Multi-camera multi-person tracking with DeepSORT and MySQL

Shashank Horakodige Raghavendra[1], Yashasvi Sorapalli[1], Nehashri Poojar S. V.[1], Hrithik Maddirala[1], Ramakanth Kumar P.[1], Azra Nasreen[1], Neeta Trivedi[2], Ashish Agarwal[2], Sreelakshmi K.[1]
[1]Department of Computer Science and Engineering, RV College of Engineering, Bangalore, India
[2]Inferigence Quotient Private Limited, Bangalore, India

Article Info

Article history:

Received Apr 21, 2024
Revised Oct 31, 2024
Accepted Nov 11, 2024

Keywords:

DeepSORT
Kalman filter
Object Occluded
Occluding
Occlusion handling
Person Occluded
YOLOv4

ABSTRACT

Multi-camera multi-object tracking refers to the process of simultaneously tracking numerous objects using a network of connected cameras. Constructing an accurate depiction of an object's movements requires the analysis of video data from many camera feeds, detection of items of interest, and their association across various camera perspectives. The objective is to accurately estimate the trajectories of the objects as they navigate through a monitored area. It has several uses, including surveillance, robotics, self-driving cars, and augmented reality. The current version of an object tracking algorithm, DeepSORT, doesn't account for errors caused by occlusion or implementation of multiple cameras. In this paper, DeepSORT has been extended by introducing new states to improve the tracking performance in scenarios where objects are occluded in the presence of multiple cameras. The communication of track information across multiple cameras is achieved with the help of a database. The suggested system performs better in situations where objects are occluded, whether due to object occlusions or person occlusions.

Corresponding Author:

Azra Nasreen
Department of Computer Science and Engineering, RV College of Engineering
Bangalore-59, India
Email: azranasreen@rvce.edu.in

1. INTRODUCTION

Recent years have seen significant advancements in computer vision, particularly in object detection and tracking, which are vital for applications such as surveillance, autonomous vehicles, video analytics, and human-computer interaction. In deep learning, tracking involves predicting the object positions across videos using spatial [1] and temporal [2] attributes. The detection module, employing techniques such as YOLOv4, locates objects, while the motion prediction module forecasts future trajectories based on past behaviour. Combining these modules enables the tracking system to follow and anticipate object movement, providing a comprehensive understanding of spatial and temporal dynamics [3].

The process comprises two stages: detection and tracking. YOLOv4 identifies objects in the detection stage, and DeepSORT [4], an extension of the SORT [5] algorithm, is used for tracking. DeepSORT integrates deep learning techniques, including Kalman filters for motion prediction and frame-by-frame data association using the Hungarian method [6]. An appearance descriptor, extracted through CNNs, minimises identity switches by encoding unique visual characteristics. This integrated approach ensures efficient tracking, accurate continuity, and a robust solution for multiple object tracking (MOT) tasks [7].

Despite these advancements, object tracking in multi-camera environments presents significant chal lenges. One major issue is occlusion [8], where objects may be partially or completely hidden from view, complicating the tracking process. Current tracking systems often struggle with maintaining object identities and continuity in the presence of occlusions or when objects move across overlapping camera views. This challenge is further compounded in multi-camera setups where synchronisation and data integration are crucial for accurate tracking. Various extensions and adjustments have been proposed in recent research to enhance tracking accuracy; however, many of these techniques are still unable to handle complicated real-world situations where cameras have overlapping fields of vision or objects are constantly obscured. Addressing these problems requires innovative solutions to enhance the robustness of tracking algorithms, particularly in complex scenarios.

This study proposes a novel strategy to address extended occlusion, reduce identity switches, and introduce a database system for seamless track information transfer between cameras. This approach maintains a unique ID for a person across cameras even in challenging scenarios. By combining these methodologies, this study contributes to advancing tracking technology and enhancing performance in complex scenarios.

The paper is structured as follows: section 2, DeepSORT enhancement, discusses improvements to the DeepSORT algorithm [9], focusing on occlusion handling, overlap management between camera views, and identity preservation. Section 3, experimental setup, details the equipment and tools used, along with the multi-camera video feed specifications. Section 4, data selection, explains the criteria for data inclusion and provides an overview of single and multi-camera footage scenarios. Section 5, modifications, outlines enhancements made to DeepSORT, including new variables, features, and the database setup for multi-camera tracking. Section 6, implementation, describes single and multi-camera tracking procedures, emphasizing occlusion handling and database management. Section 7, results and discussion, presents empirical results demonstrating the effectiveness of the proposed enhancements, discussing their impact in single and multi-camera environments, along with system limitations. Finally, section 8, conclusion, summarizes key contributions and suggests directions for future research.

2. DEEPSORT ENHANCEMENT

The DeepSORT code was modified to reduce identity mismatches [10] in cases of occlusion by performing a variety of tasks such as stopping visual feature updates and the addition of new track states. These enhancements include the following:

2.1. Occlusion handling

In the context of occlusion handling [11] in a single-camera scenario, it is essential to address situations where one person is directly behind another person, that is, multiple people are within the same bounding box. This is done to prevent unwanted visual feature updates that can lead to confusion and identity exchanges among individuals. Instead, specific measures were implemented to ensure accurate tracking.

2.1.1. Partial and complete overlap detection

The intersection over union (IOU) metric [12] is frequently used to determine when partial or complete overlap takes place between two tracks [13]. The amount of overlap between two bounding boxes is measured by IOU. If the IOU value between two matched tracks is greater than the IOU threshold (0.001), then it is a partial overlap. If the IOU value between a matched and an unmatched track is greater than the IOU threshold, then it is a complete overlap.

2.1.2. Stopping appearance feature updates

In case of the two above conditions being true, the track.update() function is modified to only update the movement features of the particular track and not the appearance features [14].

2.1.3. Avoiding identity mismatches

The tracker reduces the possibility of inaccurate identity mismatches by forgoing appearance feature updates during occlusion. This is because when numerous people are present in the same bounding box, appearance features that serve to distinguish between individuals may become inaccurate or unclear. The tracker can preserve each person’s track and reduce the likelihood of identity exchanges by giving priority to maintaining the Kalman [15] movement attributes.

3. EXPERIMENTAL SETUP

To conduct our research, we had to first capture a series of single-camera footage and multi-camera footage. These videos were recorded at a significant height and distance to allow the camera feed to be able to capture a wide enough area. These videos were then downloaded into our laptops for processing via our code to determine the accuracy and consistency of the code. They also allowed us to capture unknown edge cases and help modify the code to better fit them. We had video feeds for single-camera and multi-camera situations. Both these cases were handled differently by implementing different methodologies.

3.1. Instruments used

Table 1 has the specifications of the camera used for single-camera video feed. These videos were captured from a height of 3 stories. Table 2 has the specifications of the camera used for multi-camera video feed. Two of such cameras were installed at a height of 4 stories with some overlap between the two recording fields.

Table 1. Samsung Galaxy M31s specifications

Specification	Details
Device	Samsung Galaxy M31s
Main camera	Triple:
	64 MP, f/1.8, 26mm (wide), 1/1.73”, 0.8µm, PDAF
	12 MP, f/2.2, 123°(ultrawide)5 MP, f/2.4, (macro)
	5 MP, f/2.4, (depth)
	64 MP, f/1.8, 26mm (wide), 1/1.73”, 0.8µm, PDAF
Features	LED flash, panorama, HDR
Video recording resolution	UHD 4K (3840×2160) @ 30fps

Table 2. CP PLUS weatherproof outdoor security camera specifications

Specification	Details
Model	CP PLUS weatherproof outdoor
Type	Wired bullet security camera
Resolution	2.4 MP
Lens	3.6 MM for wide angle
Recording	1080P full HD
Features	Digital wide dynamic range (D-WDR)

3.2. Measurement for multi-camera video feed

Two cameras (CAM1 and CAM2) have been set up pointing from a height towards a field. The feeds of both cameras have a resolution of 1920×1080. CAM1 captures the left area, whereas CAM2 captures the right. Both cameras have an overlapping region in the middle. A large part of the feed beside the overlapping region has been used as the ‘writing region’. Whenever a track enters this ‘writing region’, its features are written into the database, which are immediately read and stored by the other camera. The overlap line for CAM1 extends from coordinates (1600, 0) to (2000, 1440), and for CAM2, from (1200, 0) to (600, 1440). The writing reference line is defined for CAM1 as (1900, 0) to (1900, 1440) and for CAM2 as (700, 0) to (700, 1440), marking the boundaries of this region for both cameras.

4. DATA SELECTION

Before heading into the modifications made to the existing DeepSORT algorithm, we had to obtain some footage for both single camera and multi camera tracking purposes.

4.1. Data selection criteria

Below is the selection criteria we had in place while collecting footage for testing.

4.1.1. Acceptable

- The people in the video feed are occluded: this is one of the two main problems we tackled in our research.
- The people in the video feed walk/run at varying speeds and in different directions: these scenarios were required in order to test the accuracy of our modifications.
- Any number of people in the video feed: as long as the people in the feed are detectable throughout the video, this data is acceptable and within scope of our research.

4.1.2. Out of scope

- The main object of tracking is not a person: these modifications made to the DeepSORT algorithm have only been tested on people (object=person). As we could create as many scenarios for testing by moving in different directions at will, this decision was made.

– The people in the feed must be detectable: this research only focusses on improving the tracking ability of DeepSORT. Therefore, it is an inherent requirement that all subjects must be detectable. No modifications were made to YOLOv4, the detection algorithm used in this research.

4.2. Single-camera and multi-camera footage details

A total of seven scenarios for single-camera environments and ten scenarios for multi-camera environments were carefully designed, recorded, and used as input data for the DeepSORT tracking algorithm. Table 3 presents the detailed descriptions of the situations observed in the single-camera setups, along with the tracking outcomes using the original DeepSORT algorithm. Improvements made to these initial results, leveraging the enhanced version of DeepSORT, are discussed in subsequent sections.

The multi-camera footage, captured using two synchronized cameras, was designed to encompass a wide range of scenarios, reflecting real-world complexities such as occlusions, crossovers, and directional changes. These scenarios simulate conditions often encountered in surveillance and tracking applications, ensuring that both the robustness and adaptability of the tracking algorithm were thoroughly evaluated. A detailed account of the scenario descriptions, along with the results obtained using the enhanced DeepSORT algorithm, is also provided in later sections. This combination of single and multi-camera tests ensures comprehensive validation of the improvements made to DeepSORT, demonstrating its capability to handle diverse tracking challenges effectively.

Table 3. Single-camera footage and result details on original DeepSORT

Event No.	Situation description	Output
1	Two people walk in opposite directions and cross over	IDs 1 and 2 assigned; on crossover, ID 1 becomes 3, ID 2 becomes 1.
2	Two people walk towards each other, talk, and walk off.	IDs 1 and 2 switch briefly after occlusion; Person 1 gets new ID 3 post-interaction.
3	Two people walk off at acute angles, return, and cross over.	ID 2 disappears temporarily, but is correctly re-assigned later.
4	One person walks, another runs in opposite directions	During crossover, ID 2 disappears briefly and is re-identified.
5	Two people walk in opposite directions and turn right on crossover.	ID 1 disappears; Person 1 takes ID 2, Person 2 gets new ID 3.
6	Two people walk at an acute angle; one distant person remains stationary.	IDs 1 and 2 switch during crossover and switch back shortly after.
7	Two people walk perpendicularly and cross over.	ID 1 disappears briefly but is correctly re-identified.

5. MODIFICATIONS

The original DeepSORT algorithm, though accurate, could not account for occlusions. This proposed solution handles occlusions while also making the algorithm suitable for multi-camera tracking [16], [17].

5.1. Newly introduced variables and features: single camera tracking

In case of occlusion, the IDs given to the tracks were almost never maintained, in the original DeepSORT. To improve this, we have increased the number of track states being considered and added additional metadata for better understanding of tracks assigned/changed.

5.1.1. Additional track states

Four new additional states were added along with the available DeepSORT states of ‘Tentative’, ‘Confirmed’, and ‘Deleted’.

The new states are:

a. Inactive state: if a detection associated with a ‘Confirmed’ track has not been detected again for over ‘max age’ number of frames (default of 60), then its state changes to ‘Inactive’ or else it will stay as ‘Confirmed’. In case of the original alogrithm, such tracks would have been deleted. To implement this, a track is set to ‘Inactive’ if its time since update (refer ‘Database setup and additions to code’) is greater than max age. The inactive state should be treated like the confirmed state and be included in track matching.

b. Occluding state: if a track moves in front of another track, preventing the latter track from being detected, then the former track is set to ‘Occluding’. The value of overlap between the bounding box of the involved tracks is used as a threshold for setting a track to ‘Occluding’. If it is greater than zero for any of the unmatched tracks, the state is changed to ‘Occluding’. Once this value is zero again, the state is changed to ‘Confirmed’.

c. PersonOccluded state: if a track is hidden from the camera because of another track, then its state is set to 'PersonOccluded'. The value of overlap between this track and all other matched tracked is determined. If this value is greater than 0, then the track is unmatched with a 'PersonOccluded' state and stays the same until it is matched again. Once this track is no longer hidden and detected, its state is changed to 'Confirmed'. If this occlusion lasts for more than max age frames then this track is moved to 'Inactive'.
d. ObjectOccluded state: if a track is hidden from the camera by a static object, then its state is set to 'ObjectOccluded'. Again, the value of overlap between this track and all other matched tracked is determined. If this value is greater than 0, then the track is unmatched with a 'ObjectOccluded' state and stays the same until it is matched again. If this occlusion lasts for more than max age frames then this track is moved to 'Inactive'.

5.1.2. Other added features

In addition to the new variables, the updated DeepSORT algorithm has a couple new logics introduced. These inclusions are useful in maintaing the ID of the track during occlusion.

a. Occpairs: a track in the occluding state remains as such until it no longer overlaps with any tracks predicted box. Suppose, if track A occludes track B for a long enough time, track B's predicted box could move out of the overlap causing track A to regain its confirmed state. This leads to track A being updated with incorrect features as the two detections are still overlapping. To handle this, 'Occpairs' were introduced. The 'Occpairs' variable is a list of pairs in which each pair is of the form (OccludingID, OccludedID). It is used to maintain the track states during long periods of occlusion. It is used to assign 'Occluding' and 'PersonOccluded' states before the overlap check. Every time an occlusion occurs, their respective IDs are added to this list, and only when the track with the OccludedID is matched again, the pair is removed from the list.
b. Two alphabet generator: along with the track ID, each track has a unique 2-alphabet code generated by a 2-alphabet generation logic. Every time a new track is initialised, the generator code runs and assigns a lexical value to it. This along with the track id serves to distinguish between tracks.

5.2. Database setup and additions to code: multi-camera tracking

In order to extend the single camera tracking capability to multi camera tracking, it is necessary to retain information about the tracks and communicate this between the cameras. This requires a database to store the communication details.

5.2.1. Selection of database

Before choosing a database, an experiment was performed over a video containing 10,000 frames to determine the read/write times of two databases, MySQL and MongoDB. The read/write times mentioned in Table 4, are the times taken to read and write a single feature vector respectively from and to the database. Each feature vector consists of 128 values. Based on the read/write times obtained over the 10,000 frames, various statistical values including minimum, maximum, average, variance, and standard deviation were calculated.

Table 4. Performance metrics for 10,000 frames

Metrics	MySQL(W)	MySQL(R)	MongoDB(W)	MongoDB(R)
Min	1.1149 ms	0.3381 ms	approx. 0s	approx. 0s
Max	43.7874 ms	2.9908 ms	201.677 ms	35.336 ms
Mean	2.2410 ms	0.8264 ms	1.5361 ms	0.639 ms
Vars	0.0093 ms	0.0006 ms	0.0099 ms	0.0004 ms
Std. D	0.0963 ms	0.0246 ms	0.0998 ms	0.0198 ms

5.2.2. Schema

The structure or blueprint that specifies how data is arranged, kept, and accessed within a database is referred to as a database schema. It specifies the tables, fields, relationships, constraints, and other elements of the database's logical and physical layout. Table 5 represents the schema of the two feature tables and the shared table. It also provides the attributes of the tables.

In our MySQL database, three different tables are maintained:
feature Table 1: consists of the track features written by CAM1. The table is read by CAM2.
feature Table 2: consists of the track features written by CAM2. The table is read by CAM1.
shared table: used for allowing communication to take place between both CAM1 and CAM2.

Table 5. Database table attributes

Table name	Attribute	Data type	Details
Feature table	id	int32	Specifies the track ID.
	state	int32	Specifies the state of the track (Tentative=1 and Confirmed=2)
	features	Varchar (3000)	Appearance vector (Array of 128 values)
	frame num	int32	Frame number
	hits	int32	Number of times the track has undergone a measurement update
	mean	Varchar (3000)	8-dimensional mean vector
	covariance	Varchar (3000)	8×8-dimensional covariance matrix
	track str	Varchar (2)	Random 2-alphabet string assigned to the track
	age	int32	Total number of frames since the first occurrence of the track
	occ id	int32	The ID of the track being occluded by this trac
Shared table	id	int8	Set to 1. Used only for reading/writing purposes
	next id	Varchar (2)	Used to assign track IDs, incremented (+1) by either camera when a
	comm1	int8	Initialised to 0. Set to the number of people in the writing region by CAM1 during its write operation. Reset to 0 by CAM2 after reading every row in feature Table 1.
	comm2	int8	Initialised to 0. Set to the number of people in the writing region by CAM2 during its write operation. Reset to 0 by CAM1 after reading every row in feature Table 2.

6. IMPLEMENTATION

The DeepSORT algorithm, based on the SORT [18] architecture, has received significant recognition for its capacity to track objects in video sequences. Its ability to incorporate appearance features as well as motion data makes it ideal for complex tracking scenarios involving many cameras and occlusions. Previous research has shown that DeepSORT is useful in a variety of tracking applications. For example, research has shown that DeepSORT outperforms common tracking approaches by retaining identity consistency, regardless of difficult situations.

Our methodology's results are especially important in real-world applications like surveillance [19], [20], robotics [21], and security [22], [23], where continuous tracking across many camera feeds is essential. The enhancements provide practical benefits by increasing the reliability and precision of tracking systems in challenging scenarios.

6.1. Procedure of implementation for single-camera tracking

The procedure for single-camera tracking largely follows the original DeepSORT algorithm, with modifications introduced to improve accuracy, particularly during occlusions.

6.1.1. Initialisation

The necessary libraries and dependencies for DeepSORT and YOLO [24] (or any other object detector) are loaded at the start. The YOLO detector is initialized to detect objects in video frames. Additionally, the DeepSORT tracker is initialized with specific configurations, such as setting the max age parameter to 60 frames and configuring the IOU threshold for overlap detection (e.g., 0.001).

6.1.2. Tracking with occlusion handling

For every frame in the video stream, the YOLO detector is used to detect objects, providing bounding boxes and detection confidences. These detections are fed into the DeepSORT tracker. For each detected object, the IOU is calculated between the current detection and all active tracks. Based on the IOU and feature similarity, detections are assigned to tracks.

If a detection matches a track, the track's position is updated using Kalman filter predictions. When the IOU indicates partial or complete overlap with another track, the track state is updated to either 'Occluding' or 'PersonOccluded', and the pair of tracks is added to the 'Occpairs' list. At this point, appearance feature updates for the track are suspended to prevent identity mismatches. If a detection does not match any track, a new track is initialized, assigned a unique ID, and given a random two-alphabet code.

For each track, the system checks if the time since update exceeds max age. If it does, the track's state changes to 'Inactive' and is added to the pool of tracks available for future matching. If an 'Inactive' track finds a matching detection, its state is restored to 'Confirmed'. Tracks occluded by static objects for more than max age frames also switch to 'Inactive'.

6.1.3. Handling track states

Tracks in the 'Occluding' state revert to 'Confirmed' once they no longer overlap, and the pair is removed from the 'Occpairs' list. Similarly, for tracks in the 'personOccluded' state, if the occluding track moves and the occluded track is detected again, the state changes to 'Confirmed', and the pair is removed from the list. If occlusion persists beyond max age frames, the track becomes 'Inactive'. Tracks in the 'ObjectOccluded' state change to 'Confirmed' when detected again but transition to 'Inactive' if occlusion exceeds max age frames.

6.1.4. Output

The final output consists of video frames with overlaid bounding boxes and track Ids [25]. Optionally, detailed logs of track information and states can be generated for analysis.

6.2. Procedure of implementation for multi-camera tracking

To support multi-camera tracking, additional changes are implemented alongside the modified single-camera algorithm.

6.2.1. Initialisation

Libraries and dependencies for DeepSORT and YOLO (or other object detectors) are loaded. YOLO is initialized for each camera to detect objects in their respective frames. A separate DeepSORT tracker is initialized for each camera, with the max age parameter set to 60 frames and an IOU threshold of 0.001. A shared database environment is also established to facilitate communication between the cameras, and polygon regions are defined to avoid overlap.

6.2.2. Writing to database

In the track.update() function of each camera, the system checks if the track is non-tentative and has valid features. If the track lies within the specified polygon region, its features are written to the database's feature table. Additionally, the next ID is fetched from the shared table to maintain unique ID assignments across cameras. The comm1 and comm2 values are updated based on the number of tracks in the polygon region for each camera.

6.2.3. Reading from database

In the tracker.update() function, each camera uses the comm value to determine how many tracks to read from the feature table. The system retrieves tracks ordered by frame number (in descending order) to ensure the most recent data is accessed. For each retrieved track, if the track ID exists locally, its features, hits, and age are updated. If the track is new, it is added to the local track object list.

6.2.4. Communication between cameras

Communication between cameras is managed through the comm1 and comm2 values. If comm1 is greater than 0, camera 2 reads from camera 1's feature table, and vice versa if comm2 is greater than 0. In the track.update() function, data from other cameras is used to update or create new tracks as needed. The comm values are also updated to ensure smooth communication and data consistency.

6.2.5. Output

The video output from each camera displays bounding boxes and track IDs. Track information and states are logged for further analysis. This logging of track states allows for detailed examintaion of object movement and tracking accuracy over time. As a result, a comprehensive view of multi-camera tracking performance is provided.

7. RESULTS AND DISCUSSION

This study investigated the capabilities of an enhanced DeepSORT algorithm for multi-object tracking in dynamic environments. While earlier studies have explored various tracking methods, they have not explicitly addressed the influence of continuous identity maintenance during occlusions, a critical factor in real-world applications.

The method followed in this paper demonstrated the capability to accurately identify individuals and consistently assign them the same IDs, even after periods of occlusion. We found that the enhanced DeepSORT algorithm correlates with improved tracking accuracy, as evidenced by higher accuracy percentages across various video sequences. This improvement is largely attributed to the enhanced DeepSORT's ability to maintain original IDs of individuals during occlusions by suspending appearance updates, thereby reducing the likelihood of incorrect ID assignments. The use of MySQL to facilitate the communication of track features in order to recognize objects across multiple cameras is another key innovation of this study's approach. These inno- vations not only improve the fidelity of individual tracking in complex scenarios but also set a new benchmark for future research in the field of multi-camera multi-object tracking.

The proposed method uniquely represents each person's track state at every single frame, a feature that was not present in other related studies. This frame-by-frame representation allows for continuous and detailed tracking of individuals, providing a richer dataset and enabling more granular analysis. Unlike other methods that may skip frames or provide incomplete data, this approach ensures that no information is lost between frames, thereby enhancing the accuracy and completeness of the tracking system.

7.1. Single camera results

The enhanced DeepSORT algorithm consistently demonstrates superior accuracy compared to the original, as evidenced by higher accuracy percentages across all video sequences. This improvement is largely due to the enhanced DeepSORT's ability to maintain the original ID of individuals even after occlusions, providing more reliable tracking in complex scenarios. Figure 1 illustrates the comparative performance of the original and enhanced DeepSORT algorithms across the seven video sequences. The line graph shows accuracy percentages calculated based on the proportion of frames with correct ID assignments relative to the total number of frames in each single-camera video.

Figure 2 compares ID assignment by the original DeepSORT and the proposed enhanced DeepSORT before and after occlusion. Figures 2(a) and 2(b) show the original DeepSORT's performance before and after occlusion, while Figures 2(c) and 2(d) represent the enhanced DeepSORT's results. The correct ID assignment in Figures 2(c) and 2(d) is the result of stopping visual feature updates in cases of partial or complete overlap of tracks to preserve proper track features.

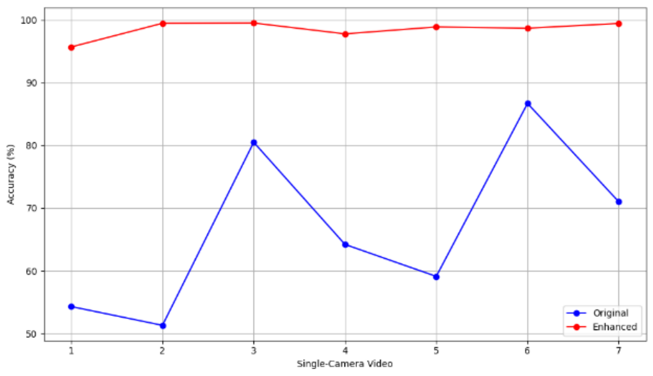
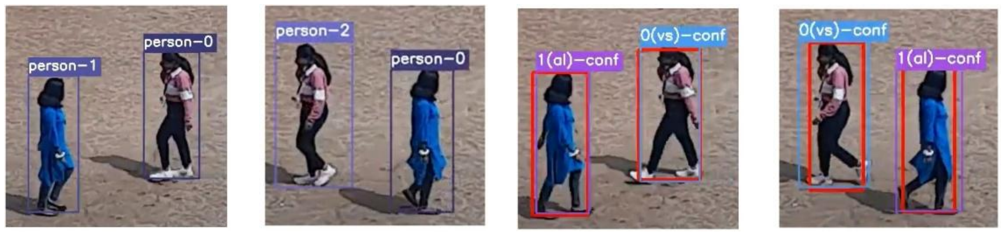


Figure 1. Accuracy comparison between original and enhanced DeepSORT for single camera video



(a) (b) (c) (d)

Figure 2. ID assignment using; (a) original DeepSORT before occlusion and (b) after occlusion, and using the (c) proposed enhanced DeepSORT before occlusion and (d) after occlusion

The transition conditions between two states of a track, obtained as a result of implementing the enhanced DeepSORT algorithm, are detailed in Table 6. The row headers represent the state of a track in the current frame, and the column headers represent the state of the track in the next frame. The cells represent the conditions necessary for a track to change from one state to another.

The parameters used in the DeepSORT algorithm include 'n init,' which specifies that a detection must be continuously detected for the first three consecutive frames before being eligible for track assignment. The 'max age' variable indicates the maximum number of frames a track can go undetected before being marked for deletion, set to 60 frames in this implementation; however, it transitions to an 'Inactive' state rather than a 'Deleted' state as in the original version. Additionally, 'time since update' tracks how long a detection has remained undetected.

Table 6. State matrix

Current Frame state name	Next frame						
	Tentative	Confirmed	Inactive	Deleted	Occluding	Person-Occluded	Object-Occluded
Tentative	-	Detected and tracked for n_init		Detected and tracked for less than n _init	-	-	-
Confirmed	-	-	Out of screen for max age		When it occludes another person	Occluded by another person	Occluded by a static object
Inactive	-	Back to screen after going out for greater than max_age		Deleted if time since update greater than 1000	If the track occludes someone as soon as it becomes active	-	-
Deleted	-	-	-	-	-	-	-
Occluding	No longer occluding another person	-	-	-	Swap between TrackIDs during person occlusion	Goes behind static object during person occlusion	-
PersonOccluded	No longer occluded by another person	Occluded by a person for more than max age	Swap between TrackIDs during person occlusion	-	-	Goes behind static object during person occlusion	-
ObjectOccluded	No longer occluded by a static object	Occluded by a static object for more than max age	-	-	Moves from behind a static object to behind a person	-	-

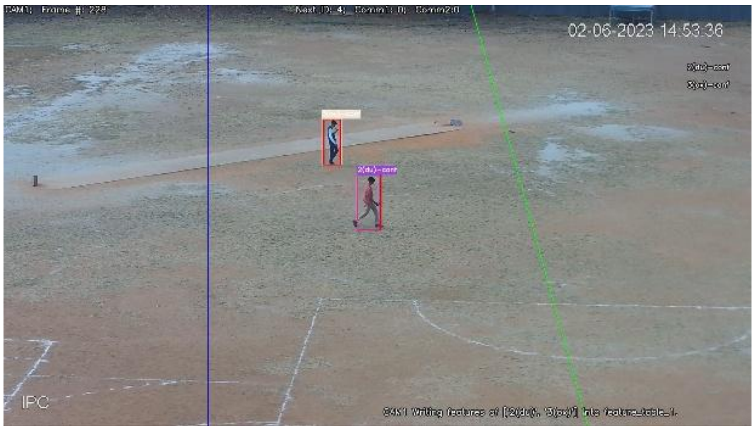
7.2. Multi-camera results

In this section, we present the results obtained from the multi-camera tracking system, emphasizing the situations recorded and their respective outputs. Table 7 details various scenarios captured by the cameras. The higher IDs (greater than 2) in the results can be attributed to missing detections at the beginning of the observation period, which leads to the creation and maintenance of these higher ID values.

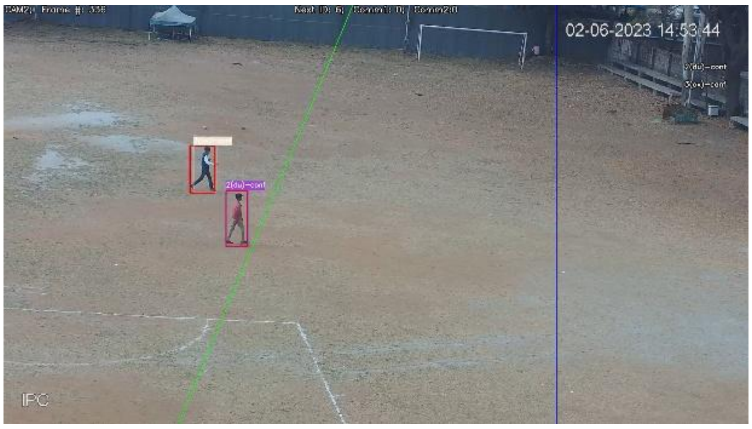
Figure 3 illustrate the outputs captured durng Event 5 (as described in Table 7). These Figures 3(a) and 3(b) showcase the real-time tracking results from both cameras, highlighting how Person 1 (with ID 3(ox)) and Person 2 (with ID 2(du)) are assigned and maintained their respective IDs while transitioning from CAM1 to CAM2 in a simultaneous horizontal movement.

Table 7. Multi-camera result details

Event No.	Situation description	Output
1	Person A walks horizontally from CAM1 to CAM2	CAM1: ID1(gj) maintained; CAM2: Same ID1(gj).
2	Person A walks horizontally from CAM2 to CAM1	CAM1: ID1(hv) maintained; CAM2: Same ID1(hv).
3	Person A walks diagonally from top-left CAM1 to bottom-right CAM2.	CAM1: ID4(wk) maintained; CAM2: Same ID4(wk).
4	Person A walks diagonally from bottom-right CAM2 to top-left CAM1.	CAM1: ID1(mz) maintained; CAM2: Same ID1(mz).
5	Persons A and B move horizontally from CAM1 to CAM2.	CAM1: IDs 3(ox), 2(du); CAM2: Same IDs.
6	Persons A and B move horizontally from CAM2 to CAM1	CAM2: IDs 1(pm), 2(cl); CAM1: Same IDs
7	Person A walks from CAM1 to CAM2; Person B follows after	CAM1: A gets ID1(pf); CAM2: B gets ID2(us) after A exits
8	Person A walks from CAM2 to CAM1; Person B follows after	CAM2: A gets ID1(oi); CAM1: B gets ID2(wy) after A exits
9	Persons A and B walk diagonally towards opposite corners	CAM1: A gets ID4(yr); CAM2: B gets ID1(yx)
10	Person A occludes Person B, both move to CAM1, then split	CAM2: IDs 1(oa), 2(ow). A occludes B, who goes inactive after 60 frames. CAM1: A new detection boxmerges them into ID8(lr). IDs 1(oa), 2(ow) restored after splitting.



(a)



(b)

Figure 3. Outputs for event 5 (described in Table 7); (a) from CAM1 and (b) CAM2 showing tracking results and ID assignment

7.3. Caveats and limitations

While the proposed method advances multi-object tracking, several limitations were encountered in the study, highlighting areas for potential refinement and future research:

- Incorrect assignment of ‘ObjectOccluded’: a person may be misclassified as ‘ObjectOccluded’ due to detection lapses or moving object occlusions, causing incorrect assumptions of static occlusion.
- Incorrect assignment of ‘Occluding’: the system determines occlusion using IOU based on bounding box overlap. However, it may label a person as ‘Occluding’ even when the object causing the overlap is hidden, resulting in inaccurate tracking statuses.
- Overlap region dilemma: when a person enters the overlap region between two cameras without prior detection, they might be assigned different IDs in each view due to the lack of database information, underscoring the need for better identity tracking across cameras.
- Not entering writing polygon: if a person exits camera 1 without entering the designated writing polygon, their features won’t be recorded. If they later appear in camera 2, they will receive a different ID, illustrating the system’s reliance on polygon regions for feature registration.

8. CONCLUSION

This study presents key enhancements to the DeepSORT algorithm, enabling the maintenance of individual IDs even after occlusions, a challenge in earlier implementations. The method was tested on both single- and multi-camera videos, achieving consistent identity tracking across cameras. A novel aspect of this research is the use of a MySQL database for sharing appearance features across cameras, marking the first attempt at such an approach. Four new states-Occluding, PersonOccluded, ObjectOccluded, and Inactive-were introduced, providing a more comprehensive framework for tracking and encouraging further research.

The findings have significant implications for improving tracking accuracy and robustness in complex environments, such as surveillance and sports analytics. By ensuring consistent IDs across occlusions and multiple cameras, this method supports scalable applications requiring reliable human tracking. The MySQL-based feature-sharing system offers new possibilities for networked tracking systems, with potential for further optimization. However, limitations were identified, including incorrect 'ObjectOccluded' and 'Occluding' status assignments due to detection errors and IOU miscalculations, as well as challenges like the overlap region dilemma and missed entry into the writing polygon.

This research addresses the core issue of identity consistency across occlusions and cameras while providing new tools and states for future studies. It lays the foundation for more accurate tracking algorithms through detailed state tracking and improved occlusion management. Future work can focus on optimizing the MySQL system for larger-scale applications, integrating the framework with advanced algorithms, and refining detection methods to minimize status errors. This study represents a significant step forward in multi-object tracking, offering practical solutions and opening new avenues for exploration.

REFERENCES

[1] S. Sakaguchi, M. Amagasaki, M. Kiyama, and T. Okamoto, "Multi-camera people tracking with spatio-temporal and group considerations," *IEEE Access*, vol. 12, pp. 36066–36073, 2024, doi: 10.1109/ACCESS.2024.3371860.

[2] J. Li and Y. Piao, "Multi-target multi-camera tracking based on mutual information-temporal weight aggregation person re-identification," in *2022 IEEE 5th International Conference on Electronic Information and Communication Technology (ICEICT)*, Aug. 2022, pp. 149–151, doi: 10.1109/ICEICT55736.2022.9908659.

[3] Y. Gao, W. Wu, A. Liu, Q. Liang, and J. Hu, "Multi-target multi-camera tracking with spatial-temporal network," in *2023 7th International Symposium on Computer Science and Intelligent Control (ISCSIC)*, Oct. 2023, pp. 196–200, doi: 10.1109/ISCSIC60498.2023.00048.

[4] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 3645–3649, doi: 10.1109/ICIP.2017.8296962.

[5] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 3464–3468, doi: 10.1109/ICIP.2016.7533003.

[6] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2018, pp. 748–756, doi: 10.1109/WACV.2018.00087.

[7] L. Zheng, M. Tang, Y. Chen, G. Zhu, J. Wang, and H. Lu, "Improving multiple object tracking with single object tracking," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 2453–2462, doi: 10.1109/CVPR46437.2021.00248.

[8] D. Stadler and J. Beyerer, "Improving multiple pedestrian tracking by track management and occlusion handling," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 10953–10962, doi: 10.1109/CVPR46437.2021.01081.

[9] T. L. Dang, G. T. Nguyen, and T. Cao, "Object tracking using improved deep SORT YOLOv3 architecture," *ICIC Express Letters*, vol. 14, no. 10, pp. 961–969, 2020, doi: 10.24507/icicel.14.10.961.

[10] M. I. H. Azhar, F. H. K. Zaman, N. M. Tahir, and H. Hashim, "People tracking system using DeepSORT," in *2020 10th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Aug. 2020, pp. 137–141, doi: 10.1109/ICCSCE50387.2020.9204956.

[11] A. Specker, D. Stadler, L. Florin, and J. Beyerer, "An occlusion-aware multi-target multi-camera tracking system," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2021, pp. 4168–4177, doi: 10.1109/CVPRW53098.2021.00471.

[12] H.-W. Huang *et al.*, "Enhancing multi-camera people tracking with anchor-guided clustering and spatio-temporal consistency ID Re-assignment," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2023, pp. 5239–5249. doi: 10.1109/CVPRW59228.2023.00552.

[13] Y. He, X. Wei, X. Hong, W. Shi, and Y. Gong, "Multi-target multi-camera tracking by tracklet-to-target assignment," *IEEE Transactions on Image Processing*, vol. 29, pp. 5191–5205, 2020, doi: 10.1109/TIP.2020.2980070.

[14] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 6036–6046, doi: 10.1109/CVPR.2018.00632.

[15] F. Yang, X. Zhang, and B. Liu, "Video object tracking based on YOLOv7 and DeepSORT," *Prepr. arXiv.2207.12202*, Jul. 2022, [Online]. Available: http://arxiv.org/abs/2207.12202

[16] D.-J. Huang, P.-Y. Chou, B.-Z. Xie, and C.-H. Lin, "Multi-target multi-camera pedestrian tracking system for non-overlapping cameras," in *2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, Jul. 2023, pp. 629–630, doi: 10.1109/ICCE-Taiwan58799.2023.10227006.

[17] A. Specker, L. Florin, M. Cormier, and J. Beyerer, "Improving multi-target multi-camera tracking by track refinement and completion," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2022, pp. 3198–3208. doi: 10.1109/CVPRW56347.2022.00361.

[18] D. M. Jiménez-Bravo, Á. L. Murciego, A. S. Mendes, H. S. San Blás, and J. Bajo, "Multi-object tracking in traffic environments: A systematic literature review," *Neurocomputing*, vol. 494, pp. 43–55, Jul. 2022, doi: 10.1016/j.neucom.2022.04.087.

[19] O. V. Hedde and M. A. Kumar, "Multi-camera multi-person tracking in surveillance system," in *2023 12th International Conference on Advanced Computing (ICoAC)*, Aug. 2023, pp. 1–7, doi: 10.1109/ICoAC59537.2023.10249836.

[20] A. Konieczka, J. Balcerek, M. Andrzejewski, S. Kaczmarek, and J. Szczygiel, "Smart real-time multi-camera people locating system," in *2022 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, Sep. 2022, pp. 98–103, doi: 10.23919/SPA53010.2022.9927922.

[21] S. X. X. Natasha and S. Srigrarom, “Person re-identification for multi-camera, multi-object tracking on robotic platforms,” in *2023 15th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Oct. 2023, pp. 121–126, doi: 10.1109/ICITEE59582.2023.10317712.

[22] K. Host, M. Ivašić-Kos, and M. Pobar, “Tracking handball players with the DeepSORT algorithm,” in *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods*, 2020, pp. 593–599, doi: 10.5220/0009177605930599.

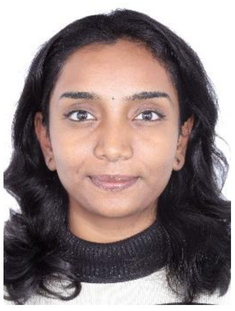
[23] H.-M. Hsu, T.-W. Huang, G. Wang, J. Cai, Z. Lei, and J.-N. Hwang, “Multi-camera tracking of vehicles based on deep features re-ID and trajectory-based camera link models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019, pp. 416–424.

[24] S. Kumar, Vishal, P. Sharma, and N. Pal, “Object tracking and counting in a zone using YOLOv4, DeepSORT and TensorFlow,” in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, Mar. 2021, pp. 1017–1022, doi: 10.1109/ICAIS50930.2021.9395971.

[25] H. Katayama, H. Yano, T. Yoshihisa, and H. Shimonishi, “A real-time human tracking system using multiple cameras to reduce network loads,” in *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, Jan. 2024, pp. 210–213, doi: 10.1109/CCNC51664.2024.10454843.

BIOGRAPHIES OF AUTHORS

Shashank Horakodige Raghavendra is a software developer who graduated with a B.E. in Computer Science from RV College of Engineering in 2024. Throughout college, he worked on several projects in various fields of Computer Science, including Computer Vision, and Blockchain Development with MERN Stack. His hobbies include playing football, watching a variety of sports, and video editing. His interests lie in full stack software development. He can be contacted at email: shashankhr.cs20@rvce.edu.in.



Yashasvi Sorapalli is a software engineering professional with a bachelor’s of engineering degree in computer science engineering from RV College of Engineering. She has worked on a myriad of projects ranging from AI and ML to blockchain to quantum computing. Apart from core computer science, her interests lie in video editing, puzzles, and swimming. She can be contacted at email: yashasvis.cs20@rvce.edu.in.

Nehashri Poojar S. V. is a software development engineer with a bachelor’s degree in Computer Science Engineering from RV College of Engineering, Bangalore, India. During her studies, she worked on projects involving NLP, Object Tracking, Machine Learning, backend development, Azure DevOps, and Azure Kubernetes Service. She is enthusiastic about exploring and contributing to the ongoing advancements in this dynamic field. Her interests lie in software development. She can be contacted at email: nehashripsv.cs20@rvce.edu.in.

Hrithik Maddirala is a software developer who graduated with a B.E. in Computer Science from RV College of Engineering, Bangalore, India. Passionate about technology and with a keen interest in innovative solutions, he is actively involved in academic pursuits and strives to contribute to the ever-evolving field of computer science. During college. He has worked on projects in areas like machine learning, NLP, and cloud computing, and actively participates in hackathons to apply his skills in solving real-world problems. He can be contacted at email: hrithikm.cs20@rvce.edu.in.

Dr. Ramakanth Kumar P. is the head of Department in the Department of Computer Science and Engineering of RV College of Engineering, Bangalore, India. Areas of research interest include digital image processing, pattern recognition, and natural language processing. He has been a Chairperson, Session Chair, and paper Reviewer at various International/National Conferences. He is also a member of the Board of Studies/ Board of Examiners in various reputed institutes in the state of Karnataka. He can be contacted at email: ramakanthkp@rvce.edu.in.

Dr. Azra Nasreen is an associate professor in the Department of Computer Science and Engineering of RV College of Engineering, Bangalore, India. Areas of research interest include video analytics, high-performance computing, and computer vision and has published articles in international journals presented several papers in international conferences and has completed many consultancy and sponsored research projects from various organizations of international repute. She can be contacted at email: azranasreen@rvce.edu.in.

Dr. Neeta Trivedi is the founder and CEO of Inferigence Quotient Pvt Ltd. She has over 3 decades of experience in manned and unmanned aircraft systems, Software and systems engineering, avionics, UAV mission systems, ground operating systems and ground support systems, UAV payload data processing, group autonomy, multisensor data fusion, and computer vision. She can be contacted at email: neeta@inferq.com.

Ashish Agarwal is a full-time computer vision engineer at inferigence Quotient Pvt Ltd. He has developed several projects and also guided several students in developing projects that have caused recognizable progress in the field of Computer Vision. He can be contacted at email: ashish@inferq.com.

Dr. Sreelakshmi K. is the Head of Department in the Department of Telecommunication Engineering at RV College of Engineering, Bangalore, India. Her area of research interest includes Microwave Communication and Nano Electronic Devices. She has received several awards and published several papers in journals and conferences of high repute. She can be contacted at email: sreelakshmik@rvce.edu.in.