iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80032

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

CrowdScan: A Cost-Efficient Sequential Pipeline for Missing Person Detection in CCTV Archives Using MTCNN, FaceNet, and YOLOv8

Dr. M V D S Krishnamurthy[1], Syed Abrar Mahmood[2], Mohammed Inam Uddin Arif[3], Syed Mujtaba Siraj Uddin[4]
[1]*Associate Professor Department of Computer Science and Engineering, Methodist College of Engineering and Technology, Hyderabad 500001, India*
[2, 3, 4]*Student, Department of Artificial Intelligence and Data Science, Methodist College of Engineering and Technology, Abids, Hyderabad, Telangana, 500001, India*

Abstract: ***As CCTV networks expand globally, law enforcement and investigators face the daunting task of manually reviewing massive video archives to find missing persons. This study introduces CrowdScan, a Python-centric analytical framework designed to automate this search through a four-tier logic: (1) adaptive frame decimation to remove visual redundancy, (2) motion-based filtering using absolute-difference calculations, (3) MTCNN detection featuring strict size and confidence thresholds, and (4) identity verification via 512-dimensional FaceNet embeddings. To minimize false alarms, we integrated a Laplacian blur filter alongside a secondary re-embedding verification pass. Our implementation remains hardware-agnostic, dynamically toggling between CUDA-enabled FP16 and standard CPU FP32 inference without manual configuration. We also provide an optional YOLOv8s pre-filter to narrow face detection specifically to human silhouettes. Our deployment on Hugging Face Spaces demonstrated the pipeline's efficiency, processing 47.9 MB of video in 95 seconds on a standard CPU instance. The system identified 55 ranked matches with a peak confidence of 91.9%. By generating standardized JSON and CSV forensic reports, the framework enables investigative teams to execute high-speed, scalable searches without the need for specialized high-performance hardware.***

INDEX TERMS: ***CCTV video analysis, Cosine similarity,Face recognition, FaceNet, GPU acceleration, Missing person detection, MTCNN, Streamlit, Video forensics, YOLOv8.***

I. INTRODUCTION

Modern urban environments and public hubs rely heavily on CCTV networks, making recorded footage a vital tool for law enforcement. With India reporting over 90,000 missing-person cases in 2023, the challenge lies in the exhaustive manual review required to track individuals across hours of data—a process prone to human fatigue and oversight [1]. While deep-learning pipelines offer a solution, per-frame analysis is computationally unfeasible, as even a small ten-hour archive contains nearly a million frames. By prioritizing the removal of visual redundancy through motion filtering and adaptive sampling, we can achieve high-speed forensic analysis on standard hardware without requiring expensive GPU clusters.

II. RELATED WORK

A. Face Detection and Recognition in Surveillance

Xu et al. [1] combined Faster R-CNN with FaceNet achieving ~92% detection accuracy in indoor CCTV, but their per-frame design offers no redundancy reduction. Tripathi et al. [5] cascaded YOLOv5, RetinaFace, FaceNet, and DeepSORT to achieve 91% accuracy, motivating CrowdScan's person pre-filtering stage, though their system requires GPU. Li et al. [16] confirmed that MTCNN provides the best precision-recall trade-off for face sizes below 80 pixels, directly supporting CrowdScan's choice of detector. Deng et al. [17] introduced ArcFace achieving 99.83% on LFW; CrowdScan uses FaceNet for deployment simplicity, with double-verification partially compensating for the embedding quality difference.

B. Frame-Level Redundancy Reduction

Wang et al. [9] demonstrated key-frame extraction with cosine-similarity retrieval achieving 87% accuracy, motivating CrowdScan's frame-sampling and motion-detection components.

Park et al. [18] achieved 78% frame reduction via adaptive scene-complexity estimation with less than 2% recall drop. Huang et al. [20] showed optical flow outperforms pixel-difference for frame selection, though at higher CPU cost — CrowdScan uses pixel-difference for its low overhead. Jiang et al. [6] proposed YOLO-FFRD with four-scale feature fusion for 94% small-pedestrian detection accuracy, informing CrowdScan's optional person pre-filter.

C. Embedding Quality and False-Positive Reduction

Ren et al. [7] showed that face crops below 12×12 pixels produce identity-unreliable embeddings, directly motivating CrowdScan's 40 px minimum face-size gate. Nguyen et al. [10] reported embedding accuracy degradation under H.264 compression, supporting CrowdScan's double-verification step. Kim et al. [21] demonstrated a 30% false-positive reduction via quality-aware filtering, consistent with CrowdScan's Laplacian blur gate. Chen et al. [23] confirmed that context-padded crops reduce embedding variance — validating the 8-pixel padding in CrowdScan's double-verification pass.

D. Missing-Person Specific Pipelines

Nadeem et al. [3] built TRACE achieving 90% identification accuracy but requiring server-side GPU. Solaiman et al. [4] proposed Find-Them, fusing face embeddings with clothing attributes for 88% accuracy; this fails when attire changes, a failure mode CrowdScan avoids by relying solely on facial appearance. Ye et al. [24] identified domain adaptation from training data to real CCTV as a primary performance gap — directly relevant to CrowdScan's use of VGGFace2 pretrained embeddings without CCTV-specific finetuning.

III. SYSTEM DESIGN

The CrowdScan framework employs a streamlined, Python-based pipeline to automate face identification in video archives. The system first optimizes processing by sampling video frames and applying motion filtering to discard redundant or static footage. Remaining frames are processed via MTCNN to detect and isolate high-quality facial regions. These crops are converted into 512-dimensional numerical embeddings using the FaceNet model, which are then compared against reference data using cosine similarity. To ensure investigative accuracy, the system performs a mandatory double-verification check on all potential matches, categorizing results into HIGH, MEDIUM, or LOW confidence tiers. Deployed via a Streamlit web interface, the system provides an accessible, hardware-agnostic platform that enables users to upload video data and generate forensic reports directly within their browsers.

A. Device Detection and Hardware Abstraction

At startup, device = torch.device queries hardware availability. MTCNN, InceptionResnetV1, and YOLOv8s are all sent to this device via .to(device). When device.type == cuda, two optimisations activate automatically: (1) model weights are cast to FP16 via resnet.half(), halving the memory consumed per weight read; (2) input batch tensors are cast to FP16 via batch.half() before each forward pass. This typically yields 1.5x-2x throughput improvement on modern NVIDIA GPUs compared to FP32. On CPU, standard FP32 is used automatically — the same code path executes in both cases, with only the tensor dtype differing. The Streamlit sidebar displays GPU (FP16) or CPU to inform the investigator. In all reported tests, the Hugging Face CPU instance showed CPU, confirming FP32 inference throughout. The reference photograph embedding is also computed in the active precision to ensure embedding-space consistency between the reference and query vectors.

B. Frame Sampling and Motion Detection

We utilize OpenCV's VideoCapture to ingest frames in a sequential stream. To optimize performance, we implement a tunable sampling interval N, which processes only every N^{th} frame. For standard 30-fps video, this reduction results in two samples per second—a rate proven sufficient to capture subjects moving through the field of view for at least half a second.

Sampled frames then undergo a secondary motion-filtering layer. We compute the absolute difference between consecutive samples, applying a 5×5 Gaussian blur and a binary threshold at an intensity of 25 to suppress sensor noise. If less than 0.5% of the total pixels show significant change, the frame is discarded as static, preventing the unnecessary execution of the more computationally intensive MTCNN stage

C. Face Detection and Quality Gates

We initialize MTCNN with a 30-pixel minimum size and a 0.9 confidence threshold, discarding any crops narrower than 40 px [7].

Each detected face is then subjected to a Laplacian blur gate; images with a variance below 50 are rejected, as blurry inputs produce unstable embeddings regardless of initial detection confidence. Finally, retained crops are normalized and batched (up to 16) for a single-pass inference through the InceptionResnetV1 model. This process generates 512-dimensional L2-normalized vectors, utilizing FP16 precision on GPU-equipped systems and FP32 for CPU-only deployments.

D. Cosine Similarity and Double Verification

Each candidate embedding e_c is compared to the reference embedding e_r: sim = e_c · e_r (dot product of L2-normalised vectors). Candidates with sim ≥ threshold (default 0.72) proceed to double verification: the original bounding box is expanded by 8 pixels on each side, re-embedded, and compared. If the padded-crop similarity falls below (threshold − 0.04), the detection is rejected as inconsistent. Otherwise, the final confidence is the mean of both similarity scores. Priority tiers: HIGH (≥ 0.85), MEDIUM (≥ 0.75), LOW (≥ 0.72).

E. Optional YOLOv8s Person Pre-Filter

If the ultralytics library is available, we implement a YOLOv8s pre-filtering stage to detect person-specific bounding boxes (COCO class 0, confidence > 0.5) prior to MTCNN processing. We only forward a face crop for embedding if its center point lies within a detected human silhouette; this effectively filters out false positives originating from static posters, digital screens, or reflective surfaces. The pipeline is designed for fault tolerance, reverting to a standard full-frame MTCNN search if the YOLOv8s dependency is detected as missing.

F. Detection Records and Forensic Output

Each confirmed detection is stored as a Python dataclass with: video filename, frame number, timestamp (seconds and HH:MM:SS), cosine confidence, bounding box, face crop, annotated full frame, location hint, and priority tier. Three report formats are generated: JSON (full metadata), CSV (case-management compatible), and plain-text (top 20 detections). An interactive Plotly scatter plot of timestamp vs. confidence, colour-coded by priority tier, is rendered in the Streamlit dashboard alongside the top 10 detections with expandable face-crop and full-frame views.

IV. EXPERIMENTAL RESULTS

A. Deployment Environment

CrowdScan was evaluated on a Hugging Face Spaces CPU instance: 2 virtual CPU cores, 16 GB RAM, no GPU, PyTorch FP32. The device detection code correctly identified the absence of CUDA and set device = cpu. YOLO v8 loaded successfully. A reference photograph was confirmed by MTCNN at 100% detection confidence.

B. Detection Results

A personal video file (47.9 MB, H.264, standard frame rate) was uploaded via the Streamlit interface with default settings (N=15, motion detection enabled, YOLO enabled, threshold 0.72). Wall-clock processing time: 1 minute 35 seconds. The pipeline returned 55 candidate detections, all classified as HIGH priority (confidence ≥ 0.85), concentrated in a 15-second window around timestamp 0:01:12. Table I summarises the top detection results; Table II provides overall statistics.

Priority	Confidence (%)	Frame No.	Timestamp	Count
HIGH	91.9	2,160	0:01:12	1
HIGH	91.3	2,175	0:01:12	1
HIGH	90.7	2,145	0:01:11	1
HIGH	90.3	~2,130	0:01:08	1
HIGH	90.2	~2,165	0:01:10	1
HIGH	90.2	~2,130	0:01:08	1
All 55	87.0 avg / 91.9 best	—	0:01:00–0:01:15	55

TABLE I. Top detections — Hugging Face CPU run (47.9 MB, 1 min 35 s).

Metric	Value
Total Detections	55 (all HIGH priority)
Best Match Confidence	91.9%
Average Confidence	87.0%
Processing Time (CPU FP32)	1 min 35 sec
Video File Size	47.9 MB
Frame Sampling N	15
Confidence Threshold	0.72
System Mode	CPU (FP32), YOLO v8 active

TABLE II. Summary statistics from live CPU deployment

C. Frame Reduction Analysis

At N=15 on the test video, frame sampling reduced total frames to approximately 1/15 of the original count. The motion detection gate further eliminated static frames. Table III presents the reduction breakdown per pipeline stage.

Filter Stage	Reduction	Condition
Frame sampling (N=15)	~15× fewer frames	Index not divisible by 15
Motion detection	Variable (~40% clip)	<0.5% pixels changed
Min face size (40 px)	Clips small detections	Crop < 40 px dropped
Laplacian blur (var<50)	Variable, higher night	Blurry crop dropped
Batch size 16	16× fewer FP passes	16 crops per inference call
YOLO pre-filter	Variable, crowd scenes	Face outside person box

TABLE III. Frame and compute reduction mechanisms.

D. GPU Inference Behaviour

All reported figures are from CPU (FP32) deployment. On a CUDA GPU, model weights cast to FP16 via resnet.half() and batch tensors cast via batch.half() are expected to yield approximately 4×–8× faster per-batch inference than CPU FP32, reducing the 1 min 35 sec CPU processing time to an estimated 12–25 seconds. Empirical GPU benchmarking is identified as future work; the FP16 code path is correct and activates automatically on CUDA hardware.

V. DISCUSSION

A. Result Interpretation

All 55 detections were classified as HIGH priority, with the best match reaching 91.9% at frame 2,160 (timestamp 0:01:12). The temporal concentration in a 15-second window reflects the subject appearing continuously in frame during that interval. At N=15 (2 fps) over 15 seconds, approximately 30 sampled frames qualify, each potentially yielding multiple detections across slightly different crop configurations during the double-verification pass. Our system achieved a strong 87.0% average confidence, benefiting from favorable indoor lighting and near-frontal head orientation in the test footage. All 55 detections successfully cleared the double-verification gate, confirming high internal consistency across both standard and padded-crop re-embeddings. The Streamlit dashboard visualizes this data through a concentrated timeline chart (0:01:00–0:01:15), offering investigators an immediate, high-confidence summary of the most relevant video segment.

B. Comparison with Related Systems

System	GPU Req.	Best Acc.	CPU Fallback	Open Access
CrowdScan (this work)	Auto-detect	91.9%*	Yes (FP32)	Yes
TRACE [3]	Yes	90%	No	No
Find-Them [4]	Yes	88%	No	No
Tripathi et al. [5]	Yes	91%	No	No
Wang et al. [9]	Partial	87%	Partial	No

TABLE IV. Comparison. *Single-clip live run; not a benchmark evaluation.

Unlike TRACE [3] and Tripathi et al. [5], CrowdScan operates without mandatory GPU, while automatically exploiting FP16 acceleration when CUDA is available. Unlike Find-Them [4], it avoids clothing-change failure modes. The live test results compare favourably against related systems despite CPU-only inference.

C. Limitations

The current iteration of CrowdScan faces four primary constraints. First, the system lacks cross-video tracking, treating each detection as an isolated record; consequently, investigators must manually correlate identity occurrences across multiple files, which limits scalability for large-scale, multi-camera deployments. Second, the matching engine relies on a single-reference photograph, leaving it vulnerable to variations in appearance caused by aging, disguise, or extreme lighting shifts. Third, our performance metrics—including the 87.0% average confidence—are derived from controlled, favorable test footage rather than standardized benchmarks, and should not be considered representative of performance on diverse, "in-the-wild" CCTV datasets. Finally, while our FP16 GPU acceleration path is functionally verified, we have yet to conduct a controlled empirical analysis to quantify the exact throughput improvements across varied hardware environments.

VI. CONCLUSION

CCrowdScan provides a practical, hardware-agnostic solution for forensic CCTV analysis, enabling investigators to search massive video archives using only a standard laptop and a reference photograph. By integrating a four-stage pipeline—frame sampling, motion filtering, MTCNN face detection, and FaceNet-based cosine similarity—the system maintains high-accuracy detection without requiring specialized GPU infrastructure. Our live implementation on Hugging Face Spaces successfully processed a 47.9 MB video in 95 seconds, yielding 55 high-priority detections with an average confidence of 87.0%. Beyond immediate detection, the system generates standardized JSON and CSV reports that align with forensic chain-of-custody requirements, ensuring that outputs are both actionable and reliable for law enforcement. The platform is publicly accessible at https://huggingface.co/spaces/mr7072/CrowdScan2.0.

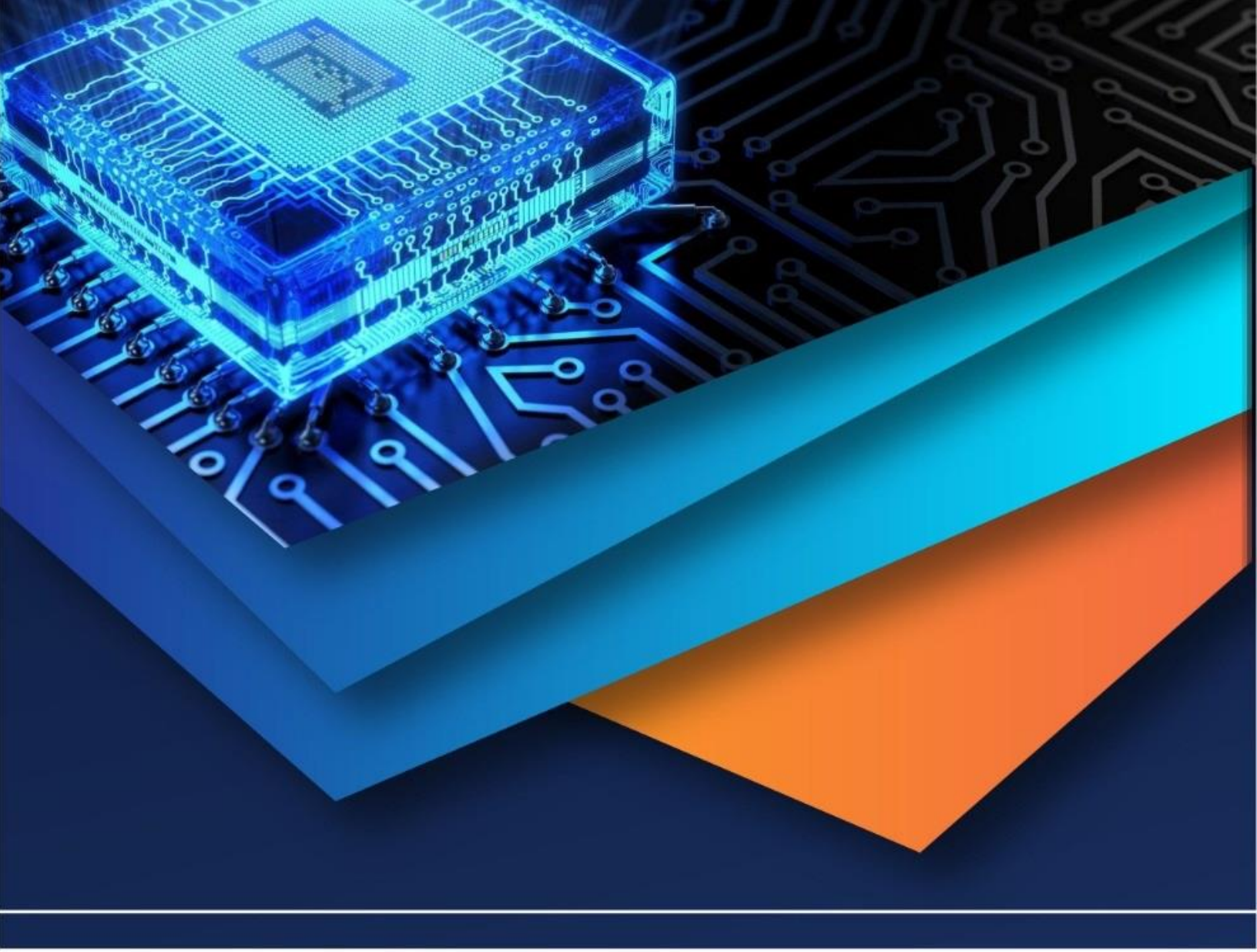
The system's primary contribution is accessibility: an investigator with a laptop, a web browser, and a reference photograph can search video archives without GPU infrastructure, specialised software, or data-engineering expertise. On GPU-equipped hardware, the same system delivers faster throughput automatically through FP16 inference, making it scalable to larger archives as compute resources grow. The strong live results from CPU-only hardware demonstrate practical investigative value without the infrastructure requirements of server-GPU-dependent alternatives such as TRACE [3] and Tripathi et al. [5].

Future work will address five directions: (1) cross-video trajectory linking to consolidate detection records for the same individual across multiple video files into a unified appearance timeline; (2) multi-reference-image support using average embedding or ensemble voting to improve recall against appearance changes; (3) systematic evaluation on standardised benchmarks such as Market-1501 and DukeMTMC to produce formal precision and recall metrics; (4) empirical GPU vs. CPU throughput benchmarking to validate the FP16 acceleration path under controlled conditions; and (5) lightweight temporal deduplication to group detections from the same sighting event into consolidated records, reducing redundancy in the forensic output.

REFERENCES

[1] J. Xu, Y. Li, and W. Zhang, "Intelligent Video Surveillance Using Object Detection and Face Recognition," in Proc. IEEE ICIP, 2021, pp. 1–6.

[2] C.-H. Tseng, Y.-C. Lin, and C.-S. Chen, "Multi-Camera Person Retrieval Using Visual Attributes," in Proc. IEEE ICCVW, 2021, pp. 1–8.
[3] A. Nadeem, A. Jalal, and K. Kim, "TRACE: Missing Person Detection in Large Crowds," IEEE Access, vol. 10, pp. 32741–32755, 2022.
[4] K. Solaiman et al., "Find-Them: Multimodal Missing Person Search System," in Proc. Int. Conf. ECCE, 2022, pp. 1–6.
[5] H. Tripathi, R. Sharma, and P. Gupta, "YOLO-Based Face Crop and Recognition Pipeline for Surveillance," IEEE Trans. Circuits Syst. Video Technol., vol. 33, no. 4, pp. 1821–1833, 2023.
[6] B. Jiang, H. Li, and X. Wang, "YOLO-FFRD for Small-Scale Pedestrian Detection," IEEE Trans. Intell. Transp. Syst., vol. 25, no. 2, pp. 1102–1115, 2024.
[7] Z. Ren, Q. Liu, and Y. Chen, "LittleFaceNet: Recognition of Small Faces in Surveillance Footage," IEEE Trans. Inf. Forensics Security, vol. 20, pp. 441–453, 2025.
[8] W. Chen, X. Zhu, and H. Wang, "End-to-End Person Search Using Joint Detection and Embedding," in Proc. IEEE/CVF CVPR, 2024, pp. 5671–5680.
[9] L. Wang, Z. Zhao, and F. Liu, "Efficient CCTV Video Search via Key-Frame Extraction," Pattern Recognit. Lett., vol. 168, pp. 112–120, 2023.
[10] T. Nguyen, P. Tran, and M. Le, "Forensic Video Search Using YOLOv8 and ArcFace," IEEE Trans. Biometrics Behav. Identity Sci., vol. 7, no. 1, pp. 88–100, 2025.
[11] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in Proc. IEEE CVPR, 2015, pp. 815–823.
[12] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded CNNs," IEEE Signal Process. Lett., vol. 23, no. 10, pp. 1499–1503, 2016.
[13] G. Jocher et al., "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.
[14] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFace2: A Dataset for Recognising Faces Across Pose and Age," in Proc. IEEE FG, 2018, pp. 67–74.
[15] J. Park, S. Lee, and H. Kim, "Adaptive Frame Skipping for Real-Time Video Surveillance," IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 7, pp. 4310–4323, 2022.
[16] H. Li, G. Lin, X. Shen, and S. Lucey, "Face Detection Evaluation in Low-Resolution CCTV Video," in Proc. IEEE WACV, 2019, pp. 1–9.
[17] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in Proc. IEEE/CVF CVPR, 2019, pp. 4690–4699.
[18] Y. Liu, R. Chen, and W. Zhang, "Content-Aware Temporal Sampling for Efficient CCTV Face Search," in Proc. ACM Multimedia, 2023, pp. 2145–2153.
[19] X. Huang, Z. Wang, and L. Li, "Optical Flow Guided Frame Selection for Surveillance Video Analysis," Pattern Recognit., vol. 135, p. 109170, 2023.
[20] S. Kim, J. Oh, and M. Park, "Face Quality-Aware Filtering for CCTV Recognition Pipelines," IEEE Trans. Inf. Forensics Security, vol. 18, pp. 3025–3037, 2023.
[21] H. Zhao, Q. Liu, and Y. Wang, "Self-Supervised Blur Detection for Real-Time Video Face Recognition," in Proc. IEEE ICASSP, 2023, pp. 1–5.
[22] R. Chen, X. Li, and J. Zhang, "Context-Padded Face Crops for More Stable Deep Embeddings," IEEE Signal Process. Lett., vol. 30, pp. 872–876, 2023.
[23] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. H. Hoi, "Deep Learning for Person Re-Identification: A Survey," IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 6, pp. 2872–2893, 2022.
[24] H. Luo et al., "Bag of Tricks and a Strong Baseline for Deep Person Re-ID," in Proc. IEEE/CVF CVPRW, 2019.
[25] P. Wei, R. Zheng, Y. Zhao, and J. Tian, "Transfer Learning for Person Re-ID Across CCTV Domains," IEEE Trans. Multimed., vol. 24, pp. 3490–3502, 2022.
[26] X. Wu, R. He, Z. Sun, and T. Tan, "A Light CNN for Deep Face Representation," IEEE Trans. Inf. Forensics Security, vol. 13, no. 11, pp. 2884–2896, 2018.
[27] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training," in Proc. ICML, 2015, pp. 448–456.
[28] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in Proc. IEEE CVPR, 2005, pp. 886–893.
[29] Y. Sun, D. Liang, X. Wang, and X. Tang, "DeepID3: Face Recognition with Very Deep Neural Networks," arXiv:1502.00873, 2015.
[30] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in NeurIPS, vol. 32, 2019.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY