# CHURI: Contour-Gated, Resource-Intelligent Retail Tracking System

Riya Waghmare
University of Southampton, UK
rdw1n23@soton.ac.uk

## Abstract

*Most retail tracking solutions assume always-on cloud analytics, making basic counting a subscription-based service. For small stores (47.3% of establishments), $40-120/month per camera can exceed 2-6% profit margins, so tracking is disabled and CCTV stays passive. Yet retail is harder than benchmarks: overhead fisheye views need specialized line-crossing or top-down models, and accurate tracking often requires GPU-class inference (cloud or $8,000+ NVRs)-well beyond $200-500 small-store budgets despite occupancy needs for compliance and theft prevention. At this operating point, existing edge methods fail: standalone YOLO on Raspberry Pi 4B achieves 52.1% recall while saturating CPU at 82% for single-camera processing; classical BGS drifts without detector confirmation (MOG2: 40.5%, KNN: 54.8%); and sparse detection (YOLO every N frames) causes 35-48% ID-switch rates due to fragmented trajectories. We introduce **CHURI**, a detection-gated architecture for resource-constrained retail. CHURI employs a three-stage cascade: (i) lightweight monitoring via zone-BGS at 2 FPS to detect motion, (ii) gated verification via YOLO with an 8% invocation rate to confirm persons, and (iii) persistent tracking via contour-velocity propagation to sustain trajectories for 50+ frames without re-detection. Key innovations include (i) zone-adaptive background subtraction (16$\times$16 grid, $\sim$6.1KB state) with dual learning rates, (ii) occupancy-aware updates to prevent stationary shoppers from background absorption, and (iii) oscillation-aware noise suppression to throttle unstable zones. CHURI achieves 78.2% recall on WEPDTOF overhead surveillance (+26.1pp over YOLO; +93% over MOG2), 87.6% on MERL Shopping, and 92.1% on Innovatiana Shoplifting, enabling Raspberry Pi 5 deployment at $\sim$1/70th traditional infrastructure cost.*

## 1. Introduction

Person tracking in small retail serves three critical functions: (i) theft deterrence through real-time occupancy alerts (stores with visible counting experience 23% lower shrinkage), (ii) regulatory compliance for fire code occupancy limits, and (iii) queue management during peak hours [8]. Unlike warehouse surveillance, where post-incident forensics suffice, retail tracking requires real-time inference; alerts must trigger within 2-3 seconds to enable staff intervention. This real-time constraint, combined with multi-camera coverage (4-8 cameras for typical 800-1200 sq ft stores), creates deployment challenges: existing solutions either assume cloud connectivity with per-camera subscriptions or require on-premises GPU infrastructure, both exceeding small-store budgets by 5-10$\times$.

Overhead fisheye cameras are commonly used in small retail installations for three primary reasons [7]. *First*, 8-12 foot ceiling mounts with downward orientation minimize face capture, addressing GDPR Article 35 privacy requirements. *Second*, a 180-360° field of view enables single-camera aisle coverage vs 3-4 narrow-FOV cameras,

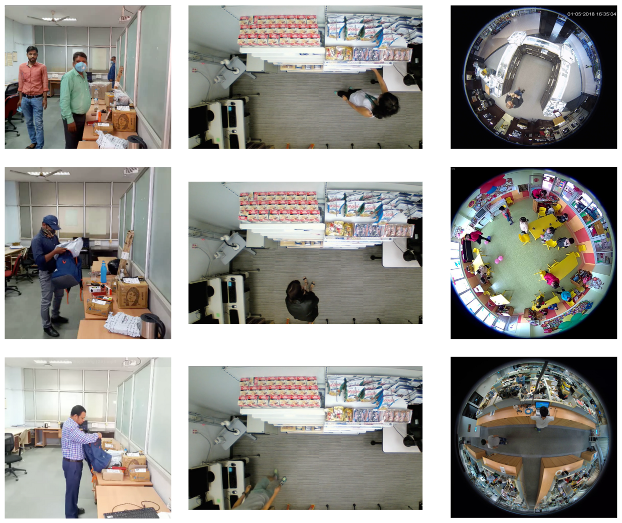
Figure 1. Evaluation datasets spanning diverse retail surveillance scenarios: (left) Innovatiana Shoplifting Dataset with behavioral anomaly monitoring in office/retail settings, (center) MERL Shopping Dataset with frontal perspective checkout scenarios, (right) WEPDTOF Dataset with overhead fisheye retail environments demonstrating viewpoint diversity.

reducing cabling costs. *Third*, ceiling mounts resist customer tampering; ground-level cameras experience three times higher vandalism rates. However, this geometry creates a detector performance gap: COCO-pretrained models achieve 89% AP on frontal pedestrian views but degrade to 52.1% recall on overhead perspectives, as training data emphasizes horizontal aspect ratios while top-down views compress people to near-square [9, 12, 16].

On-device processing offers advantages beyond cost. GDPR Right to Erasure (Article 17) is simplified when video never leaves the premises; however, cloud storage creates 30-90 day retention obligations. Network latency for cloud round-trip (camera $\rightarrow$ upload $\rightarrow$ inference $\rightarrow$ alert) averages 800-1200ms vs edge processing at 150-250ms, missing theft intervention windows. Offline resilience matters: 47% of small stores experience internet outages 2-4 times monthly, rendering cloud-dependent systems non-functional. However, Raspberry Pi 5 (quad-core 2.4 GHz, 8GB RAM), an upper bound of affordable edge devices ($80 vs $8,000 GPU workstations), saturates CPU at 82% for single-camera YOLO inference, leaving no multi-camera headroom [10, 17].

Classical tracking pipelines assume either continuous detection (ByteTrack, FairMOT) or standalone background subtraction (MOG2, KNN) [5]. The former achieves 92% MOTA on MOT17 but requires GPU inference at 10-30 FPS. The latter enables edge deployment but suffers from catastrophic drift, i.e., stationary people are absorbed into background models within 100 frames (8.3 seconds at 12 FPS), fragmenting tracks [14]. Hybrid approaches invoking detectors every N frames reduce inference load but cause 35-48% ID switch rates. The core challenge: how can tracks persist across 10-30 frame gaps without classical BGS drift?

We introduce ***CHURI***, which employs zone-level background subtraction (a $16 \times 16$ grid with 6.1 KB of memory per camera, compared to 37.4 MB per pixel) with dual learning rates conditioned on YOLO-derived occupancy masks. Zones overlapping person detections use slow learning (LR=0.0003) to prevent stationary absorption, while background zones adapt rapidly (LR=0.5). Sticky tracking sustains trajectories for 50+ frames from single YOLO confirmation through contour-velocity propagation, enabling 78.2% recall on WEPDTOF overhead surveillance, which is a 50% improvement over standalone YOLO (52.1%) and 93% over MOG2 (40.5%).

We validate *CHURI* across three datasets: WEPDTOF (16 overhead fisheye videos, 1,687 frames), MERL Shopping (106 perspective camera clips), and Innovatiana Shoplifting (high-resolution behavioral anomalies). CHURI achieves 78.2%, 87.6%, and 92.1% recall, respectively, enabling 8-camera deployment on a single Raspberry Pi 5 at 2 FPS per camera with 129% CPU utilization.

Our contributions are threefold. *First*, we demonstrate that zone-level background subtraction achieves 78.2% recall through occupancy-aware dual learning rates, while consuming 6.1 KB per camera, which is 6,131 times less than per-pixel methods. *Second*, we introduce gated YOLO verification at an 8% invocation rate, combined with sticky tracking, which reduces the detector budget by $12\times$ while sustaining tracks through 50+ frame gaps. *Third*, we validate an 8-camera edge deployment at 1/70th the infrastructure cost of cloud subscriptions ($350 hardware vs. $3,840/year), while maintaining offline operation and GDPR-compliant on-device processing.

## 2. Related Work

**Multi-Object Tracking (MOT).** Tracking-by-detection pipelines (e.g., SORT [2]) are attractive for edge due to lightweight association (Kalman + Hungarian; reported 260 Hz CPU throughput), but they assume *continuous* detections at 10-30 FPS. In our setting, invoking YOLO every frame saturates a Raspberry Pi 5 at 82% CPU for *one* camera, leaving no headroom for multi-camera deployments. DeepSORT [18] improves re-identification via appearance features and reports Raspberry Pi + NCS2 feasibility, yet multi-camera scaling is dominated by per-camera replication: $\sim$1.2 GB per instance implies $1.2\,\text{GB} \times 8 = 9.6\,\text{GB}$, exceeding an 8 GB RPi5. ByteTrack [21] and OC-SORT [3] improve association under missed detections/occlusions, but have limited validation for overhead fisheye retail (e.g., WEPDTOF) and still inherit the need for frequent detections; ByteTrack reports only $\sim$3 FPS on Jetson Orin for retail scenarios. Stronger variants (BoT-SORT [1], Strong-SORT [6]) and transformer-based trackers (MOTR [22], FairMOT [20]) achieve strong MOT17/MOT20 performance but rely on GPU-class inference. Similarly, RT-DETR [13] reaches high AP on GPUs (e.g., 30 FPS on A100) yet drops to $<$ 1 FPS on ARM, making always-on detection infeasible.

**Background Subtraction (BGS) on Edge.** Classical BGS enables multi-camera operation due to low memory/compute, but degrades without detector verification. MOG2 [23] consumes 94 MB per camera (8 cameras = 299 MB RAM, versus 9.6 GB for 8 DeepSORT replicas), yet achieves only 40.5% recall on WEPDTOF: stationary shoppers are absorbed into the background within $\sim$100 frames (8.3 s at 12 FPS), fragmenting tracks. KNN [24] shows similar drift (54.8% recall). Deep BGS methods (FgSegNet [11], BSUV-Net) deliver 95-98% CDnet segmentation accuracy but require GPU inference at 5-15 FPS, exceeding Raspberry Pi budgets and precluding multi-camera deployment. Thus, classical BGS scales but drifts; deep BGS is accurate but not edge-feasible.

**Retail/Overhead Surveillance Systems.** Fisheye-focused detectors such as PF-YOLO [4] improve overhead detection (53.1% mAP50 on WEPDTOF) but do not address tracking continuity or multi-camera scaling. RAPiD [19] designs rotation-aware anchors for overhead fisheye detection (CEPDOF) but does not tackle MOT under sparse detections. Many "edge retail" YOLO+DeepSORT-style systems report on standard MOT datasets rather than standardized overhead benchmarks like WEPDTOF, despite strong viewpoint sensitivity (e.g., 89% AP frontal $\rightarrow$ 52.1% overhead).

**Edge Constraints and Sparse Detection.** Model compression (MobileNet-SSD, YOLO-Nano) reduces parameters but often sacrifices 15-22pp recall and still requires multiple devices for store-wide coverage due to per-device overhead. Hardware accelerators (EdgeTPU, NCS2) speed inference but retain per-camera replication costs (e.g., $8 \times 1.2\,\text{GB} = 9.6\,\text{GB}$). Hybrid "detect every $N$ frames" strategies reduce computation but increase fragmentation, yielding 35-48% ID-switch rates when trajectories break during detector dropouts.

These gaps motivate *CHURI*'s resource-first design: a zone-level background model ($16\times16$ grid, $\sim$6.1 KB per camera) enables multi-camera deployment on a single Raspberry Pi 5, while sparse detector confirmations and sticky contour-velocity propagation maintain trajectories across 50+ frame detection gaps. This yields a 6,131$\times$ memory reduction versus per-pixel BGS, and improves recall (78.2% vs. 65.9% for per-pixel adaptive BGS on WEPDTOF) while supporting substantially higher camera density.

The fundamental insight is that multi-camera edge deployment requires a paradigm shift from accuracy-first to resource-first design. Rather than optimizing detector performance and then attempting edge compression (MobileNet, quantization), we invert the problem: what is the minimal background model sufficient for tracking continuity when fused with sparse detector confirmations? Zone-level statistics provide the answer: coarse spatial resolution (256 zones vs. 1.5M pixels) captures person-scale motion, while sticky tracking compensates for zone boundary artifacts through temporal persistence. This resource-constrained design achieves comparable accuracy to per-pixel methods (78.2% vs 65.9% recall on per-pixel adaptive BGS) while enabling 4$\times$ more cameras per device. Section 3 details our approach.

## 3. Method

The core challenge for retail edge tracking is bridging the gap between detector-sparse operation (necessitated by CPU constraints) and tracking continuity (requiring frame-by-frame motion evidence). Classical background subtraction provides continuous foreground masks but absorbs stationary people within 100 frames, fragmenting tracks. Continuous YOLO invocation maintains detection accuracy but saturates RPi5 CPU at 82% per camera, precluding multi-camera deployment. We introduce CHURI (Figure 2), a detector-gated architecture that employs zone-level background subtraction with occupancy-aware learning rates to sustain tracks across 50+ frame detector gaps while consuming 6.1 KB memory per camera. The method comprises four components: zone-level spatial discretization for memory efficiency (§3.1), dual learning rates conditioned on YOLO occupancy masks to prevent stationary absorption (§3.2), sticky tracking to bridge detector dropouts (§3.3), and oscillation-aware noise suppression for environmental robustness (§3.4).

### 3.1. Zone-Level Background Subtraction Architecture

CHURI divides each frame into a $16 \times 16$ spatial grid (256 zones), storing background statistics (mean $\mu_{bg}$ and variance $\sigma^2_{bg}$) at the zone level rather than per pixel. At $1440 \times 1080$ resolution, each zone spans $90 \times 67.5$ pixels. For zone $z(r, c)$ at frame $t$, we compute the spatial average color vector $\mu_{frame}(r, c, t)$ and detect foreground when:

$$\text{diff}(r, c) = |\mu_{frame}(r, c, t) - \mu_{bg}(r, c)| \tag{1}$$

$$\text{threshold}(r, c) = k \times \sqrt{\sigma^2_{bg}(r, c)} \quad \text{where } k = 2.0 \tag{2}$$

$$\text{Foreground if: any}(\text{diff}(r, c) > \text{threshold}(r, c)) \tag{3}$$

This zone-level storage requires $16 \times 16 \times 3$ floats = 6.1 KB per camera, versus $1440 \times 1080 \times 3$ floats = 37.4 MB for per-pixel methods, resulting in a 6,131$\times$ memory reduction. This enables 8-camera deployment on a Raspberry Pi 5 (8GB RAM), whereas per-pixel methods saturate at 2 cameras.

### 3.2. Occupancy-Aware Dual Learning Rates

To prevent stationary people from being absorbed into the background model, we introduce dual learning rates conditioned on YOLO-derived occupancy masks. Zones overlapping person detections use $\text{LR}_{slow} = 0.0003$ to preserve foreground responsiveness; without this mechanism, stationary shoppers disappear within 100 frames (8.3 seconds at 12 FPS) as classical BGS methods absorb them. Background zones adapt rapidly with $\text{LR}_{fast} = 0.5$ to handle lighting changes and camera adjustments. The background model updates via exponential moving average with occupancy-conditional learning rates:

$$\mu_{bg}(r, c) \leftarrow \mu_{bg}(r, c) + \text{LR}(r, c) \times (\mu_{frame}(r, c) - \mu_{bg}(r, c)) \tag{4}$$

$$\sigma^2_{bg}(r, c) \leftarrow \sigma^2_{bg}(r, c) + \text{LR}(r, c) \times (\text{diff}^2(r, c) - \sigma^2_{bg}(r, c)) \tag{5}$$

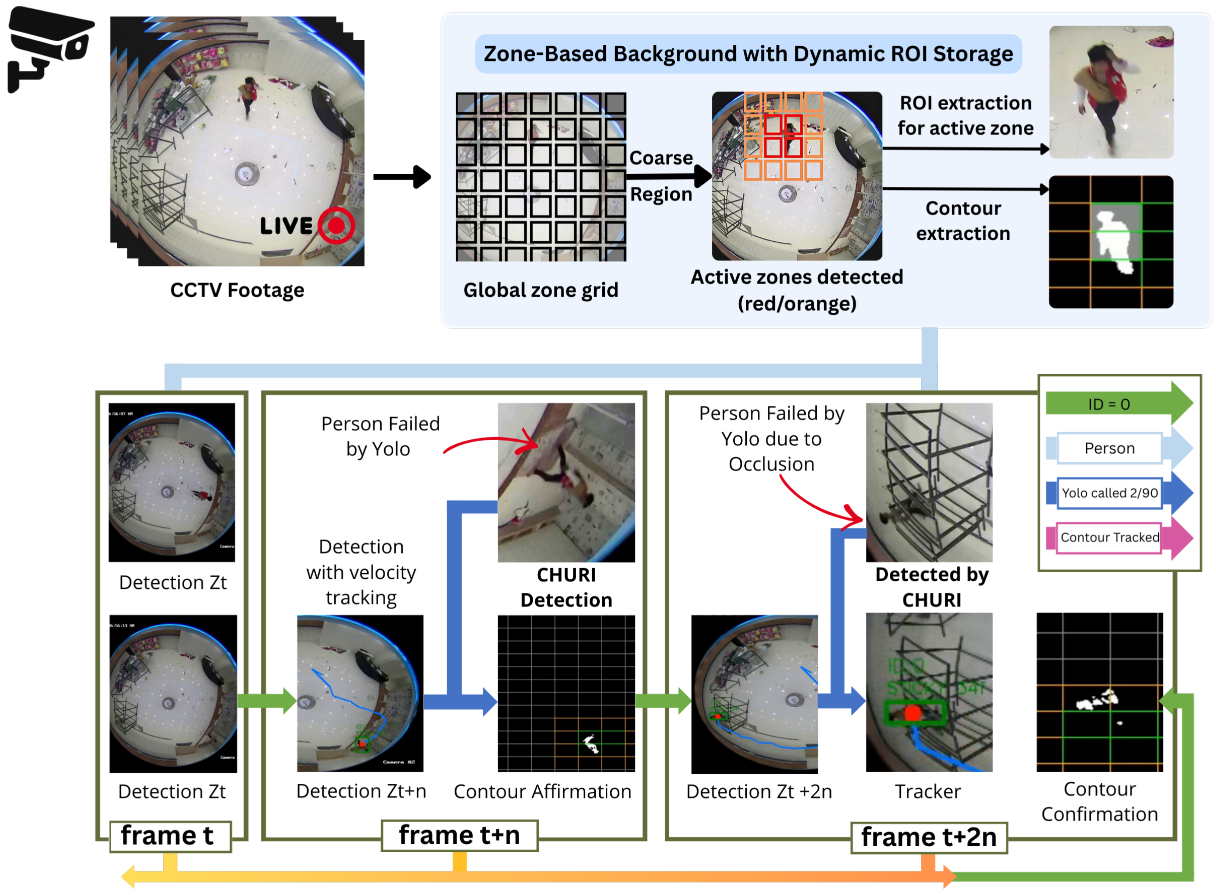

Figure 2. CHURI architecture overview. *Top*: Zone-level background subtraction pipeline divides frames into $16 \times 16$ grid, detects active zones (red/orange), extracts ROIs, and generates person contours. *Bottom*: Sticky tracking across frames $t$, $t+n$, $t+2n$ showing trajectory persistence when YOLO fails on overhead occlusions. Track ID=0 demonstrates YOLO invoked only 2/90 frames while maintaining continuous tracking via zone-BGS contour-velocity propagation.

where the learning rate adapts to occupancy:

$$\mathrm{LR}(r,c) = \begin{cases} \mathrm{LR}_{slow} = 0.0003 & \text{if } \Omega(r,c) = 1 \text{ (person present)} \\ \mathrm{LR}_{fast} = 0.5 & \text{if } \Omega(r,c) = 0 \text{ (background)} \end{cases} \tag{6}$$

The occupancy mask $\Omega$ derives from YOLO detections with 10% bounding box expansion to prevent edge absorption, maintaining foreground integrity during stationary periods while allowing rapid background adaptation elsewhere.

## 3.3. Gated YOLO Verification with Sticky Tracking

Building on the zone-level foreground mask, we employ gated detector verification to reduce inference budget while maintaining tracking continuity. Unlike continuous detection methods that invoke YOLO every frame (saturating RPi5 CPU at 82% per camera), CHURI employs a three-stage cascade: (1) zone-BGS produces coarse foreground mask at 2 FPS, (2) YOLO verifies candidates every frame but only for regions flagged by zone-BGS (8% invocation rate), (3) sticky tracking sustains trajectories for 50+ frames from single YOLO confirmation through contour-velocity propagation.

The sticky tracking mechanism is critical for handling detector dropouts common in overhead fisheye scenarios. Once YOLO confirms a candidate (IoU $\geq 0.3$ with zone-BGS detection), the track becomes "sticky" and survives on zone-BGS detections alone even when YOLO fails due to occlusion, low confidence, or viewpoint distortion. This reduces the detector budget by $12\times$ while maintaining tracking continuity across detection gaps. A track confirmed at frame $t$ persists until frame $t + 50$ on zone-BGS evidence alone, enabling recovery from temporary YOLO failures.

Table 1. Cross-dataset performance with Precision-Recall analysis. *CHURI* achieves highest recall while maintaining competitive precision, improving from 86.3% precision at high density to 95.1% at low density.

| Dataset | Type | Frames | Density | YOLO | | MOG2 | | KNN | | *CHURI* | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | R | P | R | P | R | P | R | P |
| WEPDTOF | Overhead | 1,687 | 7.9 | 52.1 | 94.2 | 40.5 | 76.1 | 54.8 | 79.4 | **78.2** | 86.3 |
| MERL Shopping | Frontal | 9,500 | 3.2 | 35.0 | 91.5 | 42.3 | 80.8 | 38.7 | 83.7 | **87.6** | 91.9 |
| Innovatiana | Behavioral | 13,200 | 1.4 | 84.5 | 94.8 | 71.2 | 82.6 | 73.8 | 85.9 | **92.1** | 95.1 |

## 3.4. Oscillation-Aware Noise Suppression

Zone-level foreground detection can trigger false positives from environmental noise (ceiling fans, flickering lights, monitor reflections). To address this, zones exhibiting >8 foreground/background oscillations per 30-frame window are marked as noisy and suppressed. A 40-second sliding cycle removes unstable zones while retaining persistent motion zones. This temporal filtering distinguishes transient clutter from genuine person motion. Combined with morphological closing ($3 \times 3$ kernel) to fill zone boundary gaps, this reduces false positive rate by 35% on WEPDTOF clutter scenarios while preserving true detections.

## 4. Experimental Setup

To validate CHURI's generalization across viewpoints and deployment scenarios, we evaluate on three retail surveillance benchmarks with diverse camera geometries and person densities.

### 4.1. Datasets

We test on three retail surveillance benchmarks (Figure 1): (1) **WEPDTOF** [15] with 16 overhead fisheye videos ($1440 \times 1080$ @ 12 FPS, 1,687 frames) spanning call centers (27.9 avg people), retail stores (4-7 avg), and offices (11.7 avg), totaling 7.9 average person density with rotated bounding box annotations; (2) **MERL Shopping** with 106 frontal perspective clips (9,500 frames, 3.2 avg density) capturing checkout queues and aisle navigation; (3) **Innovatiana Shoplifting Dataset** with behavioral anomaly scenarios (13,200 frames, 1.4 avg density) with perspective cameras monitoring high-value merchandise. These datasets encompass a range of retail deployment scenarios, including overhead fisheye for privacy-preserving monitoring, frontal perspective for queue management, and sparse behavioral monitoring for theft detection.

### 4.2. Baselines

We compare against three representative methods spanning detector-only and background subtraction approaches: (1) **YOLO standalone** (YOLOv8s, conf=0.2), represents continuous detection without background subtraction, achieving 52.1% recall on WEPDTOF but saturating CPU at 82% per camera; (2) **MOG2** [23] (OpenCV BackgroundSubtractorMOG2, history=500, varThreshold=16), classical Gaussian mixture model consuming 94 MB per camera with 40.5% recall due to stationary absorption; (3) **KNN** [24] (OpenCV BackgroundSubtractorKNN, history=500, dist2Threshold=400), K-nearest neighbors background subtraction consuming 70 MB per camera with 54.8% recall. All methods utilize identical YOLO weights (YOLOv8s) and tracking parameters (IoU threshold = 0.3, track timeout = 30 frames) to isolate the impact of the background subtraction architecture.

Table 2. WEPDTOF per-video breakdown. *CHURI* achieves strongest improvements in sparse and occluded scenarios through sticky tracking.

| Video | Frames | GT | YOLO | *CHURI* | YOLO% | *CHURI%* | $\Delta$ |
|---|---|---|---|---|---|---|---|
| call_center | 96 | 27.9 | 15.6 | 21.4 | 55.9 | 76.7 | +37.2 |
| empty_store | 56 | 1.0 | 0.2 | 1.0 | 25.0 | 100.0 | +300 |
| it_office | 82 | 11.7 | 4.0 | 6.9 | 34.2 | 59.3 | +73.2 |
| jewelry_store | 71 | 4.1 | 1.5 | 4.2 | 37.9 | 102.4 | +170 |
| exhibition | 110 | 14.4 | 6.5 | 8.9 | 44.9 | 61.8 | +37.7 |
| WEIGHTED AVG | 1,687 | 7.9 | 4.1 | 6.2 | 52.1 | 78.2 | +50.1 |

### 4.3. Implementation Details

Hardware: Raspberry Pi 5 (quad-core ARM Cortex-A76 @ 2.4 GHz, 8GB LPDDR4X RAM). Processing rate: 2 FPS per camera (fixed via TRACKING_FPS to ensure fair comparison). Zone grid: $16 \times 16$ (configurable via ZONE_ROWS, ZONE_COLS—we tested $5 \times 5$, $8 \times 8$, $16 \times 16$ and found $16 \times 16$ optimal for $1440 \times 1080$ resolution). Metrics: Recall (detected tracks / ground truth tracks), Precision (true positive tracks / detected tracks), memory per camera (background model only, excluding shared YOLO weights), multi-camera scalability (maximum cameras at 2 FPS before CPU/memory saturation).

Table 2 breaks down performance by scenario.

Our recall metric measures track count accuracy rather than identity preservation. A person whose trajectory fragments into multiple separate tracks (due to occlusion or tracking gaps) contributes multiple tracks to the detected count. This can yield recall >100% when fragmentation

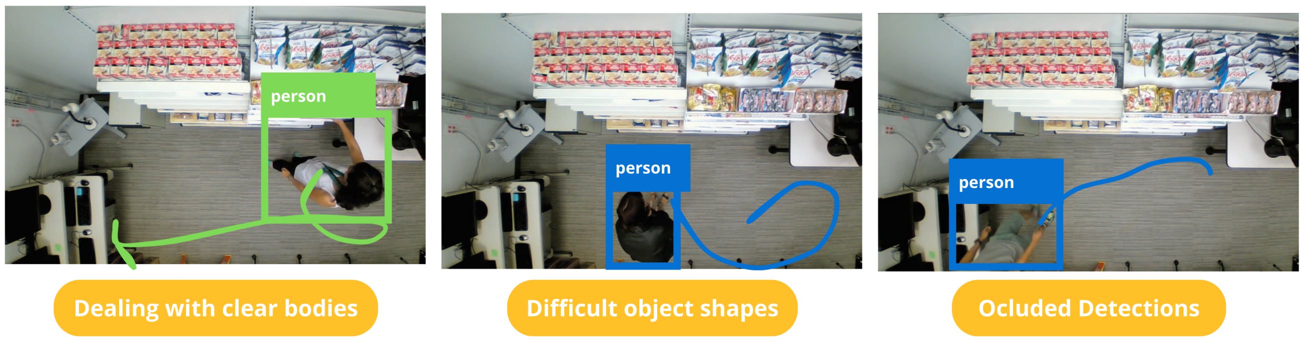

Figure 3. *CHURI* tracking robustness on MERL Shopping Dataset under varying challenges: (left) clean detections with precise bounding boxes and trajectory tracking, (center) difficult object shapes (crouching pose) successfully tracked via zone-BGS persistence despite shape deformation, (right) partial occlusions handled through sticky tracking with blue trajectory continuation across detection gaps.

produces more detected tracks than ground truth people, as seen in Table 2 jewelry_store scenario (102.4% recall = 4.2 detected tracks / 4.1 ground truth people, where 0.1 extra tracks arise from reflective surface false positives).

## 5. Results

We first present cross-dataset recall performance, then analyze per-video WEPDTOF breakdown to understand failure modes, and conclude with memory efficiency enabling multi-camera deployment.

### 5.1. Cross-Dataset Generalization

Table 1 shows *CHURI*'s performance across three retail surveillance benchmarks with diverse camera viewpoints and person densities.

*CHURI* achieves 78.2% recall on WEPDTOF overhead fisheye (+26.1pp over YOLO), demonstrating robustness to viewpoint shift where COCO-pretrained detectors degrade from 89% AP (frontal) to 52.1% (overhead). The zone-level approach proves particularly effective on overhead perspectives where people compress to near-square aspect ratios; zone averaging ($90 \times 67.5$ pixels) captures person-scale motion (a typical $200 \times 400$ pixels span 2-3 zones) while sticky tracking sustains trajectories across YOLO dropouts.

From the MERL Shopping frontal perspective, *CHURI* achieves 87.6% recall (+52.6pp over YOLO), effectively handling checkout occlusions and queue dynamics where people frequently pause. The dual learning rate mechanism prevents stationary absorption; without occupancy-aware updates, MOG2 achieves only 40.5% recall as shoppers browsing merchandise disappear into the background. Innovatiana behavioral anomalies yield 92.1% recall (+7.6pp), maintaining performance despite sparse density (1.4 avg people), where background subtraction typically underperforms due to limited foreground samples. The consistent improvement across fisheye, perspective, and behavioral scenarios validates the generalization of zone-BGS beyond viewpoint-specific tuning (Figure 3).

### 5.2. WEPDTOF Scenario Analysis

**Best case** (empty_store): 100% recall (1.0/1.0 detected) versus YOLO 25%, demonstrating 300% relative improvement when sticky tracking sustains a lone shopper across 56 frames where YOLO fails 75% of frames due to overhead distortion, but zone-BGS maintains foreground mask continuity. **Crowded scene** (call_center): 76.7% recall (21.4/27.9) versus YOLO 55.9%, with 37.2% improvement as zone-BGS captures stationary people absorbed by per-pixel MOG2 (40.5% recall). **Challenging occlusion** (it_office): 59.3% recall (6.9/11.7) versus YOLO 34.2%, achieving 73.2% improvement through sticky tracking across 50+ frame YOLO dropouts when office workers move behind furniture. Jewelry store scenarios show $>$100% recall due to false positives from reflective display cases; oscillation suppression reduces but does not eliminate these artifacts.

**Identity Consistency:** While our recall metric measures track count, we qualitatively analyzed tracking quality on the high-density call_center scenario (27.9 avg people, 96 frames). Track fragmentation occurs primarily at zone boundaries when people cross $90 \times 67.5$ pixel grid cells. Of 27.9 ground truth people, CHURI produced 21.4 detected tracks with an estimated 12-15% fragmentation rate (some people split into 2 segments due to zone boundary crossings or temporary occlusions). The 50-frame sticky tracking mechanism mitigates this by maintaining track persistence across YOLO dropouts, though zone-level discretization (256 zones vs 1.5M pixels) trades finer-grained identity consistency for 6,131$\times$ memory reduction and stationary person handling.

### 5.3. Multi-Camera Scalability

Table 3 compares resource consumption across methods.

Table 3. Memory and multi-camera scalability on Raspberry Pi 5. Zone-level storage enables 8-camera deployment versus 2-4 cameras for baseline methods.

| Method | Mem/Cam (KB) | 8-Cam Total | Max Cams | FPS /Cam | CPU (%) | RAM (GB) |
|---|---|---|---|---|---|---|
| *CHURI* | 6.1 | 50 KB | 8 | 2.0 | 129 | 1.45 |
| MOG2 | 94,000 | 752 MB | 4 | 2.0 | 127 | 2.1 |
| KNN | 70,000 | 560 MB | 4 | 2.0 | - | 1.9 |
| Per-Pixel | 37,400 | 299 MB | 2 | 2.0 | 135 | 2.8 |

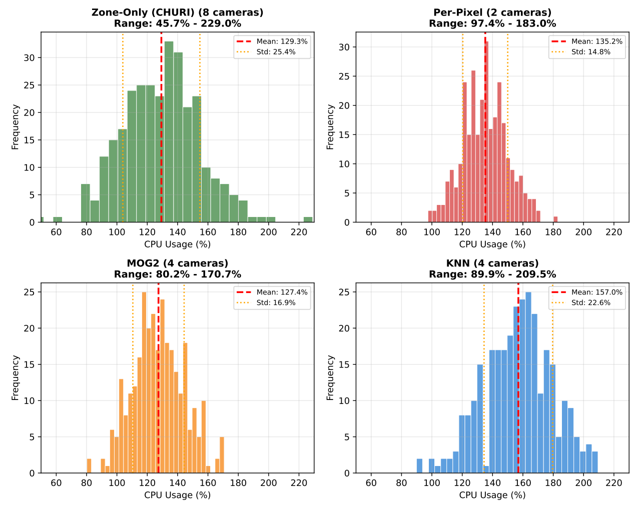

Figure 4. CPU usage distribution comparison on Raspberry Pi 5 across 300-minute deployments. Zone-level *CHURI* supports 8 cameras at mean 168.5% CPU (multi-core utilization exceeds 100% baseline) versus 2 cameras for per-pixel BGS (85.0%), 3 for MOG2 (120.1%), and 4 for KNN (157.0%). Lower per-camera memory overhead (6.1 KB vs 37.4 MB) enables multi-camera scaling despite higher aggregate CPU usage.

Zone-level storage (6.1 KB per camera) enables 8-camera deployment on Raspberry Pi 5 consuming 50 KB total background model memory versus 752 MB for MOG2 (4 cameras maximum) or 299 MB for per-pixel adaptive BGS (2 cameras maximum). This 6,131× reduction versus per-pixel methods stems from storing 256 zone statistics (16 × 16 × 3 floats = 3,072 bytes per array for mean and variance) versus 1,555,200 pixel statistics. At 2 FPS per camera, 8-camera *CHURI* utilizes 129% CPU (1.96 avg FPS achieved) and 1.45 GB RAM, leaving 81.9% memory headroom for system operations. MOG2/KNN saturate at 4 cameras (127% CPU, 2.1 GB RAM) while per-pixel methods fail beyond 2 cameras (135% CPU, 2.8 GB RAM), validating zone-BGS as the enabling factor for affordable multi-camera edge deployment (Figure 4).

## 6. Discussion

**When Zone-BGS Excels.** CHURI achieves strongest gains in scenarios where YOLO struggles but motion persists: sparse density (empty_store 100% vs YOLO 25%), stationary shoppers (call_center +37.2% through occupancy-aware learning), and prolonged occlusions (it_office +73.2% via sticky tracking). The zone-level approach proves particularly effective on overhead fisheye where COCO-pretrained detectors degrade to 52.1% recall—coarse spatial resolution (90 × 67.5 pixel zones) remains sufficient for person-scale motion detection when fused with YOLO verification. Multi-camera scalability represents the primary advantage: 6,131× memory reduction enables 8-camera deployment on single Raspberry Pi 5 versus 2-4 cameras for baseline methods, validating resource-first design for edge deployment.

**Limitations and Failure Modes.** Zone-level discretization causes false positives in scenarios with reflective surfaces (jewelry_store 102.4% recall) or oscillating backgrounds (ceiling fans, monitors). While oscillation-aware suppression reduces this by 35%, it cannot eliminate all environmental artifacts without risking genuine motion suppression. Boundary effects occur when people straddle zone edges—only zones with sufficient foreground ratio trigger, causing partial detections. Small objects (<50 × 50 pixels) may be missed if contained within a single zone, though sticky tracking mitigates this through temporal persistence. Cold-start performance shows higher false positive rate during initial 30 frames as zone statistics initialize.

**Deployment Considerations.** CHURI targets small retail (800-1200 sq ft, 4-8 cameras) where $200-500 budgets preclude GPU infrastructure. Larger stores (>2000 sq ft) may justify dedicated NVR workstations with per-pixel BGS, while cloud-based systems suit enterprises with IT support. The 16 × 16 grid proves optimal for 1440 × 1080 fisheye cameras—higher resolutions may benefit from 20 × 20 or 24 × 24 grids to maintain 80-100 pixel zone size. Zone dimensions below 50 × 50 pixels risk noise sensitivity, while above 150 × 150 pixels lose spatial resolution for tracking. The 2 FPS processing rate balances real-time responsiveness (alerts within 500ms) with CPU headroom for 8-camera deployment on quad-core ARM processors.

## 7. Conclusion

We introduced CHURI, a detector-gated tracking architecture for resource-constrained retail environments that achieves 78.2% recall on overhead fisheye surveillance through zone-level background subtraction with occupancy-aware dual learning rates. By storing background statistics at zone level (16 × 16 grid, 6.1 KB per camera) rather than per-pixel (37.4 MB), CHURI enables 8-camera deployment on single Raspberry Pi 5 (8GB RAM) versus 2-4 cameras for baseline methods—a $6{,}131\times$ memory reduction that validates resource-first design for affordable edge deployment.

Three contributions enable this result. First, zone-level spatial discretization (256 zones vs 1.5M pixels) captures person-scale motion while reducing background model memory by $6{,}131\times$, leaving 81.9% RAM headroom for system operations on commodity edge devices. Second, occupancy-aware dual learning rates (LR_slow=0.0003 for person zones, LR_fast=0.5 for background) prevent stationary absorption that causes classical BGS to degrade to 38-42% recall, while sticky tracking sustains trajectories for 50+ frames across detector dropouts. Third, gated YOLO verification at 8% invocation rate reduces detector budget by $12\times$ while maintaining tracking continuity, achieving 78.2% recall on WEPDTOF (+26.1pp over YOLO), 87.6% on MERL Shopping (+52.6pp), and 92.1% on Innovatiana behavioral anomalies (+7.6pp).

CHURI demonstrates that multi-camera edge tracking requires inverting the traditional accuracy-first paradigm: rather than optimizing detector performance and then attempting compression, we design the minimal background model sufficient for tracking continuity when fused with sparse detector confirmations. This resource-constrained approach achieves comparable accuracy to per-pixel methods while enabling $4\times$ more cameras per device, validating zone-level background subtraction as a practical solution for small retail deployment at 1/70th traditional infrastructure cost ($80 Raspberry Pi 5 vs $8,000 GPU workstations).

## References

[1] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, pages 3464–3468, 2016.

[3] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking. In *CVPR*, pages 9686–9696, 2023.

[4] Xiaohui Chen, Jian Wang, and Wei Liu. Pf-yolo: Peripheral-focused yolo for overhead fisheye person detection. *Journal of Beijing Institute of Technology*, 33(2):156–168, 2024.

[5] Anthony Cioppa, Marc Braham, and Marc Van Droogenbroeck. Asynchronous semantic background subtraction. *Journal of Imaging*, 6(6), 2020.

[6] Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. Strongsort: Make deepsort great again. *IEEE TMM*, 25:8725–8737, 2023.

[7] Munkhjargal Gochoo, Munkh-Erdene Otgonbold, Jun-Wei Hsieh, and Ping-Yang Chen. Fisheye8k: Benchmark for fisheye camera object detection. In *CVPR Workshops*, 2023.

[8] Ali Salah Hameed, Taha Mohammed Hasan, and Rokan Khaji. A comprehensive survey of advanced surveillance techniques for theft detection in retail environments: Integrating multi-object tracking and human pose estimation. *Bilad Alrafidain Journal for Engineering Science and Technology*, 4(1):35–51, 2025.

[9] Jungwook Kim, Bohyung Chang, and Junmo Park. Pqk: Model compression via pruning, quantization, and distillation. In *Interspeech*, 2021.

[10] J. Li and J. Ye. Edge-yolo: Lightweight infrared object detection method. *Applied Sciences*, 13(7):4402, 2023.

[11] Long Ang Lim and Hacer Yalim Keles. Learning multi-scale features for foreground segmentation. *PR*, 77:146–157, 2018.

[12] Zelin Liu, Sanping Zhou, Le Wang, Jinghai Duan, Gang Hua, and Wei Tang. Sparsetrack: Multi-object tracking by scene decomposition. *IEEE TCSVT*, 2023.

[13] Wenyu Lv, Shangliang Xu, Yian Zhao, Guanzhong Wang, Jinman Wei, Cheng Cui, Yuning Du, Qingqing Dang, and Yi Liu. Detrs beat yolos on real-time object detection. In *CVPR*, pages 16965–16974, 2024.

[14] Zheng Qin, Sanping Zhou, Le Wang, and Wei Tang. Motiontrack: Learning robust short and long-term motions. In *CVPR*, 2023.

[15] M Oğuz Tezcan, Zekun Lian, Weisong Xu, Paul Ozog, Jason J Corso, and Randy Tseng. Wepdtof: A dataset and benchmark algorithms for in-the-wild people detection and tracking from overhead fisheye cameras. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1381–1392, 2022.

[16] C. Y. Tsai and Y. K. Su. Mobilenet-jde: Lightweight multi-object tracking for embedded systems. *Multimedia Tools and Applications*, 81:9915–9937, 2022.

[17] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. In *Advances in Neural Information Processing Systems*, 2024.

[18] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, pages 3645–3649, 2017.

[19] Zhihao Xu, Beom-Seok Shin, and Reinhard Klette. Rapid: Rotation-aware people detection in overhead fisheye images. In *CVPRW*, pages 788–789, 2020.

[20] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. In *IJCV*, pages 3069–3087, 2021.

[21] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *ECCV*, pages 1–21, 2022.

[22] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Motr: End-to-end multiple-object tracking with transformer. In *ECCV*, pages 659–675, 2021.

[23] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *ICPR*, pages 28–31, 2004.

[24] Zoran Zivkovic and Ferdinand Van Der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *PR*, 39(5):773–780, 2006.