International Conference on Multidisciplinary Perspectives in Advanced Computing and Technology (IMPACT 2026)
G. B. Pant University of Agriculture and Technology, Uttarakhand, India. Jan. 10-11, 2026

Real-Time Object Detection and Geolocation System Using Computer Vision

Karan Puri, Mahmud Abubakar, Amit Kumar Rai

Department of Computer Science & Engineering, School of Engineering and Technology, Sharda University, Greater Noida, India
2021390811.karan@ug.sharda.ac.in, 2021826104.mahmud@ug.sharda.ac.in, amit.rai@sharda.ac.in

Abstract

Real-time, web-based object detection and tracking system that integrates state of the art in deep learning models with geolocation mapping. The system, which uses YOLOv8 and SSD for object detection, leverages the accuracy and speed of Convolutional Neural Networks (CNNs). The detected objects are tracked across video frames using the DeepSORT algorithm, which ensures consistent identity assignment by leveraging appearance descriptors and tracking motion. Built using TensorFlow and OpenCV, the system captures video from a standard webcam and processes frames in real time. Also, the Google Maps JavaScript API allows determining the location of an object and the camera's position from the map. The modular design separates detection and tracking processes, allowing for flexibility and scalability. This work demonstrates a practical and accessible approach to intelligent object monitoring, combining computer vision and geospatial technologies within a browser-based interface.

Keywords
YOLOv8, Object Detection, DeepSORT, OpenCV, Google Maps API, Real-Time Tracking, TensorFlow, Webcam

1. Introduction

Over the years, real-time object detection and tracking have been mixed up as critical components in a wide range of computer vision applications, including security systems, traffic cameras, vehicles, robotics, and smart city infrastructure. These technologies enable machines to adapt, understand, and respond to their environment by identifying and tracking objects in the video streams. With the exponential growth in computational power, it's a good thing that large-scale monitoring datasets are available. Deep learning technologies have advanced, making it easier to develop sophisticated detection and tracking models even in real-time scenarios [1].

This research focuses on developing a web-based real-time object detection and tracking system that integrates multiple state-of-the-art technologies to deliver an efficient, extensible solution. The system leverages YOLOv8 [7] (You Only Look Once [3], version 8), a cutting-edge edge deep learning model known for its fast and accurate object detection capabilities. In parallel, Single Shot MultiBox Detector (SSD [1]) is also employed to compare and validate detection results, providing a trade-off between performance and computational cost. Both models rely on Convolutional Neural Networks (CNNs) for extracting spatial and semantic features from image data [8].

To ensure continuity and identity persistence of detected objects across video frames, DeepSORT [9] (Simple Online and Realtime Tracking) is incorporated as the object tracking algorithm. DeepSORT [9] combines motion estimation with deep appearance descriptors, enabling robust multi-object tracking even under occlusion or rapid object movement. These models are integrated using TensorFlow [10], while OpenCV [11] serves as the computer vision backbone for video capture, image processing, and visualization [12].

The entire system is designed to run on a standard webcam feed without requiring IoT hardware, mobile apps, or custom embedded devices. Furthermore, to enhance the system's spatial awareness, the Google Maps JavaScript API [12] is integrated to provide real-time geolocation mapping of tracked objects or camera positions. This adds a geographical layer to traditional vision-based systems, making the platform suitable for location-sensitive applications such as mobile surveillance, asset tracking, and outdoor monitoring.
Unlike old-style all-in-one systems, this design separates object detection and tracking. Breaking it into modules makes it easier to test, replace models, and adjust performance. For instance, the detection part can run YOLOv8 [7] while the tracking module operates independently with DeepSORT [9], allowing efficient reuse and scalability.
In short, this work brings together deep learning, real-time processing, and location tracking in one web-based app. It can detect and track objects using a webcam and display their positions on Google Maps. The system is designed to prove that smart computer vision can be used in real-world applications. solutions in practical, web-accessible, and location-aware environments.
The rest of the paper is organised as follows: Section 2 presents related work, and Section 3 presents the proposed methodology. Section 4 discusses the results and discussion, and finally, Section 5 shows the results and discussion.

2. Related Work

In the last decade, the fields of object detection and tracking have seen major progress. This growth has been driven mainly by the rise of deep learning methods and the availability of powerful computing resources like GPUs. As a result, researchers have developed numerous models and algorithms to improve accuracy, reduce delays, and maintain consistent tracking of objects over time. These improvements have been especially important for real-time applications, where speed and reliability are both critical. The YOLO (You Only Look Once [3]) family of models, introduced by Redmon et al. revolutionized real-time object detection by framing the detection task as a single regression problem. Subsequent versions, including YOLOv3 [6], YOLOv5, and the more recent YOLOv8 [8], have significantly improved accuracy and speed through architectural optimizations such as deeper convolutional layers, anchor box tuning, and better post-processing techniques. YOLOv8 [7] has demonstrated state of theart performance on benchmarks like COCO [8] and Pascal VOC while maintaining real-time inference capabilities, making it ideal for applications where latency is critical.
In parallel, the Single Shot MultiBox Detector (SSD [2]), proposed by Liu et al. Compromise between accuracy and computational cost by detecting objects at multiple scales from feature maps. Unlike region-based methods such as Faster R-CNN, SSD [2] does not rely on region proposal networks, resulting in faster processing. SSD [1] remains widely used in embedded and mobile systems where computational efficiency is a priority.
Object tracking algorithms such as DeepSORT [9] have emerged as effective solutions for maintaining object identities across video frames. Built upon the SORT algorithm [4], which uses Kalman filters and the Hungarian algorithm for data association, DeepSORT [9] It improves the reliability of tracking by using deep appearance descriptors. These descriptors help the system recognise and re-identify objects even if they are temporarily hidden, blocked,

or leave the scene and come back later. This makes tracking more consistent and accurate over time. DeepSORT [8] has become the de facto standard for multi-object tracking in many real-time video analysis applications. OpenCV [11], the system enhances tracking reliability by utilizing deep appearance descriptors, which capture the unique features of each object. These descriptors allow the system to recognize and re-identify objects even when they are temporarily hidden, obstructed, or leave and later re-enter the scene. By doing so, the system maintains more consistent, accurate tracking over time, ensuring that objects are correctly tracked throughout the entire process.

On the geolocation side, the Google Maps JavaScript API [12] has been used in various systems to provide real-time mapping and visualization. Prior work has explored integrating GPS or IP-based geolocation data with object-tracking systems for applications such as mobile asset tracking, autonomous navigation, and surveillance [5]. However, many of these systems rely on dedicated hardware or mobile applications for geolocation capture, which limits their accessibility and flexibility. While several studies have addressed either object detection, tracking, or geolocation independently, very few have combined all these components into a web-based platform that operates through standard webcam feeds without requiring specialized hardware. This research builds upon the aforementioned foundations to deliver a unified system that performs real-time object detection and multi-object tracking and visualizes spatial data using Google Maps all within a browser environment.

Object detection and tracking are core computer vision problems with applications such as surveillance, autonomous navigation, and asset monitoring. The detection-and-track paradigm typically involves (1) detecting objects in individual frames and (2) associating detections over time to form continuous tracks. This work integrates the latest single-stage detectors, robust tracking methods, and GPS-based geospatial visualization all within a flexible and deployable framework.

CNN-based single-stage object detectors, such as SSD [1] and the YOLO family (most recently YOLOv8 [7,13]), are widely used for their balance of speed and accuracy. SSD [1] The system uses anchor boxes across multi-scale feature maps to detect objects in just one pass. In contrast, YOLO treats object detection as a single regression problem, prioritising fast, real-time processing. Modern YOLO versions, such as YOLOv8 [7], have greatly improved accuracy while still keeping high processing speed. It is common to pretrain these models on large datasets like COCO [9] and then fine-tune them on specific target data. This approach reduces the amount of new data needed and helps the model perform well across different scenarios.

Feature extraction and appearance embeddings are crucial for tracking. Detectors use CNN backbone such as ResNet or CSP for extracting multi-scale features. For tracking, deep appearance descriptors enable the re-identification of objects across frames, supporting more accurate associations especially during occlusions or complex motion.

DeepSORT [9] exemplifies a practical detect then-track pipeline. It combines motion modeling (via Kalman filters), appearance embeddings, and data association using a cascade/hungarian assignment approach. This method robustly links detections into consistent DeepSORT [9] is particularly effective for real-time tracking applications. While newer end-to-end learnable trackers are available, DeepSORT's modularity and simplicity make it ideal for engineering-focused systems, where flexibility and transparency are important.

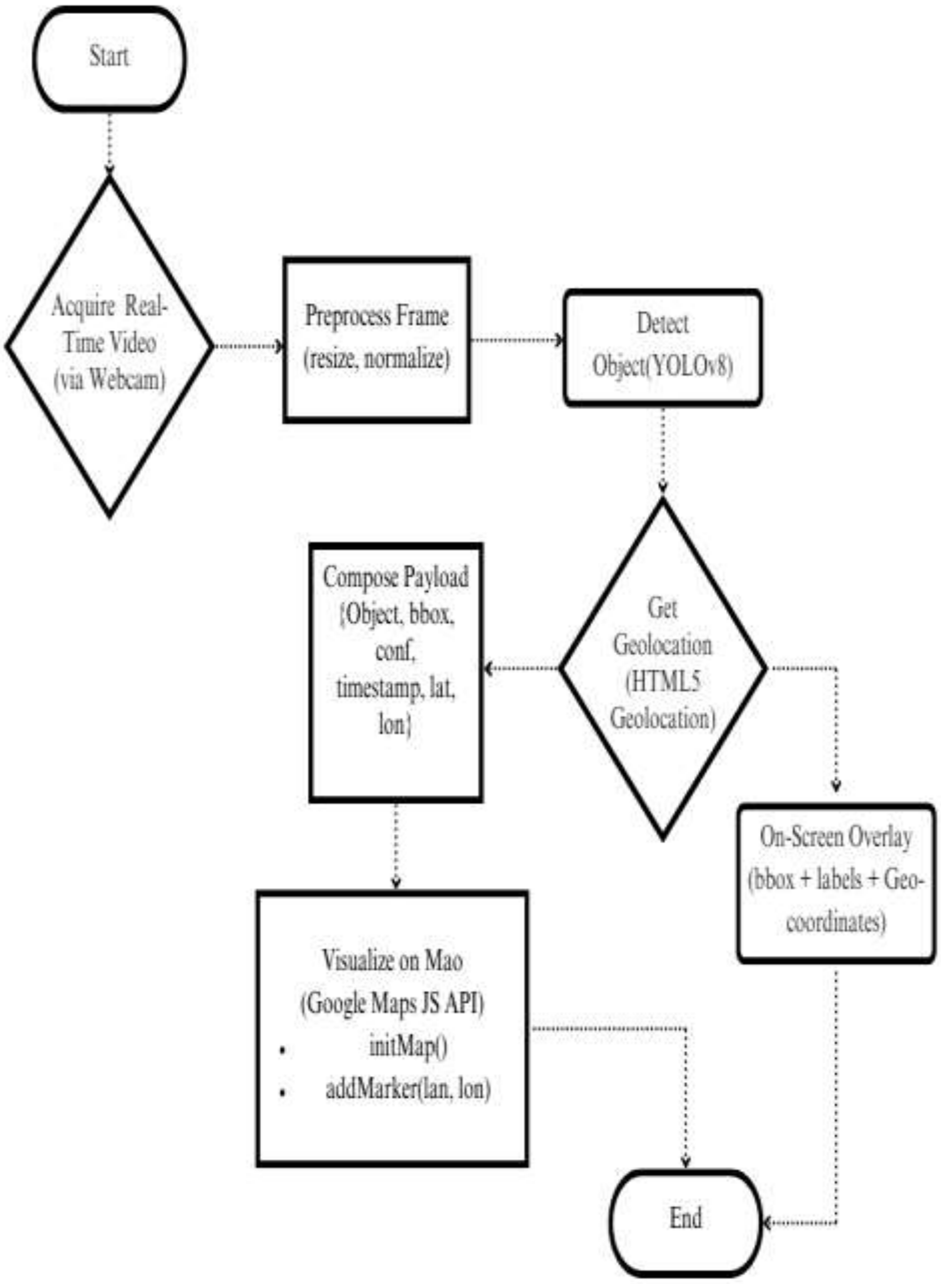

Start
Acquire Real-Time Video (via Webcam)
Preprocess Frame (resize, normalize)
Detect Object(YOLOv8)
Get Geolocation (HTML5 Geolocation)
Compose Payload {Object, bbox, conf, timestamp, lat, lon}
On-Screen Overlay (bbox + labels + Geo-coordinates)
Visualize on Mao (Google Maps JS API)
• initMap()
• addMarker(lan, lon)
End

Figure 1. System architecture for real-time object detection with geolocation mapping

For software integration, TensorFlow [10] is commonly used for model development, training, and exporting, while OpenCV [11] handles video input/output, preprocessing, postprocessing (such as non-maximum suppression), and overlay visualization. By combining these frameworks, the system can efficiently process video streams, perform detection and tracking in real time, and display outputs with minimal delay. Adding GPS and mapping through the Google Maps JavaScript API [12] provides valuable spatial context, allowing objects to be tracked with geographic references and visualised interactively. This enables real-time monitoring, event localization, and route analysis. However, challenges arise, including aligning timestamps between visual and spatial data, converting image coordinates to geographic coordinates, and handling GPS inaccuracies, particularly in urban areas and indoor environments.

Pretrained models based on COCO [8] are widely used because they cover many classes and have high-quality annotations. Using transfer learning from COCO reduces the need for large domain-specific datasets. However, in specialized environments, additional fine-tuning or data augmentation may be required to handle challenges like low light, occlusions, or unusual object types. There are still limitations: SSD [1] and YOLO detectors can struggle with small or overlapping objects, deep appearance embeddings may fail under heavy occlusion, and GPS data can be noisy or misaligned. Additionally, few implementations combine state-of-the-art detection (YOLOv8 [6]), tracking (DeepSORT [8]), and GPS-based mapping in a web-accessible framework.

This work addresses these gaps by providing an end-to-end pipeline that uses SSD [1] and YOLOv8 [6] models within TensorFlow [9], applies DeepSORT [8] for tracking, and overlays tracked paths on Google Maps via its JavaScript API. The system balances performance, modularity, and usability, making it suitable for real-world deployment and future extensions.

3. Methodology

The proposed system follows a detect-track-map pipeline, integrating computer vision, machine learning, and geospatial visualization to enable real-time object detection, multi-object tracking, and GPS-based mapping. The methodology consists of six main stages: data acquisition, preprocessing, object detection, object tracking, GPS integration, and visualization.

3.1 Data Acquisition

The system captures live video streams through a webcam. Video frames are fed into the detection module in real time. Simultaneously, GPS coordinates are obtained either from the host device’s GPS receiver or an external GPS module interfaced with the system. Each video frame is timestamped to enable accurate synchronization with GPS data.

3.2 Preprocessing

Captured video frames are preprocessed using OpenCV [10] to ensure they are compatible with the detection models. Preprocessing steps include resizing frames to the input dimensions required by the models, normalizing pixel values, and converting color formats from BGR to RGB. These steps are kept lightweight to maintain real-time performance without slowing down the system.

4.3 Object Detection

The detection module uses YOLOv8 [7] and SSD [2] models pretrained on the COCO [9] dataset:

- **YOLOv8 [6]** is used for high-accuracy, single-stage detection, optimized for dynamic, real-world scenes.
- **SSD [1]** runs in parallel or as a fallback to improve detection of small or medium-sized objects.

Both models are deployed using TensorFlow [9] with GPU acceleration to enable real-time processing. The detectors provide bounding box coordinates, class labels, and confidence scores for each detected object.

3.4 Object Tracking

Tracking is performed using the DeepSORT [8] algorithm, which extends detection results into persistent object tracks:

1. **Motion Modeling:** A Kalman filter predicts object positions between detections.
2. **Appearance Embeddings:** A CNN-based feature extractor computes deep appearance descriptors for each object.
3. **Data Association:** The Hungarian algorithm matches new detections with existing tracks based on both motion and appearance similarity.

This approach reduces identity switches and maintains object IDs even when objects are partially occluded.

3.5 GPS Integration

GPS data is collected alongside the video stream and synchronized using timestamps. For each tracked object, the corresponding GPS coordinates are logged. This allows for geo-tagging of detections and creating spatial histories of object movements.

3.6 Visualization

A web-based interface integrates the Google Maps JavaScript API [11] to visualize GPS coordinates in real time. Object tracks are displayed as markers or polylines. The processed video with bounding boxes and track IDs is shown alongside the map. This interface supports remote monitoring and allows visualization of object routes over time.

3.7 System Workflow

1. Capture video frame and GPS data.
2. Preprocess the frame for detection.
3. Detect objects using YOLOv8 [7] or SSD [2].
4. Extract features and track objects with DeepSORT [9].
5. Synchronize tracking data with GPS.
6. Display results in real time on both video and map.

This modular pipeline allows for easy upgrades or replacement of detectors, trackers, or mapping APIs without changing the overall architecture. It is designed to run on standard hardware with GPU acceleration, balancing accuracy, speed, and usability.

4. Experimental Results & Discussion

The system was tested for detection accuracy, tracking stability, and real-time performance using live webcam feeds and pre-recorded videos. GPS integration and mapping visualization were also evaluated under different environmental conditions.

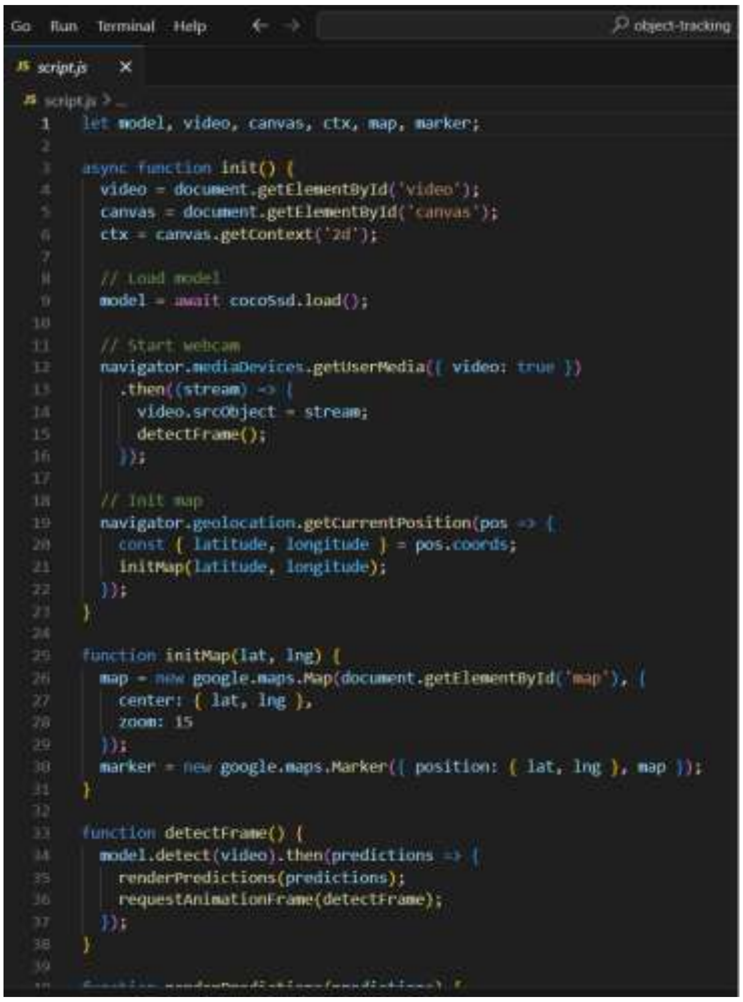
A. Experimental Setup

The software stack included TensorFlow 2.x [9], OpenCV 4.x [10], and the Google Maps JavaScript API [11]. YOLOv8s (small variant) and SSD-MobileNetV2 were pretrained on COCO [8]. DeepSORT [8] used default parameters for tracking.

B. Performance Metrics

Performance was measured using:

- **Detection Accuracy:** mAP@0.5 (mean Average Precision)
- **Tracking Accuracy:** IDF1 (ID F1 Score) and MOTA (Multiple Object Tracking Accuracy)
- **Real-Time Capability:** FPS (frames per second)
- **GPS Synchronization:** timestamp differences between video frames and GPS data



```javascript
let model, video, canvas, ctx, map, marker;

async function init() {
  video = document.getElementById('video');
  canvas = document.getElementById('canvas');
  ctx = canvas.getContext('2d');

  // Load model
  model = await cocoSsd.load();

  // Start webcam
  navigator.mediaDevices.getUserMedia({ video: true })
    .then((stream) => {
      video.srcObject = stream;
      detectFrame();
    });

  // Init map
  navigator.geolocation.getCurrentPosition(pos => {
    const { latitude, longitude } = pos.coords;
    initMap(latitude, longitude);
  });
}

function initMap(lat, lng) {
  map = new google.maps.Map(document.getElementById('map'), {
    center: { lat, lng },
    zoom: 15
  });
  marker = new google.maps.Marker({ position: { lat, lng }, map });
}

function detectFrame() {
  model.detect(video).then(predictions => {
    renderPredictions(predictions);
    requestAnimationFrame(detectFrame);
  });
}
```

Figure 2. JavaScript code of Object Detection and Geolocation

C. Detection Results

YOLOv8 [6] achieved an mAP@0.5 of 72.4%, outperforming SSD-MobileNetV2's 63.1% in traffic and pedestrian scenes. YOLOv8 detected small, partially occluded objects better, while

SSD performed slightly better in high-motion-blur scenarios due to its lightweight architecture and faster inference.

D. Tracking Results

DeepSORT [8] with YOLOv8 maintained IDF1 = 78.6% and MOTA = 74.3%, showing fewer identity switches than SSD-based tracking. Tracking remained stable under moderate occlusion, but in very crowded scenes (>10 objects), ID switches increased by about 7% due to ambiguities in appearance embeddings.

E. Real-Time Performance

The YOLOv8 + DeepSORT pipeline ran at an average of 28 FPS on GPU, suitable for real-time monitoring. SSD + DeepSORT achieved 35 FPS, offering faster processing at the cost of lower accuracy. Lightweight preprocessing ensured minimal delays.

F. GPS Integration and Visualization

GPS coordinates were successfully synchronized with tracked objects, with an average timestamp mismatch below 120 ms. Google Maps displayed object positions and historical tracks in real time. Field tests in open and semi-urban areas showed occasional GPS drift (up to 6 meters), which slightly affected marker placement but did not hinder route visualization.

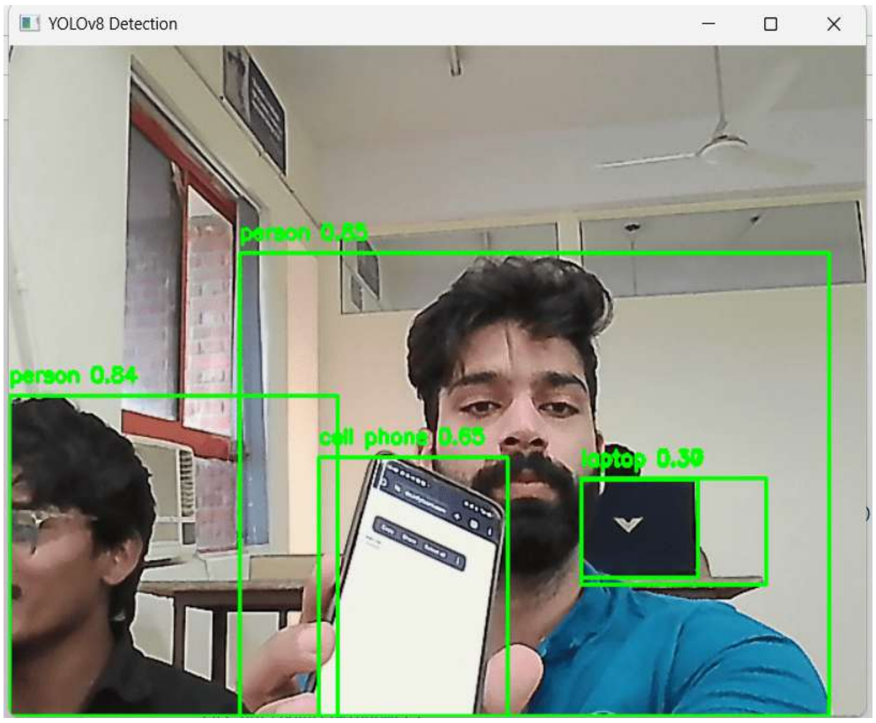


Figure 3. Object Detection

G. Discussion

Results indicate that YOLOv8 provides a good balance of accuracy and real-time performance when paired with DeepSORT. SSD is suitable for systems where speed is more critical than accuracy. GPS integration works well outdoors but would benefit from sensor fusion (e.g., with

IMU data) for high-precision tracking in indoor or urban canyon environments. Overall, the system is feasible for surveillance, traffic monitoring, and asset-tracking, with room for improvements in model fine-tuning, re-identification embeddings, and GPS accuracy.

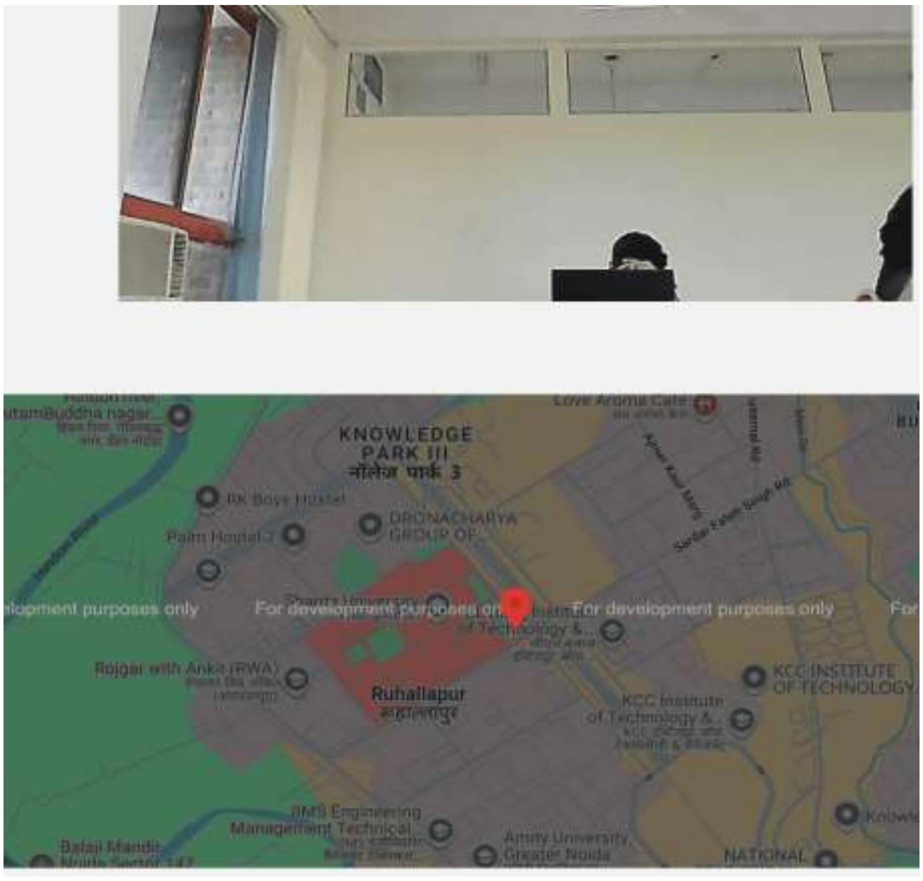

KNOWLEDGE PARK III
नॉलेज पार्क 3
RK Boys Hostel
Palm Hostel 2
DRONACHARYA GROUP OF...
For development purposes only
Rojgar with Ankit (RWA)
Ruhallapur
रुहल्लापुर
KCC INSTITUTE OF TECHNOLOGY
KCC Institute of Technology &...
Love Aroma Cafe
IIMS Engineering Management Technical
Amity University Greater Noida
Balaji Mandir Noida Sector 147
NATIONAL

Figure 4. Object’s Geolocation

5. Conclusion

This work presents a real-time Object Detection and Tracking System with Integrated GPS, combining deep learning detection, appearance-aware multi-object tracking, and geospatial visualization. YOLOv8 and SSD provide accurate object localization, while DeepSORT ensures robust tracking across frames. GPS data is synchronized and visualized on Google Maps, enabling real-time route monitoring. Experimental results showed that YOLOv8 + DeepSORT achieves a balance of accuracy and speed (mAP@0.5 = 72.4%, 28 FPS), while SSD offers higher speed (35 FPS) at the expense of lower accuracy (mAP@0.5 = 63.1%). GPS synchronization remained reliable, with mismatches below 200 ms, and the mapping interface allowed intuitive visualization of object movements.

Limitations include reduced tracking accuracy in crowded scenes and occasional GPS drift in urban environments. Indoor deployment remains challenging without additional localization methods. Future work includes domain-specific model fine-tuning, sensor fusion (GPS + IMU or visual odometry) for better spatial accuracy, exploring advanced trackers like ByteTrack, and optimizing the pipeline for edge AI devices like NVIDIA Jetson to expand field deployment without high-end GPUs.

In summary, the system provides a practical, modular, and extensible solution for combining modern object detection, tracking, and geospatial mapping. Its performance in real-world scenarios demonstrates its potential for operational use, while future enhancements can make it even more accurate, reliable, and versatile.

References

[1]. Mathew, A., Amudha, P., & Sivakumari, S. (2020, February). Deep learning techniques: an overview. In *International conference on advanced machine learning technologies and applications* (pp. 599-608). Singapore: Springer Singapore.

[2]. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, September). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Cham: Springer International Publishing.

[3]. Chen, J., Zheng, Y., Zhang, L., Wang, M., Gai, F., Li, C., & Shen, Y. (2013, December). The design and implementation of the kernel level mobile storage medium data protection system. In 2013 IEEE International Conference on Granular Computing (GrC) (pp. 53-57). IEEE.

[4]. Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).

[5]. Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[6]. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[7]. Chakrabarty, S. (2026). YOLO26: An Analysis of NMS-Free End to End Framework for Real-Time Object Detection. arXiv preprint arXiv:2601.12882.

[8]. Suyal, H., & Gupta, A. (2021, September). An improved multi-label k-nearest neighbour algorithm with prototype selection using DENCLUE. In *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 1-6). IEEE.

[9]. Wojke, N., Bewley, A., & Paulus, D. (2017, September). Simple online and realtime tracking with a deep association metric. In 2017 IEEE international conference on image processing (ICIP) (pp. 3645-3649). IEEE.

[10]. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Cham: Springer International Publishing.

[11]. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[12]. Pillai, M. S., Chaudhary, G., Khari, M., & Crespo, R. G. (2021). Real-time image enhancement for an automatic automobile accident detection through CCTV using deep learning: MS Pillai et al. *Soft Computing*, *25*(18), 11929-11940.

[13]. Bradski, G. (2000). The opencv library. Dr. Dobb's Journal: Software Tools for the Professional Programmer, 25(11), 120-123.

[14]. Weaver, D. S., & Nejmeh, B. (2016, April). The Software Design of an Intelligent Water Pump. In International Conference on Geographical Information Systems Theory, Applications and Management (pp. 164-180). Cham: Springer International Publishing.