Object Tracking Algorithms: From Conventional Methods to Deep Learning Approaches

Kirteyman Singh Rajput*, Kritika Koushal*, Vikram Singh*, and Suchitra Pandey*
*Department of Electronics & Telecommunication, Bhilai Institute Of Technology, Durg, India

***Abstract*—Visual tracking remains a challenging task in computer vision due to factors such as target distortions, varying lighting conditions, scale changes, rapid and sudden movements, partial occlusions, motion blur, object deformation, and background clutter. Traditional tracking algorithms, while effective in certain scenarios, often suffer from high computational complexity, sensitivity to noise, and unreliability when objects move beyond a limited neighborhood region—resulting in lost tracks, especially during fast motion or significant occlusions. Recent advances in object detection have enabled the development of robust tracking-by-detection methods that address many of these challenges. This project focuses on detecting and tracking objects—specifically vehicles and pedestrians—in video sequences using a tracking-by-detection framework. The approach involves detecting objects in each frame, assigning unique IDs to initial detections, and maintaining these identities as objects move across frames. This enables not only continuous tracking but also accurate counting of unique objects within a video. By combining the real-time detection capabilities of YOLO with the efficient and reliable tracking of Deep SORT, this work demonstrates a powerful and practical solution for multi-object tracking in dynamic environments, paving the way for further research and applications in surveillance, traffic monitoring, and autonomous systems.**

***Index Terms*—object tracking, YOLO, Deep Sort**

I. Introduction

Motion analysis and object tracking are among the most challenging tasks in digital video processing and computer vision, playing a pivotal role in applications such as autonomous vehicles, surveillance, sports analytics, and human-computer interaction. Object detection and classification serve as fundamental prerequisites for tracking, enabling the determination of an object's spatiotemporal trajectory as it moves across frames in a video sequence.

With the increasing demand for real-time multi-object tracking (MOT), modern systems must handle complex scenarios involving occlusion, illumination changes, and dynamic backgrounds . Recent advancements in deep learning-based object detection (e.g., YOLO, Faster R-CNN) have significantly improved tracking-by-detection frameworks, where object detections are linked across frames to maintain consistent identities.

Object tracking has a wide array of applications across various domains. Airspace monitoring utilizes radar tracking in military surveillance systems to identify aircraft, determine their characteristics, and assess potential intentions, although challenges arise from radar signal noise caused by environmental factors. Video surveillance is crucial for security in public and governmental spaces, driving research into intelligent object surveillance software. Weather monitoring relies on tracking weather balloons to gather high-altitude atmospheric data, with systems employing triangulation and radar to maintain continuous tracking. In cell biology, pathologists and medical researchers study cell behavior, using object-tracking algorithms to analyze cell division, birth, and death rates in studies of humans, insects, plants, and animals, particularly in immunology, to understand lymphocyte lifecycles.

The manuscript is organized as follows: Section I introduces the topic and objectives. Section II reviews conventional methods, while Section III outlines current challenges. Section IV describes the proposed methodology. Section V presents the results, followed by a discussion in Section VI. Finally, Section VII concludes the study and suggests future directions.

II. Conventional Methods

Object tracking in computer vision involves estimating the motion of objects across consecutive frames in a video sequence. Among the most widely used techniques are optical flow, Mean Shift/Cam Shift, Kalman Filter, and tracking-by-detection.

The optical flow defines the direction and time rate of pixels in a time sequence of two consequent frames. It provides a two-dimensional velocity vector, carrying evidence on the direction and the velocity of motion of each pixel in a given place of the frame [1]. It can be used to calculate the motion between two frames. The optical flow algorithm first takes a given set of points and the frame and then attempts to find those points in the next frame. And it's actually up to the user to supply the points to track. This method is based on object tracking using the Spatiotemporal image brightness variations at a pixel level to obtain a displacement vector for the object to be tracked across the frames. Tracking with optical flow rests on four important assumptions i.e., Brightness consistency (brightness around a small region is expected to remain nearly unchanged, though the location of the region might change), spatial coherence (neighboring points in the scene typically belong to the same surface and hence typically have similar motions), temporal persistence (motion of a patch has a gradual change), limited motion (points do not move

Corresponding author: M. Shell (email: kirteymanr@gmail.com).

very far or randomly). Once these conditions are satisfied, we use something called the Lucas-Kanade method to obtain an equation for the velocity of certain points to be tracked (usually, these are easily detected features) [2]. While the Lucas -Kanade calculates optical flow for a sparse feature set (meaning only the points it was told to track). The gunner farnerback's algorithm is used to calculate dense optical flow [3]. This dense optical flow calculates flow for all points in an image It will color them black if no flow is detected. The computational cost of sparse tracking is relatively low as compared to dense tracking [4] [5].

Mean shift is a non-parametric feature-space analysis method for finding the maxima of a density function [6]. The algorithm finds the nearest extremum and implements efficient tracking through the color probability distribution of objects. Mean-shift or Mode seeking is a well-known algorithm, which is mostly used in clustering and other related unsupervised problems. For the detection of an object in the frame, we extract certain features from the detection (color, texture, histogram, etc). By applying the mean-shift algorithm, we have a general idea of where the mode of the distribution of these features lies in the current state [7]. Now when we have the next frame, where this distribution has changed due to the movement of the object in the frame, the mean-shift algorithm looks for the new largest mode and hence tracks the object [**?**]. Continuously Adaptive Mean Shift Tracking (CAM Shift) is based on a mean-shift tracking technique and was primarily proposed to track human faces in a user interface system. This technique has the benefit that it adjusts the search window adaptively, unlike mean-shift tracking. In the mean-shift technique, a search window is chosen that gives the antecedent location, type (Gaussian or uniform), shape (symmetric, rectangular, or rounded), and size of the object. The window's center of mass is calculated, and this, in turn, is converged with the window's center. These steps are continuously repeated until the window stops moving.

Kalman filter is a recursive estimator that predicts the state of a moving object based on previous states and current measurements, accounting for noise and uncertainty. The core idea of a Kalman filter is to use the available detections and previous predictions to arrive at a best guess of the current state while keeping the possibility of errors in the process. The Kalman filter works recursively, where we evaluate current readings, to predict the current state, then use these measurements and update the predictions [8]. Most engineering problems that comprise prediction in a temporal or time series sense consist of computer vision, guidance, navigation, stabilizing systems, or even economics. It requires a "Kalman Filter". The core concept of a Kalman filter is to use the available detections and previous predictions to arrive at the best guess of the current state while keeping the likelihood of errors in the process [**?**].

Several other tracking algorithms are available as part of the OpenCV library, including BOOSTING, TLD, MIL, Median Flow, and KCF trackers. Each offers trade-offs in terms of speed, robustness, and suitability for different object types and motion patterns.

Tracking-by-detection frameworks have become the de facto standard for MOT in recent years. This approach involves two main stages [9]:

- **Object Detection:** A deep learning-based detector (e.g., YOLO, Faster R-CNN) identifies objects of interest in each frame, outputting bounding boxes and class labels.
- **Data Association:** A tracking algorithm (e.g., Deep SORT, Hungarian algorithm) assigns unique IDs to detected objects and maintains their identities across frames, even in the presence of occlusions or missed detections.

This approach leverages sensor fusion (e.g., cameras, LiDAR, radar) to enhance robustness, particularly in autonomous driving and smart surveillance systems. The integration of powerful detectors like YOLO with robust trackers such as Deep SORT has significantly advanced state of the art, enabling real-time, high-accuracy tracking in complex environments. Deep SORT, for example, leverages appearance features extracted by deep neural networks to improve data association, reducing identity switches and fragmentation.

III. Current Challenges

Tracking an object on a straight road or a clear area might sound very easy. But hardly any real-world application is that straightforward and free of any challenges. Some common problems that might be encountered while using an object tracker.

1) **Occlusion:** The occlusion of objects in videos is one of the most common obstacles to seamless tracking of objects. The man in the background is detected, while the same man goes undetected in the succeeding frame. Now, the challenge for the tracker lies in identifying the same man when he is detected in a much later frame and relating his older track and features with his trajectory.
2) **Variations in viewpoints:** In real-life examples, the task would be to track an object across several cameras. This may be used in hi-tech stores where there is no cashier, and we are required to track a customer throughout the store. As a result of this, there will be significant variations in how we view the object in individual cameras. In such cases, the features used to track an object become very significant as we need to make sure they are invariant to the variations in views.
3) **Non-stationary camera:** When the camera used for tracking a certain object is also in motion with respect to the object, it can often lead to unintentional consequences. Many trackers take into account the features of an object to track them. Such a tracker might fail in situations where the object looks different because of the camera motion (appear larger or smaller). A robust tracker for this problem can be very helpful in vital applications like object-tracking drones or autonomous navigation.

4) **Annotating training data:** One of the most irritating things about building an object tracker is getting good training data for a particular set-up. Unlike building a dataset for an object detector, we can have randomly unrelated images. In object tracking, it's important to have video or frame sequences where each instance of the object is uniquely identified for each frame.

IV. Methodology

The experiment is structured into two main stages: object identification and object tracking. The entire implementation is developed in Python and executed on the Google Colab platform, leveraging its computational resources for efficient processing. The approach follows a tracking-by-detection framework, where object detection is performed first, followed by tracking the detected objects across video frames. The detection and tracking are carried out using the YOLOv3 [11] and Deep SORT [12] algorithms, respectively.

1. Object Detection Using YOLOv3
YOLO (You Only Look Once) is a state-of-the-art, real-time object detection system that uses a single deep convolutional neural network to simultaneously perform object localization and classification. The key features of YOLOv3 used in this project are:

- **Single Neural Network Architecture:** YOLOv3 processes the entire image in one pass, dividing it into a grid and predicting bounding boxes and class probabilities for each grid cell. This unified approach enables high-speed detection suitable for real-time applications.

- **Input and Output:** The input to YOLOv3 is raw image pixels from each video frame. The output consists of bounding boxes around detected objects, along with confidence scores and class IDs indicating the object category.

- **Pre-trained Weights:** The model uses pre-trained weights trained on the Microsoft COCO dataset, which contains over 80 object classes and a large number of annotated images. This choice ensures robust detection performance without the need for extensive custom training, especially given the limited availability of annotated data in this project.

- **Detection Process:** For each frame extracted from the input video, YOLOv3 generates a set of bounding boxes with associated class probabilities [13]. These detections serve as the input for the tracking stage.

2. Object Tracking Using Deep SORT
Deep SORT (Simple Online and Real-time Tracking with a Deep Association Metric) is an extension of the SORT algorithm that incorporates appearance information to improve tracking accuracy and reduce identity switches. The tracking process involves the following components:

- **Kalman Filter for State Estimation:** The Kalman filter is used to predict the future state of each tracked object based on its previous state and motion model. The state vector for each object includes eight variables:
  The Kalman filter predicts the next position of the bounding box, smoothing the trajectory and handling occlusions or missed detections.

- **Hungarian Algorithm for Data Association:** To associate detected bounding boxes with existing tracks, Deep SORT uses the Hungarian algorithm to solve the assignment problem optimally. It matches detections to predicted tracks based on a cost matrix that combines motion (from the Kalman filter) and appearance similarity.

- **Appearance Descriptor:** Deep SORT uses a deep neural network to extract appearance features from detected objects, which helps distinguish between similar objects and maintain consistent identities over time.

- **Track Management:** The algorithm manages the track lifecycle by creating new tracks for unmatched detections and deleting tracks that have not been matched for a predefined number of frames. This ensures robust handling of objects entering and leaving the scene.

Integration of Detection and Tracking
The overall pipeline integrates YOLOv3 and Deep SORT as follows:

- **Video Input and Frame Extraction:** The input video is loaded, and frames are extracted sequentially.

- **Object Detection:** Each frame is passed through the YOLOv3 detector, which outputs bounding boxes and class IDs for detected objects.

- **Tracking:** The detections from YOLOv3 are fed into the Deep SORT tracker. The Kalman filter predicts the current state of each track, and the Hungarian algorithm associates detections to these tracks based on motion and appearance.

- **Output:** The tracker assigns unique IDs to each object and updates their trajectories across frames. The final output includes bounding boxes with class labels and consistent track IDs, enabling visualization and analysis of object movement.

The implementation leverages pre-trained models, software libraries, and a robust computational environment to ensure efficient and accurate object detection and tracking. Pre-trained Models for YOLOv3 and Deep SORT are utilized, with weights sourced from their respective repositories. These weights, trained on the Microsoft COCO dataset, provide a strong foundation for generalizing detection and tracking performance due to the dataset's diversity and scale.

The project is developed in Python, utilizing key libraries such as OpenCV for video processing, TensorFlow or PyTorch for deep learning model integration, and NumPy for numerical computations. These tools streamline the development process and ensure compatibility with modern machine-learning workflows.

To accelerate computations, the implementation is run on Google Colab, which offers GPU acceleration. This significantly enhances the speed of deep learning inference and tracking operations, making it suitable for handling real-time object-tracking tasks efficiently.

V. Results

The implementation of YOLOv3 and Deep SORT for object detection and tracking demonstrates notable strengths and limitations, highlighting areas for improvement and optimization. YOLOv3 proved to be an effective choice for object

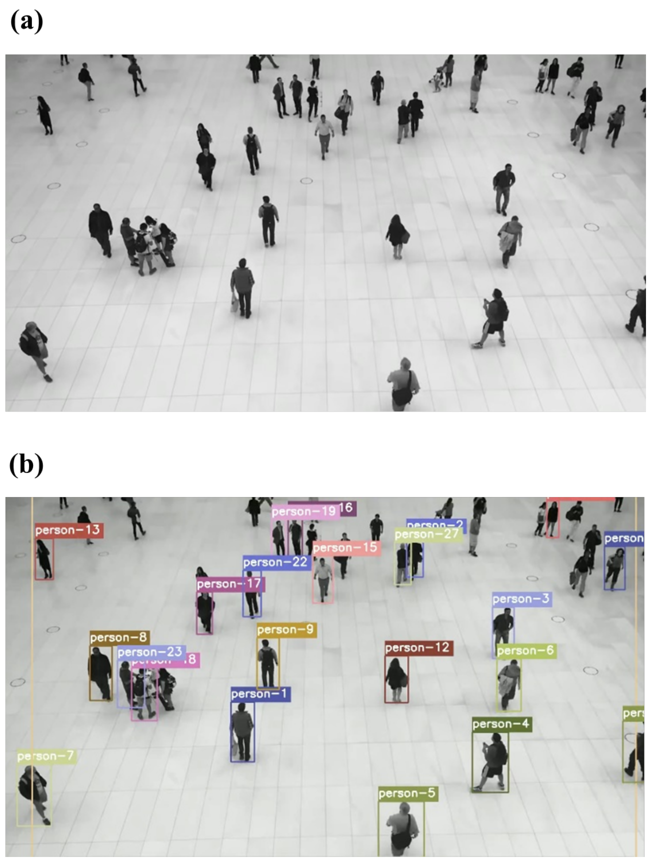

Fig. 1. Fig. 1: (a) Input video frame before object tracking processing and (b) Output video frame after object tracking processing.

detection due to its high speed and viable detection performance. The algorithm efficiently processes video frames in real-time, delivering bounding boxes and class IDs for detected objects. However, the results revealed a trade-off between speed and accuracy, with certain inaccuracies in detection adversely impacting the tracking process. These inaccuracies, such as false positives or missed detections, can propagate errors into the tracking stage, particularly in scenarios with dense or complex backgrounds.

Deep SORT performed well in maintaining track identities across frames, thanks to its integration of motion (via the Kalman filter) and appearance descriptors. The tracker exhibited minimal ID modifications during sequences with clear visibility. However, occlusion handling emerged as a significant limitation. In test sequences where objects were temporarily occluded, Deep SORT struggled to maintain consistent track identities due to the lack of robust association between feature vectors before and after occlusion. This limitation resulted in identity switches or lost tracks, particularly in crowded or dynamic environments.

VI. Discussion

1. Trade-off Between Speed and Accuracy
While YOLOv3 is one of the fastest object detectors available, its reliance on general-purpose pre-trained weights introduces accuracy limitations when applied to specific tasks. To address this gap, a custom-trained YOLOv3 model focused exclusively on objects of interest could enhance detection precision without compromising speed. By training on a domain-specific dataset, the model can reduce false positives and improve localization accuracy, thereby mitigating the adverse effects on tracking performance.

2. Improving Occlusion Handling in Deep SORT
The inability of Deep SORT to handle occlusions effectively stems from its reliance on feature vectors that may become inconsistent during occlusion events. Incorporating advanced appearance modeling techniques, such as deep feature extraction using temporal context or recurrent neural networks (RNNs), could improve robustness during occlusions. Additionally, implementing a confidence-based correction strategy could help reassign identities when objects reappear after occlusion, reducing identity switches.

3. Optimizing Confidence Thresholds
The choice of confidence threshold significantly impacts system performance. A larger threshold can filter out low-confidence detections that are likely incorrect but may also ignore valid detections with moderate confidence scores. A dynamic confidence thresholding mechanism that adjusts based on scene complexity or detection history could strike a balance between filtering false positives and retaining valid detections.

4. Alternative Detection Algorithms
While YOLOv3 offers a balanced trade-off between speed and accuracy, newer models such as YOLOv5 or EfficientDet provide improved accuracy with comparable runtime efficiency. Exploring these models could further enhance detection per-

formance without sacrificing speed, especially for applications requiring higher precision in dense environments.

5. Dataset Considerations

The use of pre-trained weights from the Microsoft COCO dataset provided a strong baseline but limited adaptability to specific scenarios in this experiment. Expanding the dataset with annotated examples from the target domain (e.g., surveillance videos or specific object categories) would enable fine-tuning of both YOLOv3 and Deep SORT models for improved performance under real-world conditions.

VII. Conclusion

The combination of YOLOv3 and Deep SORT offers a solid foundation for real-time object detection and tracking systems but leaves room for improvement in terms of accuracy and robustness under challenging conditions like occlusions or dense scenes. Customizing detection models and enhancing tracker algorithms can significantly refine system performance while addressing inherent limitations such as missed detections or identity switches. Future work should focus on integrating advanced techniques like domain-specific training, adaptive thresholds, and improved appearance modeling to create a more reliable and efficient tracking pipeline suitable for practical applications like video surveillance or autonomous systems.

References

[1] J. L. Barron, D. J. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, pp. 43–77, 1994. doi: 10.1007/bf01420984.

[2] G. Zhang and H. Chanson, "Application of local optical flow methods to high-velocity free-surface flows: Validation and application to stepped chutes," *Experimental Thermal and Fluid Science*, vol. 90, pp. 186–199, 2018. doi: 10.1016/j.expthermflusci.2017.09.010.

[3] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian Conference on Image Analysis*, 2003, pp. 363–370.

[4] S. Aslani and H. Mahdavi Nasab, "Optical flow based moving object detection and tracking for traffic surveillance," *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, vol. 7, no. 9, 2013.

[5] K. Kale, S. Pawar, and P. Dhulekar, "Moving object tracking using optical flow and motion vector estimation," in *4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)*, 2015, pp. 1–6.

[6] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, Aug. 1995. doi: 10.1109/34.400568.

[7] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002. doi: 10.1109/34.1000236.

[8] X. Li et al., "A multiple object tracking method using Kalman filter," in *2010 IEEE International Conference on Information and Automation*, 2010, pp. 1862–1866.

[9] K. S. Chahal and K. Dey, "A survey of modern object detection literature using deep learning," *arXiv preprint arXiv:1808.07256*, 2018. [Online]. Available: https://arxiv.org/abs/1808.07256.

[10] J. Zhao, Q. W. Men, and G. Z. Zhang, "An approach based on mean shift and Kalman filter for target tracking under occlusion," in *2009 International Conference on Machine Learning and Cybernetics*, 2009, vol. 4, pp. 2058–2062. doi: 10.1109/ICMLC.2009.5212129.

[11] J. Redmon et al., "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[12] A. Bewley et al., "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing*, 2016, pp. 3464–3468.

[13] S. Kapania et al., "Multi-object tracking with UAVs using deep SORT and YOLOv3 RetinaNet detection framework," in *Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems*, 2020, pp. 1–6.