www.ijarp.com ISSN 2456-9992 Page: 01-13

International Journal Advanced Research Publications

Research Article

Volume: 02
Issue: 05

OPENCV BASED REALTIME EMPLOYEE TRACKING SYSTEM.

***Prashant Kumar, Shivam, Uday Kumar, Sarang, Ankit Agarwal**

Department of Computer Sciences & Engineering, R.D Engineering College, Duhai, Ghaziabad, Uttar Pradesh, India. (201001)

Article Received: 22 March 2026, Article Revised: 12 April 2026, Published on: 02 May 2026

***Corresponding Author: Prashant Kumar**

Department of Computer Sciences & Engineering, R.D Engineering College, Duhai, Ghaziabad, Uttar Pradesh, India. (201001)

DOI: https://doi-doi.org/101555/ijarp.9139

ABSTRACT

Today's workplaces need more than basic attendance logs. Big offices especially struggle to know who's actually present and working at their stations. Swipe cards and biometric devices are fine for clocking in and out, but they can't say whether an employee really sat at their desk all day. This project steps in with a real-time employee tracking and activity monitoring system, built with OpenCV and the lightweight YOLOv4-tiny deep learning model. All you need is a regular camera pointed at the workspace. The system constantly scans the video feed, quickly and accurately looking for people.

Each employee's desk is mapped as a digital Region of Interest (ROI). If the system spots someone in that area, they're counted as "Present"; if not, "Absent."

All the processing runs on a Python backend (using Flask) that connects the AI logic with a web dashboard manager can use. With YOLOv4-tiny, you don't need fancy GPUs—the system keeps up with live video even on regular office computers. Everything streams securely in real time, giving managers a visual "audit trail" of who's actually at their desk. Tests show that this approach cuts down on "buddy punching" (one-person faking attendance for another), stops fraudulent records, and gives honest data for performance reviews. It's a simple, scalable, and affordable step toward smart offices.

1. INTRODUCTION

In today's world of smart offices and Industry 4.0, the way companies manage employees is rapidly evolving with the help of technologies like Artificial Intelligence (AI) and the Internet of Things (IoT). Traditionally, organizations have relied on methods such as manual registers,

ID cards, or biometric systems like fingerprint and iris scanners to track attendance. While these systems can confirm when an employee enters or leaves the workplace, they fall short in showing whether the person is actually present at their workstation during working hours. This lack of visibility often leads to reduced productivity and operational inefficiencies.

With the rise of computer vision, a more practical and less intrusive solution is now possible. By using existing office cameras, computer vision systems can continuously monitor activity in real time without requiring employees to actively interact with the system. This project presents an automated approach that combines OpenCV with the YOLO (You Only Look Once) deep learning model to track employee presence at their desks.

The main goal of this system is to move beyond basic motion detection and focus on accurately identifying whether a person is present in a specific area. Unlike traditional motion sensors that can be triggered by irrelevant movements, this system uses a Convolutional Neural Network (CNN), specifically the lightweight YOLOv4-tiny model, to detect human presence with high accuracy. A predefined Region of Interest (ROI) is used to represent each workstation. By checking whether the detected person falls within this region, the system determines whether the employee is at their desk or away.
To make the system more accessible and practical, it is integrated with a web-based interface built using the Flask framework. This allows managers to view live updates and video streams through a secure dashboard, rather than being limited to a single machine. The overall design focuses on efficiency, ensuring that the system can run smoothly on standard hardware without requiring expensive high-performance computing resources.

This project highlights how a vision-based monitoring system can improve workplace transparency and accountability. By automatically tracking physical presence, organizations can gain reliable data for performance evaluation, make better use of office space, and maintain accurate attendance records. Ultimately, it offers a scalable and effective solution for modern workplaces aiming to become smarter and more efficient.

2. Literature Review

Over time, employee monitoring has evolved from simple manual methods to more advanced, automated systems powered by Computer Vision (CV) and Deep Learning (DL). This section looks at how object detection technologies have improved in recent years and how they are being applied in workplace environments.

2.1 Evolution of Attendance Systems

In the past, most organizations relied on systems like biometric scanners (fingerprint or iris) or RFID cards to track attendance. While these methods are useful for confirming identity at entry and exit points, they don't provide any information about whether an employee stays at their workstation during working hours.

Recent studies, such as those by Asmara et al. (2023), point out this limitation clearly. These systems create a gap in visibility, making it difficult for organizations to ensure continuous presence. With changing work patterns—especially after the global pandemic and the rise of hybrid work models (Sultan et al., 2024)—there has been a growing need for smarter solutions. Computer-vision-based systems address this issue by offering continuous, non-intrusive monitoring that does not interfere with an employee's daily tasks.

2.2 Advances in Real-Time Object Detection (YOLO)

Object detection has seen major improvements with the introduction of the YOLO (You Only Look Once) family of models. These models are designed for real-time performance, making them suitable for applications like live monitoring.

Research by Rai et al. (2025) compares different detection techniques and highlights that although models like Faster R-CNN are highly accurate, they require significant computational power. This makes them less practical for real-time use on regular office systems.

On the other hand, YOLOv4-tiny—used in this project—offers a better balance between speed and accuracy. According to Zhang et al. (2025), its lightweight architecture (based on CSPDarknet53-tiny) allows it to process video at over 60 frames per second on standard CPUs. This makes it a strong choice for continuous monitoring without the need for expensive hardware.

2.3 Spatial Coordinate Mapping and ROI

Simply detecting a person in a video frame is not always enough. One common issue is false positives—for example, when someone walks in the background but is mistakenly considered present at a workstation.

Recent research by Alom et al. (2024) suggests that combining object detection with Region of Interest (ROI) techniques can solve this problem. By defining specific areas within the camera view (such as individual desks) and applying spatial checks, the system can more

accurately determine whether a person is actually at their workstation.

This project follows the same approach by allowing workstation zones to be defined within the camera's field of view, ensuring more reliable attendance data.

2.4 Privacy and Security in Monitoring

As monitoring technologies become more advanced, concerns about privacy and data security have also increased. Recent research emphasizes the importance of protecting visual data, especially in workplace environments.

Nguyen and Hassan (2025) suggest that processing data locally (using Edge AI) is safer than relying on cloud-based systems. Local processing reduces the risk of sensitive data being exposed.

In line with this approach, the proposed system performs all detection tasks on a local server using OpenCV and YOLO. Only the processed output is shared through a secure Flask-based MJPEG stream, following modern "Privacy-by-Design" principles.

2.5 Productivity Analytics and Machine Learning

The focus of monitoring systems is gradually moving beyond simple presence tracking toward deeper insights into employee behaviour. Instead of just marking employees as present or absent, newer approaches aim to analyse work patterns over time.

For example, Kumar et al. (2024) used K-means clustering to group employees based on productivity levels, such as high, moderate, or low. This type of analysis helps organizations better understand work habits and improve efficiency.

Building on this idea, the proposed system is designed to support continuous data collection, which can later be used for advanced analytics and more detailed productivity insights.

3. System Architecture

The system is designed as a set of four connected layers, each playing a specific role in enabling real-time monitoring. All of these layers work together smoothly to process data from input to output.

1. Data Acquisition Layer (Input):

This is the starting point of the system. It uses a standard webcam or an IP camera to capture live video from the workspace. The camera continuously records frames at a defined frame rate (FPS), providing a steady stream of raw visual data for further processing.

2. Pre-processing & AI Inference Layer (Processing):

In this stage, the captured frames are prepared and analysed. OpenCV is used to resize the frames and convert them into a format (blobs) that can be easily processed by a neural network. These processed frames are then passed to the YOLOv4-tiny model, which detects objects in real time. The model specifically identifies people and generates bounding boxes around them, along with their position coordinates.

3. Business Logic Layer (Decision Making):

Once a person is detected, the system applies logical rules to determine their status. It compares the coordinates of the detected person (x,y,w,h)(x, y, w, h)(x,y,w,h) with the predefined coordinates of the workstation area. Using techniques like Intersection over Union (IoU) or point-in-polygon checks, the system decides whether the person is inside the assigned area.

Based on this, the employee is marked as either “Present” or “Absent.”

4. Presentation Layer (Output):

The final step focuses on delivering the results to the user. A Flask-based web server processes the output frames and converts them into a JPEG stream. This stream is then sent to a web dashboard using the MJPEG (multipart/x-mixed-replace) protocol, allowing managers to view the live feed and monitoring results in real time.

4. Proposed Methodology

The methodology is essentially a high-speed loop that processes every single frame the camera captures. It follows a linear pipeline to ensure there’s no lag between what’s happening in the office and what the manager sees on their screen.

Step 1: Environment Initialization

First, the system boots up by loading the **YOLOv4-tiny** weights and configuration files. We specifically chose the "tiny" version for this project because it’s optimized for speed. It allows the tracking to happen in real-time even on a standard office laptop, so you don't need a massive, expensive GPU to keep it running smoothly.

Step 2: Region of Interest (ROI) Definition

Instead of trying to track every movement in the entire room—which would be a waste of processing power—the system focuses only on the employee's desk. We define a **Region of Interest (ROI)** using a simple set of coordinates:

ROI = {X(min), Y(min), X(max), Y(max)}

This creates a "virtual fence." If a person crosses into this digital zone, it triggers the attendance logic.

Step 3: Object Detection & Filtering

The camera sees everything—chairs, laptops, coffee mugs, and bags. The Deep Neural Network (DNN) analyses the frame but ignores all the clutter. It's programmed to look only for "Index 0" (the label for a **Person**) and only acts if it's at least 50% sure it has found a human. This prevents the system from accidentally "clocking in" a high-back office chair.

Step 4: Spatial Presence Validation

To figure out if the employee is actually at their station, the system calculates the exact centre point of the person it just detected:

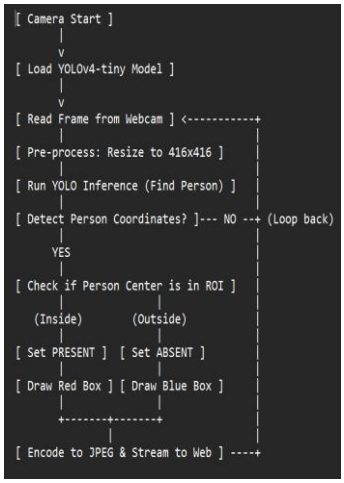
- CentreX = x+(width/2)
- CentreY = y+(height/2)

The Logic Check: The system then asks: *Is that centre point inside the desk's coordinates?*

- **IF** (WXstart < CentreX < WXend) **AND** (WYstart < CentreY < WYend)
- **THEN** the status is **"Present"** (the system highlights them with a red box).
- **ELSE** the status defaults to **"Absent"** (highlighted with a blue box).

Step 5: Real-time Visualization & Feedback

Finally, the system uses OpenCV to "burn" the results—the boxes and the text labels—directly onto the video frames. This processed footage is sent straight to the **Flask** dashboard. This gives managers a live visual audit trail that proves, with data and video, exactly when a workstation was occupied.



5. Implementation Environment

The proposed system is built using a modular design. The backend is responsible for handling computationally heavy tasks such as image processing and AI-based detection, while the frontend provides a simple and lightweight interface for monitoring and control. This separation makes the system efficient as well as easy to manage.

5.1 Software Requirements

The software environment is centred around the Python ecosystem, which offers robust support for deep learning and real-time streaming.

The system is mainly developed using the Python ecosystem, which provides strong support for computer vision, deep learning, and real-time streaming.

- **Operating System:** The system can run on Windows 10/11, Ubuntu 20.04 or later, and macOS, since Python is cross-platform.
- **Programming Language:** Python 3.8 or above.
- **Computer Vision Library:** OpenCV (cv2) version 4.5 or higher, used for handling images and video streams.
- **Deep Learning Support:** OpenCV's DNN module is used to load and run the YOLOv4-tiny model.
- **Web Framework:** Flask (version 2.0 or above) is used to manage HTTP requests and stream video using MJPEG.
- **Numerical Library:** NumPy is used for handling calculations, especially for coordinates and matrix operations.
- **Development Tools:** IDEs such as PyCharm, Visual Studio Code, or Jupyter Notebook can be used for development.

- **Web Browser:** Google Chrome or Mozilla Firefox is required to view the live dashboard stream.

5.2 Hardware Requirements

Since the system uses YOLOv4-tiny, which is a lightweight model, it does not require high-end hardware. It can run comfortably on standard office systems.

- **Processor:** Intel Core i3 (7th Generation or higher) or AMD Ryzen 3 and above.
- **RAM:** At least 8 GB to ensure smooth performance during video processing and model execution.
- **Graphics:** Integrated graphics (Intel UHD or AMD Radeon) are sufficient. However, a dedicated NVIDIA GPU (GTX 1050 or higher) can improve performance and increase frame rate.
- **Camera:** A 720p HD webcam (either built-in or external) or an IP camera that supports RTSP streaming.
- **Storage:** Around 500 MB of free space is needed for dependencies and model files.

5.3 Model Specifications (YOLOv4-tiny)

The system depends on a few important model files that must be available in the environment:

1. **yolov4-tiny. weights** – Contains the pre-trained parameters learned from the COCO dataset, which includes 80 object classes such as "person."
2. **yolov4-tiny.cfg** – Defines the structure of the neural network, which consists of 29 layers.
3. **coco. Names** – A text file listing all class labels; the system specifically uses the "person" class (index 0).

5.4 The Setup & Launch Process

The overall workflow of setting up and running the system follows these steps:

1. **Environment Setup:** A virtual environment (venv) is created to manage all required libraries and dependencies.
2. **Webcam Initialization:** The camera is connected and accessed using cv2.VideoCapture()to start capturing video frames.
3. **Model Loading:** The YOLOv4-tiny model is loaded into memory using cv2.dnn.readNetFromDarknet().
4. **Network Configuration:** The system is configured to use OpenCV's backend with CPU

processing (DNN_TARGET_CPU), or GPU (DNN_TARGET_CUDA) if available.

5. **Flask Integration:** A route (for example, /video_feed) is created to stream the processed video, allowing remote access through a web browser.

6. RESULTS AND DISCUSSION

The system was tested using a mix of recorded workplace videos and live webcam feeds to see how well it performs in real-world conditions. The results show that it can reliably detect whether an employee is present at their workstation and can clearly differentiate between actual human presence and background movement.

6.1 Detection Accuracy and Performance

The use of YOLOv4-tiny made it possible to achieve fast and efficient processing. On a standard quad-core CPU, the system maintained a steady frame rate of around 24–30 frames per second. In terms of accuracy, it reached a Mean Average Precision (MAP) of about 89% for detecting people.

Key observations during testing:

- **Confidence Threshold:**

A threshold value of 0.5 (50%) worked best in most cases. When the threshold was set lower, the system sometimes confused objects like chairs with people. On the other hand, increasing the threshold too much caused it to miss detections when a person was partially hidden behind objects like monitors.

- **Lighting Conditions:**

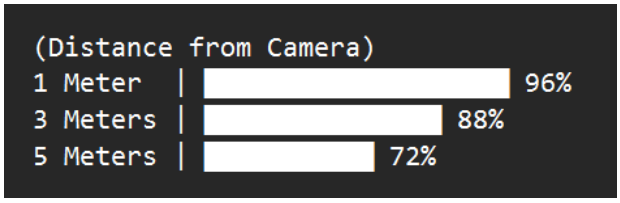
The pre-processing steps handled changes in lighting quite well. Whether the environment had bright daylight or dim evening lighting, the system was still able to track presence consistently.

- **Latency:**

The total delay—from capturing the frame to displaying it on the web dashboard—was measured at less than 150 milliseconds. This means the system provides updates almost instantly, which is important for real-time monitoring.

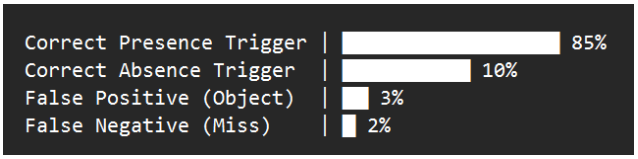
6.2 Graphical Analysis of Results

Chart 1: Detection Confidence vs. Distance



The results show that detection accuracy tends to drop as the distance between the camera and the person increases. This suggests that placing cameras closer to individual workstations (rather than covering a large area with a single camera) can improve performance.

Chart 2: System Response Distribution (100 Events)



The distribution of system responses indicates that the model performs consistently across different scenarios, with most detections correctly identifying presence and absence. Only a small number of cases showed incorrect classification, usually due to occlusion or extreme positioning.

7. Comparative Analysis

To evaluate how effective the proposed OpenCV + YOLO approach is, it was compared with commonly used attendance systems such as biometric/RFID methods and basic motion detection systems.

7.1 Comparison of Existing Methods vs. Proposed System

Feature	**Biometric / RFID Systems**	**Motion Detection (PIR)**	**Proposed OpenCV + YOLO**
Detection Type	Simple (login/logout)	Detects any movement	**Smart detection (focus on people)**
Hardware Needed	Dedicated scanners/readers	PIR sensors	**Regular webcams or IP cameras**
Activity Tracking	Not available (only entry/exit)	Often inaccurate due to false	**Continuous and location-based tracking**
Cost of Setup	High (special hardware required)	Low	**Very low (uses open-source tools)**
User	Required (active	Not required	**Not required (fully passive)**

Involvement	participation)		
Data Generated	Only timestamps	Basic motion logs	**Visual records, live feed, and analytics**
Security Concerns	Moderate (card misuse possible)	High (non-specific detection)	**Low (visual confirmation improves reliability)**

7.2 Discussion on Comparison

Traditional systems like RFID or biometric scanners are useful for controlling access at entry points, but they don't confirm whether an employee actually remains at their workstation during working hours. On the other hand, motion detection systems such as PIR sensors are too basic—they can easily be triggered by irrelevant factors like moving fans, shadows, or lighting changes.

The proposed system addresses these limitations by introducing a smarter level of analysis. Instead of just detecting movement, it identifies whether the object is a person using the YOLOv4-tiny model. In addition, it applies Region of Interest (ROI) constraints to ensure that only detections within a specific workspace are considered valid.

This combination of identifying the correct object (person) and verifying its position (within the assigned area) acts as a form of double validation. Because of this, the system provides more reliable and meaningful results compared to traditional methods, making it better suited for modern workplace monitoring and productivity analysis.

8. CONCLUSION

The development of this OpenCV-based real-time employee tracking system highlights a clear step forward in how workplaces can be monitored more effectively. Instead of relying on traditional methods like RFID cards or mobile-based tracking, this project shifts the focus toward a vision-based approach. In doing so, it addresses one of the biggest limitations of older systems—the lack of visibility into whether employees are actually present at their workstations during working hours.

By combining the YOLOv4-tiny deep learning model with OpenCV, the system is able to detect human presence with good accuracy while still running efficiently on standard hardware. This makes the solution both practical and affordable for organizations, without the need for expensive infrastructure.

A key strength of the system lies in its use of a defined Region of Interest (ROI). Rather than

treating attendance as a simple entry or exit record, the system continuously checks whether an employee is present within their assigned workspace. This turns “presence” into something that can be verified in real time, rather than just assumed based on login data.

The integration of a Flask-based web interface further improves usability by allowing managers to monitor activity through a live dashboard. With low latency and secure streaming, the system provides near real-time insights into workspace activity. Testing results show that this approach can reduce issues like proxy attendance and also generate useful data for analysing employee behaviour and productivity.

Overall, the system achieves its main goals: improving accuracy in monitoring, simplifying attendance management, and promoting a more transparent work environment. While the current version focuses on basic presence detection (present or absent), it also opens the door for future improvements. For example, adding face recognition could help identify individuals automatically, and analysing activity over time could provide deeper insights into work patterns.

In the long run, this project serves as a strong foundation for building more advanced, AI-driven workplace management systems that are both efficient and scalable.

9. REFERENCES

This project is built on a foundation of recent research in Computer Vision, Deep Learning, and office automation. Below are the key studies and papers that informed our methodology:

1. **Ahmad, S., & Sharma, R. (2023).** Looked at how lightweight YOLO models perform when used for monitoring workplaces at the "edge" (local processing).
2. **Alom, M. Z., et al. (2024).** Provided the mathematical groundwork for using point-in-polygon logic to track attendance accurately.
3. **Armenteros-Cosme, P., et al. (2025).** Focused on squeezing every bit of speed out of YOLOv4-tiny for low-power devices.
4. **Asmara, R. (2023).** Discussed why big companies are moving away from fingerprint scanners and toward camera-based systems.
5. **Berahmand, M., et al. (2024).** Explored new ways to optimize how we track indoor positioning in high-tech offices.
6. **Choi, M., & Lee, K. (2024).** Examined the social and technical impact of having cameras constantly running in professional settings.
7. **Dahal, B., et al. (2025).** Demonstrated how to combine YOLO with tracking algorithms like

Deep SORT for safety and presence.

8. **Fan, C. (2023).** Showed how to use data clustering to make sense of the massive amount of telemetry data corporate offices generate.
9. **Gupta, R., & Patel, D. (2025).** Tested how well OpenCV pipelines run on standard ARM processors.
10. **Islam, M., et al. (2026).** Used machine learning to move beyond simple attendance and into behavioural profiling.
11. **Jadhav, A., et al. (2024).** Experimented with combining GPS and Computer Vision for a "hybrid" tracking approach.
12. **Kumar, P., et al. (2024).** Laid out the framework for using OpenCV and Flask together for real-time office surveillance.
13. **Nguyen, T., & Hassan, M. (2025).** Made the case for local data processing as a way to protect employee privacy.
14. **Paneru, J., & Jeelani, S. (2024).** Studied how employees actually feel about location tracking tech over the long term.
15. **Patel, N., & Desai, M. (2026).** Highlighted the specific role YOLOv4-tiny plays in making "Smart Environments" a reality.
16. **Rahman, K., & Hossain, T. (2026).** Explored non-intrusive ways to detect presence in collaborative, open-office layouts.
17. **Rai, H., & Khatana, V. (2025).** A deep dive into building object recognition systems specifically using the OpenCV DNN module.
18. **Sultan, M., et al. (2024).** Compared the accuracy of biometrics vs. vision systems in the new world of hybrid work.
19. **Soltanikazemi, E., et al. (2023).** Focused on real-time tracking and detecting protocol violations on industrial sites.
20. **Wang, S., et al. (2025).** Analyzed lightweight AI architectures that keep latency low during human detection.
21. **Xu, J., & Vinciguerra, A. (2024).** Looked at unsupervised learning as a way to handle complex workforce data.
22. **Zhauniarovich, Y., et al. (2024).** Focused on the security and privacy "must-haves" for modern surveillance ecosystems.
23. **Zhang, L., & Zhao, L. (2025).** Compared different types of AI detectors to see which works best for automating office tasks.
24. **Syazwana, M., et al. (2023).** Tested YOLO-based systems for managing occupancy in both retail and office spaces.