labs, and office spaces.

Such a model lowers the barrier to adoption for small and medium sites that cannot justify enterprise surveillance analytics platforms. GPU-backed browser technologies such as WebGL or WebGPU may still be useful for richer analytics or deep-learning extensions, but they are not required for the motion-awareness workflow evaluated in this paper. This deployment model avoids several common infrastructure costs, although the study does not claim to establish the lowest total cost among all available browser-based or AI-based systems.

4. Rule-Based Motion Trigger

The current Alert360 implementation assumes approximately static cameras and performs motion analysis using a lightweight pipeline.

4.1. Frame Preprocessing

For each incoming frame, the system computes a grayscale representation:

$$I_t(x) = 0.299R_t(x) + 0.587G_t(x) + 0.114B_t(x), \tag{1}$$

where x denotes a pixel and t denotes time. A small local blur is then applied to suppress sensor noise and stabilise differencing.

4.2. Adaptive Background Model

A running background estimate $B_t(x)$ is updated using exponential averaging:

$$B_t(x) = (1 - \alpha)B_{t-1}(x) + \alpha I_t(x), \tag{2}$$

where α is a small learning rate. This allows the system to adapt to gradual illumination change while still preserving sensitivity to transient motion. A smaller α makes the background model more stable but slower to adapt to gradual lighting change; a larger α adapts more quickly but may absorb slow-moving foreground objects into the background.

4.3. Combined Motion Criterion

The foreground mask is derived using both background difference and temporal difference. Let

$$D_b(x) = |I_t(x) - B_t(x)|, \tag{3}$$

$$D_m(x) = |I_t(x) - I_{t-1}(x)|. \tag{4}$$

A pixel is considered active if

$$M_t(x) = \begin{cases} 1, & \text{if } (D_b(x) > \tau_b \wedge D_m(x) > \tau_m) \ \vee \ D_m(x) > \tau'_m \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where τ_b, τ_m, and τ'_m are thresholds. This combined decision rule is designed to suppress minor background drift while preserving responsiveness to genuine movement. The background threshold τ_b controls sensitivity to deviation from the learned scene, τ_m controls sensitivity to frame-to-frame change, and τ'_m acts as a high temporal-difference trigger for sudden motion. Lower thresholds increase sensitivity but may increase false positives from noise, shadows, screen flicker, or reflections; higher thresholds reduce false positives but may miss subtle or distant motion.

The default implementation uses $\alpha = 0.012$, $\tau_b = 18$, $\tau_m = 11$, and $\tau'_m = \tau_m + 6 = 17$. These values provide the baseline configuration used for the reported experiments.

4.4. Mask Cleanup and Blob Extraction

The raw mask is refined using a local neighbourhood rule that suppresses isolated pixels and encourages spatially coherent motion regions. Connected-component analysis is then used to extract the largest blob. The motion strength for a frame is computed as:

$$P_t = 100 \times \frac{\sum_x M_t(x)}{HW}, \tag{6}$$

where H and W are the frame height and width.

In the default implementation, frames are analysed at a processing scale of 0.45, so the motion mask and connected components are computed on a down-scaled analysis canvas rather than on the full-resolution camera frame. The minimum blob size is set to 55 pixels to suppress isolated noise, and the default trigger threshold is 0.25, corresponding to 0.25% of the processed frame area.

The largest active blob is then drawn on the displayed canvas, and the corresponding percentage P_t is used for both alerting and camera focus selection. Using the largest connected component is intentionally simple and operator-interpretable, but it can underrepresent multiple dispersed movements, small distant targets, or simultaneous low-intensity activities in different parts of the frame. Future scoring can combine multiple components, zone-level weights, and temporal consistency to better represent distributed motion.

4.5. Per-Camera Persistence and Alerting

To reduce flicker, the system maintains a short persistence state per camera. A camera enters the alert state when motion remains above threshold long enough to satisfy the persistence rule. Once active, the system can:

- display a motion banner,
- highlight the camera tile,
- trigger acoustic notification,
- move focus to the corresponding panel,
- save a snapshot locally if storage is enabled.

5. Implementation Features

The system is implemented as a monitoring application rather than a general-purpose computer-vision library. Its browser-native implementation combines media capture, frame processing, interface state, alert behaviour, and local event review in one workflow, as shown in Figure 2. This distinguishes it from browser-side vision frameworks, which provide processing primitives or model execution capabilities but do not by themselves define a multi-camera monitoring interface. Alert360 is implemented using JavaScript for processing logic, HTML and CSS for the interface, canvas-based rendering for motion visualisation, and local browser storage for optional event retention. This implementation approach avoids the need for dedicated client software and supports lightweight deployment on ordinary desktop or laptop systems. The public prototype is available at https://ecms.nz/alert360 (accessed on 1 October 2026).



Figure 2. End-to-end processing pipeline of Alert360. Each browser-acquired camera stream is converted to grayscale, smoothed, compared against background and temporal references, cleaned into coherent motion masks, summarised through blob-based motion estimation, and then used to select focus, trigger alerts, and optionally save motion snapshots locally.

The current system is designed around practical operator needs in real-world monitoring environments, incorporating the features listed in Table 2. It supports simultaneous

monitoring of multiple camera streams, configurable per-camera processing rates, adjustable motion sensitivity, automatic or manual focus selection, off-screen panel guidance, optional motion alert sounds, and privacy-conscious local snapshot review. Together, these features make the platform suitable for low-cost local monitoring where usability, configurability, and deployment simplicity are more important than heavy analytical complexity.

Table 2. Core implementation features of Alert360.

Feature	Description
Multi-camera monitoring	Supports simultaneous viewing and processing of multiple local camera streams within a unified browser-based dashboard.
Configurable per-camera FPS	Allows each camera to operate at an adjustable processing rate from low-frequency lightweight monitoring to more responsive real-time analysis.
Adjustable motion sensitivity	Enables per-camera threshold tuning so that the system can be adapted to different scene conditions and operational requirements.
Movement focus mode	Supports either automatic focus on the most active camera or manual operator-controlled focus selection.
Off-screen focus guidance	Automatically scrolls the interface to bring an active camera panel into view when motion occurs outside the current visible screen area.
Alert sound configuration	Provides optional audible notifications for motion events, with multiple selectable sound profiles and a silent default setting.
Local snapshot review	Stores motion-triggered image snapshots locally for later inspection through an integrated review interface.
Retention-aware local storage	Automatically removes stored snapshots after seven days to reduce storage growth and maintain a privacy-conscious local-only design.

6. System Implementation and Demonstration

Alert360 is presented as a deployable systems contribution centred on a working browser-based implementation. The prototype demonstrates the core functions required for practical local monitoring: concurrent multi-camera capture, browser-side motion detection, configurable processing rates, event-focused attention management, motion-triggered snapshot review, and local-first storage. The emphasis is on deployability, usability, and privacy-conscious operation in ordinary environments such as small stores, offices, laboratories, classrooms, reception areas, and temporary monitoring sites.

A key design decision is that the system avoids a fixed processing profile. Instead, each camera can be configured independently according to scene complexity, available hardware, and monitoring requirements. Per-camera frame-rate control allows users to reduce computational demand when high temporal resolution is unnecessary. For example, low-frequency processing at 1–5 frames per second may be sufficient for many situational awareness tasks, while higher rates can be selected for more responsive monitoring. This makes the system adaptable to heterogeneous camera layouts and modest client hardware.

Storage is also handled in a lightweight manner. Rather than recording continuous video, Alert360 stores snapshots only when motion is detected and only when local storage is explicitly enabled. This event-based approach reduces storage demand compared with continuous surveillance recording while preserving useful visual evidence for short-term review. In the baseline configuration, motion-triggered snapshots remain in local browser

storage and are not uploaded to a cloud service. This reduces bandwidth requirements, lowers deployment complexity, and keeps monitoring data under local control.

The main monitoring interface and snapshot review interface are shown in Figures 3 and 4. These interfaces illustrate the practical orientation of the system: operators can view multiple camera panels, adjust processing parameters, identify active scenes, and review recent motion-triggered images without installing native software or maintaining a dedicated video analytics server.

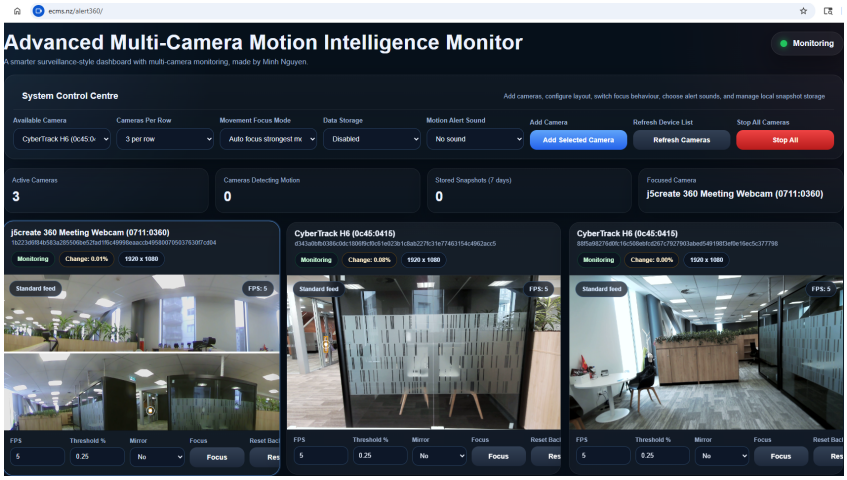


Figure 3. Main Alert360 monitoring interface showing a three-camera configuration with two standard webcams and one 360-degree camera, together with the control panel, motion-aware dashboard layout, and browser-based multi-camera monitoring view.

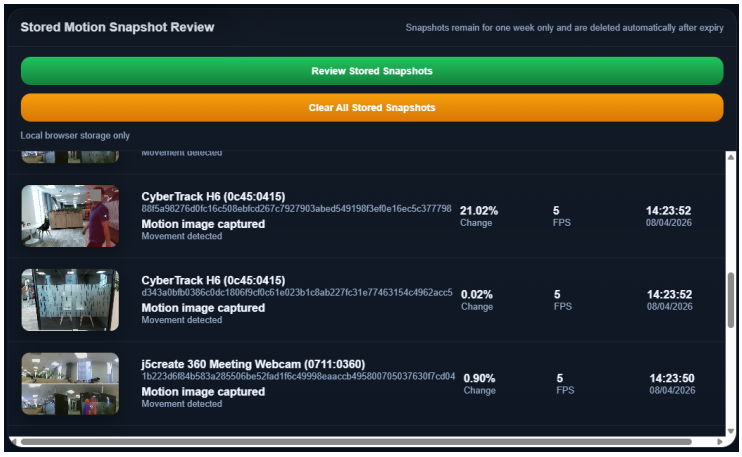


Figure 4. Snapshot review interface of Alert360 showing motion-triggered image thumbnails with associated timestamps and camera identifiers, enabling lightweight local event inspection and privacy-conscious short-term review.

Illustrative Office Deployment

Alert360 was also deployed in a small office-area environment using three actual cameras. The space contained large windows with light-reduction coverings and some smaller uncovered windows, creating mixed artificial and natural lighting. The interface in Figure 3 illustrates the deployment environment.

A limited functional demonstration was conducted during lunchtime with relatively strong external sunlight using the j5create 360 Degree All Around Meeting Webcam (model JVCU360; Kaijet Technology International Corporation, New Taipei City, Taiwan). Thirty scripted human-movement scenarios were performed within an approximately 15 m monitoring distance. In all thirty scenarios, the movement activated the expected motion-

triggered attention and local recording/snapshot workflow. This exercise provides an end-to-end functional check under the stated office conditions. Because it did not include a defined period of continuously labelled negative observation, it does not provide a false-positive rate, false-alarm frequency, precision, recall, or F1-score.

Taken together, these design choices position Alert360 as a lightweight and adaptable monitoring platform for ordinary local computing environments. To complement this deployment-oriented discussion, the following section reports a controlled browser-side synthetic performance evaluation of the motion-analysis pipeline under multiple camera-resolution and throughput scenarios.

7. Computational Characterisation of the Browser Processing Loop

To assess whether the proposed browser-native design can support practical multi-camera workloads, we evaluated Alert360 using a controlled browser-side processing benchmark that reproduces the major processing stages used in the live monitoring page. The workload includes frame down-scaling, grayscale conversion, 3×3 local blur, adaptive background differencing, temporal differencing, mask cleaning, and largest-blob extraction. In this context, "synthetic" means that the client-side processing loop was exercised using repeatable generated frame data and configured camera-throughput scenarios, rather than using labelled surveillance videos or a live field deployment. Repeatable generated frame data were used to provide consistent input across configurations and isolate the computational behaviour of the browser-side processing pipeline. This choice prevents uncontrolled variations in camera placement, lighting variation, background activity, and human movement patterns.

The client-side evaluation was run on the laptop-class Windows environment summarised in Table 3. The application was run on a relatively standard personal computer.

Table 3. Client-computer specifications used for the browser-side computational evaluation.

Item	Specification
Client computer	HP EliteBook 860 16 inch G10 Notebook PC
Processor	13th Gen Intel® Core™ i5-1335U, 1.30 GHz base frequency
Installed memory	32.0 GB RAM, 31.6 GB usable, 5200 MT/s memory speed
Graphics	Intel® UHD Graphics, reported graphics memory 128 MB
Operating system	Windows 11 Enterprise
System type	64-bit operating system

Seven completed scenarios were evaluated, covering one-, two-, and four-camera configurations at 720p and 1080p input resolutions, a higher-throughput 1080p case, and a wide 360-degree-like 4K input case (see Table 4). In all scenarios, frames were down-scaled using a processing scale of 0.45 before analysis. This reflects a central design assumption of Alert360: operational motion awareness does not require full-resolution analysis of every frame. Instead, lower-resolution processing can preserve sufficient motion information while reducing computational cost.

Across the seven scenarios, Alert360 achieved a mean processing time of 10.62 ms per processed frame, with an average p95 processing time of 19.30 ms. The maximum observed per-frame processing time was 42.30 ms in the wide 360-degree-like 4K scenario. Importantly, all evaluated scenarios completed with zero late frames, indicating that the browser-side processing loop was able to keep up with the workload under the tested conditions.

The highest achieved total throughput was observed in the 4 cam 720p 5 fps scenario, which processed approximately 20.00 frames per second across four camera streams. This scenario achieved 19.996 processed frames per second against a target of 20 frames per

second, with an average processing time of 6.31 ms, p95 processing time of 12.90 ms, and no late frames. Under the stability criterion used in this study—at least 95% of target throughput, no more than 5% late frames, and no more than 85% estimated processing duty cycle—the highest-throughput stable configuration among those tested was therefore 4 cam 720p 5 fps, corresponding to four cameras at 1280 × 720 input resolution and 5 frames per second per camera.

Table 4. Controlled browser-side synthetic performance evaluation of Alert360 across completed multi-camera scenarios.

Scenario	Cams	Input Res.	Target FPS	Achieved FPS	Avg. ms	p95 ms	Late Frames	Est. Load
1 cam 720p 5 fps	1	1280 × 720	5	5.00	9.27	18.00	0%	4.63%
2 cam 720p 5 fps	2	1280 × 720	10	10.00	5.09	11.00	0%	5.09%
4 cam 720p 5 fps	4	1280 × 720	20	20.00	6.31	12.90	0%	12.62%
2 cam 1080p 5 fps	2	1920 × 1080	10	10.00	12.69	22.20	0%	12.69%
4 cam 1080p 5 fps	4	1920 × 1080	20	19.99	9.42	15.70	0%	18.85%
2 cam 1080p 10 fps	2	1920 × 1080	20	19.99	12.21	22.70	0%	24.43%
1 360-like 4K 5 fps	1	3840 × 1080	5	5.00	19.35	32.60	0%	9.68%

The maximum estimated processing duty cycle observed across all scenarios was 24.43%, occurring in the 2 cam 1080p 10 fps scenario. This is substantially below the 85% load threshold used in the stability criterion. Note that this value is not equivalent to measured browser or system CPU utilisation.

These results support the central design claim of Alert360: lightweight multi-camera motion awareness can be performed directly in a standard browser without requiring cloud processing, specialised edge appliances, or GPU-accelerated deep learning models. The evaluation also confirms the practical value of configurable per-camera frame rates and frame down-scaling. By processing lower-resolution analysis frames rather than full-resolution video frames, Alert360 preserves responsiveness while reducing computational cost. The synthetic component of the evaluation measures throughput, latency, late-frame behaviour, estimated processing duty cycle, and indicative memory behaviour. The office-area pilot adds a small real-world check of motion-triggered event detection and recording, in which 30 designed human-movement scenarios activated the expected workflow. However, the study still does not quantify precision, recall, F1 score, false alarm rate, or missed-event rate against a labelled benchmark because no fully annotated ground-truth video dataset with positive and negative cases was used in the present study.

The results characterise controlled client-side execution under the tested configurations rather than final deployment or comparative performance. Real-world performance will depend on the client environment, camera behaviour, lighting variation, scene complexity, browser scheduling, and the number of simultaneously active streams. Future empirical work should also include labelled video data, standard foreground-detection benchmarks such as CDnet, and comparisons with established baselines such as MOG2, KNN, ViBe, and browser-side OpenCV.js implementations.

8. Discussion

Alert360 occupies a middle ground between passive CCTV viewing and computationally intensive semantic video analytics. This positioning is important because many real monitoring contexts do not require object recognition, identity tracking, or behaviour classification. In small stores, reception areas, classrooms, laboratories, and temporary installations, the immediate operational need is often simpler: identify where activity is occurring, direct the operator's attention to the relevant camera, and preserve lightweight evidence for short-term review.

The results support this design direction. The browser-side evaluation showed that the proposed pipeline can sustain multiple camera workloads with low processing latency and no late frames under the tested scenarios. In particular, the four-camera 720p configuration at 5 FPS per camera achieved approximately 20 processed frames per second in total while remaining well below the selected processing-load threshold. The limited office-area pilot further showed that all thirty designed human-movement scenarios within a 15 m monitoring distance activated the expected workflow under moderate lunchtime lighting variation. This suggests that browser-native local processing is computationally feasible for the tested lightweight workload. However, the field validation remains preliminary and should not be read as proof that the detector is robust across all real scenes, because labelled benchmark accuracy and false-alarm behaviour were not evaluated.

The use of a rule-based motion detection pipeline is also deliberate. Although deep learning has significantly advanced video understanding [23], not all monitoring tasks require deep semantic interpretation. Classical differencing and blob-based methods remain attractive when explainability, low cost, tunability, and deployment simplicity are more important than high-level scene understanding. Alert360 demonstrates how these methods can be combined with modern browser capabilities to produce a practical and accessible monitoring system. Compared with browser-side deep-learning frameworks, this approach avoids model download, accelerator dependence, and opaque semantic decisions. Compared with established OpenCV-style foreground-subtraction baselines, the current paper contributes a complete browser-native workflow rather than a benchmark-optimised detector. A controlled empirical comparison with MOG2, KNN, ViBe, and OpenCV.js variants remains necessary.

The local-first architecture provides additional advantages. By avoiding continuous cloud upload in the baseline configuration, Alert360 reduces bandwidth demand and limits unnecessary external exposure of visual data. This is particularly relevant in workplaces, educational spaces, and small businesses where privacy expectations and data governance obligations must be considered. However, local processing should not be interpreted as eliminating all privacy concerns. Real deployments still require appropriate signage, access control, retention policies, user consent where required, and compliance with relevant institutional or legal requirements. Local browser storage also creates its own governance requirements. Stored snapshots may be accessible to users of the same device profile, may persist until deletion or retention expiry, and may be exposed if the client device is compromised. Therefore, privacy-conscious operation requires technical controls and organisational procedures, including restricted device access, clear retention settings, local deletion policies, and careful consideration of whether snapshots should be enabled in sensitive areas.

The present system also has limitations. It assumes approximately static cameras and is therefore less suitable for moving-camera scenarios. It can be affected by global illumination changes, shadows, reflections, and repetitive background motion. Motion detection alone does not determine whether an event is meaningful, suspicious, or benign. The current snapshot review mechanism is designed for short-term local inspection rather than long-term evidential archiving. Finally, the reported evaluation is a repeatable browser-side simulation and should be complemented by future field testing across different browsers, hardware platforms, camera types, and real deployment environments. Important failure cases include sudden global lighting changes, screen flicker, reflective surfaces, moving shadows, camera vibration, crowded scenes, and small or dispersed motion that does not form a dominant blob. These are known challenges for lightweight differencing pipelines and should be evaluated systematically before the system is used in higher-stakes monitoring contexts.

The interpretation of a false alarm also depends on the intended target. Alert360 responds to observable scene motion or visual change; it does not classify whether the source is human or whether the activity is operationally important. A moving curtain therefore represents genuine physical motion at the visual-processing level, although it may be a nuisance alert for a human-monitoring application. Lighting changes, reflections, shadows, and screen flicker may similarly cause unwanted activations. The present pilot did not quantify the frequency of such activations, and no claim regarding false-alarm performance is made.

These limitations define a clear research and development pathway. Future work will investigate anonymous people counting, zone-based motion rules, occupancy estimation, dwell-time analysis, heatmap generation, improved local storage through IndexedDB, optional privacy-preserving remote alerts, and acceleration using emerging browser technologies such as WebGPU. Further work will also evaluate labelled detection accuracy, false-positive and false-negative rates, robustness under real illumination and background variation, and comparative performance against classical background-subtraction baselines and browser-side CV implementations. These extensions could increase the analytical capability of Alert360 while preserving its core principles of affordability, explainability, local control, and low-friction deployment.

9. Conclusions

This paper presented Alert360, a low-cost browser-based multi-camera motion awareness system for local monitoring. The system addresses a practical gap between passive CCTV systems and high-cost cloud or hardware-based video analytics platforms. Rather than pursuing complex semantic surveillance, Alert360 focuses on lightweight operational awareness: detecting motion, identifying the most active camera, guiding operator attention, and supporting short-term local event review.

The proposed system combines browser-native camera capture, canvas-based frame processing, configurable per-camera analysis, motion-based focus selection, optional alerting, and local snapshot retention. Its motion pipeline uses down-scaled grayscale processing, local smoothing, adaptive background differencing, temporal differencing, mask refinement, and blob-based activity estimation. A browser-side computational evaluation across seven scenarios showed low processing time, zero late frames, and substantial computational headroom under the tested conditions. An office-area pilot further tested thirty real human-movement scenarios within a 15 m monitoring distance during lunchtime lighting conditions, and all thirty scenarios activated the expected workflow. This evaluation supports computational feasibility and limited real-world applicability, while comprehensive analyses and comparative baseline testing remain future work.

The significance of Alert360 lies in demonstrating that multi-camera motion awareness and operator-attention management can be implemented through ordinary browser technology without requiring continuous cloud analytics, a dedicated processing server, a specialised edge appliance, or dedicated GPU-based model inference. The contribution is therefore best understood as a browser-native systems integration and operator-workflow contribution rather than a new rule-based motion trigger used within the system. This makes the system particularly relevant for small stores, offices, laboratories, classrooms, kiosks, reception areas, and other local environments where affordability, explainability, privacy awareness, and ease of deployment are central requirements.

Future work will extend Alert360 toward richer local analytics, broader field evaluation, improved storage management, IP camera integration, and optional privacy-preserving alert integration while maintaining the system's lightweight and local-first

design philosophy. The prototype is publicly available at https://ecms.nz/alert360 (accessed on 1 October 2026).

Author Contributions: Conceptualisation, M.N.; methodology, M.N., T.N. and X.H.; software, M.N.; validation, M.N., T.N. and X.H.; formal analysis, M.N.; investigation, M.N. and T.N.; resources, M.N.; data curation, M.N. and T.N.; writing—original draft preparation, M.N.; writing—review and editing, M.N., T.N., X.H. and B.X.; visualisation, M.N. and B.X.; supervision, M.N.; project administration, M.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The summary evaluation results and scenario configurations are included in this article. The Alert360 prototype and its client-side HTML, CSS, and JavaScript source code are publicly accessible at https://www.ecms.nz/alert360/. The source code can be inspected using the browser's View Page Source function. Further inquiries regarding the evaluation materials may be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Stauffer, C.; Grimson, W.E.L. Adaptive Background Mixture Models for Real-Time Tracking. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Fort Collins, CO, USA, 23–25 June 1999; pp. 246–252.
2. Barnich, O.; Droogenbroeck, M.V. ViBe: A Universal Background Subtraction Algorithm for Video Sequences. *IEEE Trans. Image Process.* **2011**, *20*, 1709–1724. [CrossRef] [PubMed]
3. Sobral, A.; Vacavant, A. A Comprehensive Review of Background Subtraction Algorithms Evaluated with Synthetic and Real Videos. *Comput. Vis. Image Underst.* **2014**, *122*, 4–21. [CrossRef]
4. Goyal, K.; Singhai, J. Review of Background Subtraction Methods Using Gaussian Mixture Model for Video Surveillance Systems. *Artif. Intell. Rev.* **2018**, *50*, 241–259. [CrossRef]
5. Brutzer, S.; Höferlin, B.; Heidemann, G. Evaluation of Background Subtraction Techniques for Video Surveillance. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, CO, USA, 20–25 June 2011; pp. 1937–1944. [CrossRef]
6. Wang, Y.; Jodoin, P.M.; Porikli, F.; Konrad, J.; Benezeth, Y.; Ishwar, P. CDnet 2014: An Expanded Change Detection Benchmark Dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Columbus, OH, USA, 23–28 June 2014.
7. Hu, M.; Luo, Z.; Pasdar, A.; Lee, Y.C.; Zhou, Y.; Wu, D. Edge-Based Video Analytics: A Survey. *arXiv* **2023**, arXiv:2303.14329.
8. Cob-Parro, A.C.; Losada-Gutiérrez, C.; Marrón-Romera, M.; Gardel-Vicente, A.; Bravo-Muñoz, I. Smart Video Surveillance System Based on Edge Computing. *Sensors* **2021**, *21*, 2958. [CrossRef] [PubMed]
9. Poddar, R.; Ananthanarayanan, G.; Setty, S.; Volos, S.; Popa, R.A. Visor: Privacy-Preserving Video Analytics as a Cloud Service. *arXiv* **2020**, arXiv:2005.08523.
10. European Parliament and Council of the European Union. *Regulation (EU) 2016/679 (General Data Protection Regulation)*; European Parliament and the Council of the European Union: Luxembourg, 2016.
11. State of California. *California Consumer Privacy Act of 2018*; California Civil Code, Division 3, Part 4, Title 1.81.5; State of California: Sacramento, CA, USA, 2018.
12. Nguyen, T.D.; Hihara, K.; Hoang, T.C.; Utada, Y.; Torii, A.; Izumi, N.; Thuy, N.T.; Pham, D.T.; Tran, L.Q. Retail Store Customer Behavior Analysis System: Design and Implementation. *arXiv* **2023**, arXiv:2309.03232.
13. Shili, M.; Jayasingh, S.; Hammedi, S. Advanced Customer Behavior Tracking and Heatmap Analysis for Retail Environments. *Electronics* **2024**, *13*, 4730. [CrossRef]
14. World Wide Web Consortium. Media Capture and Streams. 2025. Available online: https://www.w3.org/TR/mediacapture-streams/ (accessed on 8 April 2026).
15. Smilkov, D.; Thorat, N.; Assogba, Y.; Nicholson, C.; Kreeger, N.; Yu, P.; Cai, S.; Nielsen, E.; Soegel, D.; Bileschi, S.; et al. TensorFlow.js: Machine Learning for the Web and Beyond. In *Proceedings of the Machine Learning and Systems*; Talwalkar, A., Smith, V., Zaharia, M., Eds.; MLSys: Indio, CA, USA, 2019; Volume 1, pp. 309–321.
16. Lugaresi, C.; Tang, J.; Nash, H.; McClanahan, C.; Uboweja, E.; Hays, M.; Zhang, F.; Chang, C.L.; Yong, M.; Lee, J.; et al. MediaPipe: A Framework for Perceiving and Processing Reality. In Proceedings of the Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019, Long Beach, CA, USA, 16–20 June 2019.

17. Taheri, S.; Vedienbaum, A.; Nicolau, A.; Hu, N.; Haghighat, M.R. OpenCV.js: Computer vision processing for the open web platform. In Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18), New York, NY, USA, 12–15 June 2018; pp. 478–483. [CrossRef]
18. Haas, A.; Rossberg, A.; Schuff, D.L.; Titzer, B.L.; Holman, M.; Gohman, D.; Wagner, L.; Zakai, A.; Bastien, J. Bringing the web up to speed with WebAssembly. In Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), New York, NY, USA, 18–23 June 2017; pp. 185–200. [CrossRef]
19. Khronos Group. *WebGL Specification*; Technical Specification; Khronos Group: Beaverton, OR, USA, 2025.
20. World Wide Web Consortium. WebGPU. W3C Candidate Recommendation Draft. 2025. Available online: https://www.w3.org/TR/webgpu/ (accessed on 22 June 2026).
21. ZoneMinder Project. ZoneMinder Documentation (Stable Release). 2026. Available online: https://zoneminder.readthedocs.io/en/stable/ (accessed on 22 June 2026).
22. Shinobi Systems. Shinobi: Open Source CCTV and NVR Solution. Project Documentation. 2025. Available online: https://docs.shinobi.video/ (accessed on 22 June 2026).
23. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object Detection with Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.