Deep Learning Project phase 1

Authors

Zahra Maleki - 400110009

2025-01-24

Contents

1 Algorithm descriptions

1.1 Simple Online and Real-time Tracking(SORT)

SORT operates on a tracking-by-detection framework where objects are detected as bounding boxes in each frame, and the objective is to associate these detections across frames to form continuous trajectories. This approach emphasizes computational efficiency, relying on a simple combination of motion modeling using the Kalman filter and data association using the Hungarian algorithm.

The state of each object in SORT is represented as a seven-dimensional vector:

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]$$

where u and v are the center coordinates of the bounding box, s represents the scale (area) of the bounding box, and r is the aspect ratio. The velocity components, $\dot{u}$, $\dot{v}$, and $\dot{s}$, are included to model object motion. The algorithm assumes a linear constant velocity model for motion prediction, enabling efficient inter-frame displacement estimation. Using this model, the state at time t is predicted from the state at time $t-1$ using the Kalman filter:

$$x_{t|t-1} = Fx_{t-1|t-1}$$

where F is the state transition matrix. The Kalman filter also performs a correction step when new detections are associated with existing tracks. If no detection is associated with a track, the Kalman filter updates the state prediction without correction, relying solely on the motion model.

Data association is a critical component of the SORT algorithm, as it links detections in the current frame to the predicted positions of existing tracks. To compute the similarity between detections and predicted bounding boxes, SORT uses the intersection-over-union (IoU) metric, defined as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

A higher IoU indicates a better match. The cost matrix for data association is constructed based on IoU distances, and the Hungarian algorithm is used to solve the assignment problem optimally. Assignments are only considered valid if the IoU exceeds a predefined threshold, IoU_{min}. This ensures that tracks are not assigned to detections with minimal or irrelevant overlap.

When detections cannot be matched to any existing track, new tracks are initialized. The geometry of the detectionâs bounding box is used to define the initial state of the track, with the velocity components set to zero. Since the velocity is not observed at initialization, the covariance of the velocity components is set to large values to reflect this uncertainty. Tracks undergo a probationary period during which they must consistently be associated with detections to prevent false positives from being tracked. On the other hand, tracks are deleted if they are not matched with any detections for a specified number of frames, T_{lost}. The SORT algorithm is designed to handle short-term occlusions naturally. When an object is briefly occluded, its predicted position is not updated by a detection but continues to evolve based on the motion model.

1.2 Deep SORT Algorithm

The Deep SORT algorithm builds upon the Simple Online and Realtime Tracking (SORT) framework to improve object tracking performance by integrating motion and appearance-based metrics. While SORT relies solely on motion information (bounding box positions and velocities), Deep SORT introduces a deep neural network to extract appearance features.

At the core of Deep SORT is a Kalman filter, which predicts the position of tracked objects over time. The state of each object is represented in an 8-dimensional space:

$$(u, v, \gamma, h, \dot{u}, \dot{v}, \dot{\gamma}, \dot{h})$$

Where:

- u, v: The center of the bounding box in image coordinates.
- γ: The aspect ratio of the bounding box.
- h: The height of the bounding box.
- $\dot{u}, \dot{v}, \dot{\gamma}, \dot{h}$: Velocities of these parameters.

The Kalman filter assumes constant velocity and uses a linear observation model. Based on new detections, the Kalman filter updates its predictions to estimate the objectâs current position. Data association links detections to existing tracks by solving an assignment problem. Deep SORT addresses this problem by combining motion-based metrics from the Kalman filter with appearance-based metrics from a deep neural network.

The Mahalanobis distance measures how closely a detection matches the predicted state of a track. It is defined as:

$$d^{(1)}(i, j) = (d_j - y_i)^T S_i^{-1} (d_j - y_i)$$

Where:

- y_i: The predicted state of track i.
- S_i: Covariance matrix of the Kalman prediction.
- d_j: The new detection.

This metric calculates how many standard deviations a detection lies from the predicted track position. If the distance exceeds a threshold (based on the χ^2 distribution), the detection is excluded from consideration.

To improve tracking through occlusions, Deep SORT uses appearance features extracted by a deep neural network. Each detection is represented by a 128-dimensional feature vector (r_j), normalized to lie on the unit hypersphere ($\|r_j\| = 1$). The similarity between a detection and a trackâs historical appearance features (R_k) is measured using cosine distance:

$$d^{(2)}(i, j) = \min\{1 - r_j^T r_k^{(i)} \mid r_k^{(i)} \in R_k\}$$

Deep SORT combines the motion and appearance metrics into a single cost function:

$$c_{i,j} = \lambda d^{(1)}(i, j) + (1 - \lambda) d^{(2)}(i, j)$$

Here, λ is a weighting factor. For scenes with significant camera motion, it may be set to zero to rely purely on appearance features. Tracks and detections are associated using the Hungarian algorithm, which minimizes the overall cost.

When an object is occluded for several frames, its predicted position becomes less accurate. To address this, Deep SORT introduces a matching cascade, which prioritizes tracks that have been recently updated. Tracks are grouped by their age (number of frames since their last update), and associations are performed iteratively, starting with the newest tracks. This ensures that active tracks are matched first, reducing the likelihood of identity switches.

A pre-trained convolutional neural network (CNN) is used to extract discriminative appearance features for each detection. The network, trained on a large-scale person re-identification dataset, outputs a 128-dimensional feature vector for each bounding box. These features are critical for distinguishing objects with similar motion patterns but different visual appearances.

Listing 1 Matching Cascade

```
Input: Track indices $\mathcal{T} = \{1, \ldots, N\}$, Detection indices $\mathcal{D} = \{1, \ldots, M\}$, Maximum age $A_{\max}$
1: Compute cost matrix $\boldsymbol{C} = [c_{i,j}]$ using Eq. 5
2: Compute gate matrix $\boldsymbol{B} = [b_{i,j}]$ using Eq. 6
3: Initialize set of matches $\mathcal{M} \leftarrow \emptyset$
4: Initialize set of unmatched detections $\mathcal{U} \leftarrow \mathcal{D}$
5: for $n \in \{1, \ldots, A_{\max}\}$ do
6:     Select tracks by age $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$
7:     $[x_{i,j}] \leftarrow$ min_cost_matching$(\boldsymbol{C}, \mathcal{T}_n, \mathcal{U})$
8:     $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i,j) \mid b_{i,j} \cdot x_{i,j} > 0\}$
9:     $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} \cdot x_{i,j} > 0\}$
10: end for
11: return $\mathcal{M}, \mathcal{U}$
```

Figure 1: Matching Cascade

Name	**Patch Size/Stride**	**Output Size**
Conv 1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv 2	$3 \times 3/1$	$32 \times 128 \times 64$
Max Pool 3	$3 \times 3/2$	$32 \times 64 \times 32$
Residual 4	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 5	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 6	$3 \times 3/2$	$64 \times 32 \times 16$
Residual 7	$3 \times 3/1$	$64 \times 32 \times 16$
Residual 8	$3 \times 3/2$	$128 \times 16 \times 8$
Residual 9	$3 \times 3/1$	$128 \times 16 \times 8$
Dense 10		128
Batch and ℓ_2 normalization		128

Figure 2: Overview of the CNN architecture

Deep SORT manages tracks dynamically for Initialization so that New tracks are created for unmatched detections. Tracks are updated with associated detections Tracks are deleted if they remain unmatched for a predefined number of frames.

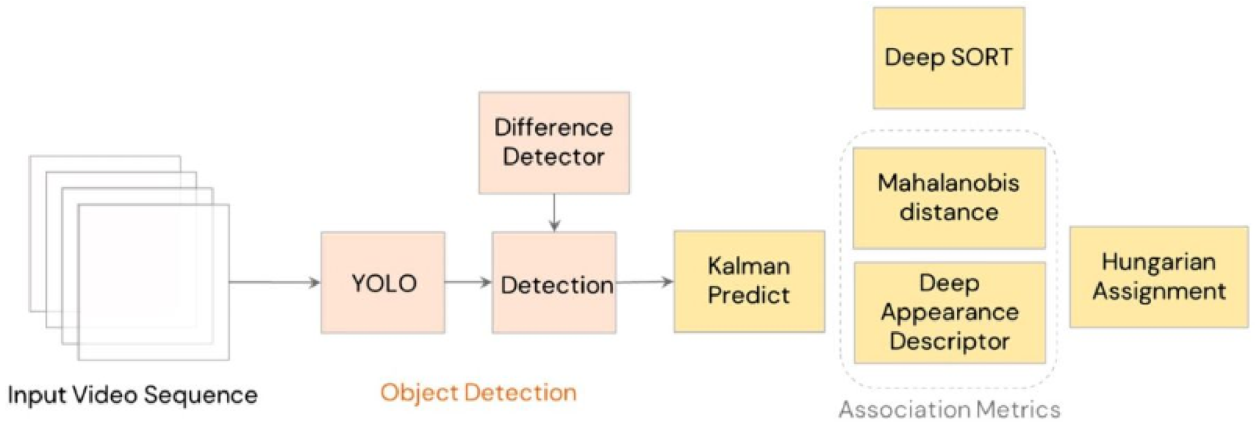


Figure 3: Architecture of Deep SORT

1.3 Fairness of Detection and Re-Identification in Multiple Object Tracking(FairMOT)

FairMOT combines object detection and re-identification (re-ID) into a unified framework to address issues of fairness and competition between these tasks. The input image is processed through an encoder-decoder network to generate high-resolution feature maps, enabling precise predictions for object detection and re-ID embeddings. The algorithm includes three primary components: detection heatmap estimation, bounding box regression, and re-ID feature extraction.

The detection heat map is generated to represent object centers. For a ground-truth bounding box $b_i = (x_{i1}, y_{i1}, x_{i2}, y_{i2})$, the object center is computed as:

$$c_x^i = \frac{x_{i1} + x_{i2}}{2}, \quad c_y^i = \frac{y_{i1} + y_{i2}}{2}.$$

On the feature map, the center is scaled by the downsampling stride s, giving:

$$c_x'^i = \lfloor c_x^i / s \rfloor, \quad c_y'^i = \lfloor c_y^i / s \rfloor.$$

The heatmap response M_{xy} at a pixel (x, y) is defined as:

$$M_{xy} = \max_i \exp\left(-\frac{(x - c_x'^i)^2 + (y - c_y'^i)^2}{2\sigma_c^2}\right),$$

where σ_c controls the spread of the Gaussian distribution. The focal loss is used to train the heatmap:

$$L_{\text{heat}} = -\frac{1}{N} \sum_{xy} \begin{cases} (1 - \hat{M}_{xy})^\alpha \log(\hat{M}_{xy}) & \text{if } M_{xy} = 1, \\ (1 - M_{xy})^\beta (\hat{M}_{xy})^\alpha \log(1 - \hat{M}_{xy}) & \text{otherwise.} \end{cases}$$

Here, $\hat{M}_{xy}$ is the predicted heatmap value, and α and β are hyperparameters.

Bounding box regression predicts the height and width of the bounding box $s = (h, w)$ and fractional corrections for center locations to adjust for downsampling. The loss for bounding box regression is:

$$L_{\text{box}} = \sum_{i=1}^{N} \|s_i - \hat{s}_i\|_1 + \lambda_s \|o_i - \hat{o}_i\|_1,$$

where λ_s is a weighting factor, $\hat{s}_i$ is the predicted size, and $\hat{o}_i$ is the predicted offset.

The re-ID branch extracts a 128-dimensional embedding $E_{x,y}$ for each object center. These embeddings are used to associate detections across frames. The re-ID branch is trained using a classification loss:

$$L_{\text{identity}} = -\sum_{i=1}^{N} \sum_{k=1}^{K} L_i(k) \log(p(k)),$$

where $L_i(k)$ is the one-hot ground truth label for the i-th object, K is the total number of identities, and $p(k)$ is the predicted class probability.

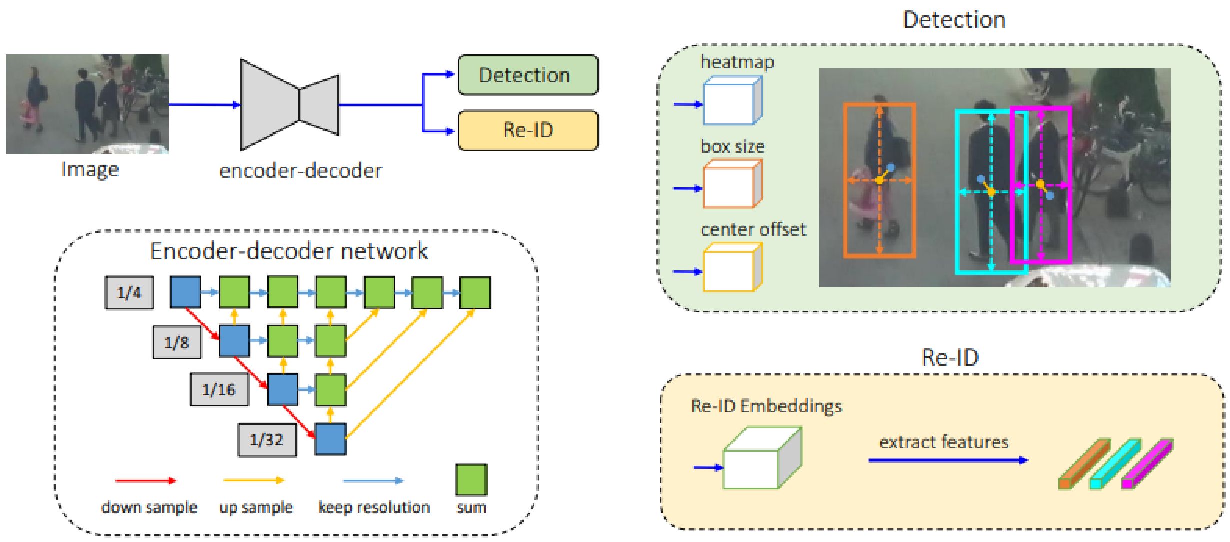


Figure 4: Overview of one-shot tracker FairMOT

The total loss combines detection and re-ID losses:

$$L_{\text{detection}} = L_{\text{heat}} + L_{\text{box}},$$

$$L_{\text{total}} = \frac{1}{2}\left(\frac{1}{e^{w_1}}L_{\text{detection}} + \frac{1}{e^{w_2}}L_{\text{identity}} + w_1 + w_2\right),$$

where w_1 and w_2 are learnable parameters to balance the two tasks.

During inference, the network outputs heatmaps, bounding box sizes and offsets, and re-ID embeddings. Non-Maximum Suppression (NMS) is applied to the heatmaps to extract object centers. Corresponding bounding boxes and embeddings are then used for tracking. Re-ID embeddings are compared using cosine similarity, and detections are associated across frames using the Hungarian algorithm.

1.4 ByteTrack: Multi-Object Tracking

The ByteTrack algorithm addresses challenges in multi-object tracking by incorporating both high-confidence and low-confidence detection boxes, ensuring robust tracking even in challenging scenarios like occlusions or motion blur. The detection results in each frame t, denoted by D_t, are separated into two categories based on a confidence threshold τ. High-confidence detections are defined as:

$$D_{\text{high}} = \{d \in D_t \mid \text{score}(d) > \tau\},$$

and low-confidence detections as:

$$D_{\text{low}} = \{d \in D_t \mid \text{score}(d) \leq \tau\}.$$

While most traditional algorithms discard low-confidence detections, ByteTrack uses them in a second stage to enhance tracking performance.

To predict the motion of objects, ByteTrack employs a Kalman filter. The state of each track k is represented as:

$$x_t = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}],$$

where u, v are the center coordinates of the bounding box, s is the scale (area), r is the aspect ratio, and $\dot{u}, \dot{v}, \dot{s}$ are their respective velocities. Using the Kalman filter, the state is predicted as:

$$\hat{x}_t = F x_{t-1} + \omega,$$

where F is the state transition matrix, and ω represents process noise.

For matching detections to tracks, ByteTrack uses the Intersection over Union (IoU) metric, defined as:

$$\text{IoU}(b_1, b_2) = \frac{\text{Area}(b_1 \cap b_2)}{\text{Area}(b_1 \cup b_2)},$$

which measures the overlap between two bounding boxes b_1 and b_2. A cost matrix is constructed for each track k and detection d as:

$$C_{kd} = 1 - \text{IoU}(b_k, b_d).$$

The Hungarian algorithm is then used to minimize this cost and assign detections to tracks.

The matching process occurs in two stages. In the first stage, high-confidence detections D_{high} are matched to all active tracks using IoU. Tracks and detections that remain unmatched after this step are denoted as T_{remain} and D_{remain}, respectively. In the second stage, the remaining tracks T_{remain} are matched to low-confidence detections D_{low} using the same IoU-based approach. This two-stage matching ensures that low-confidence detections, which are often discarded by traditional methods, contribute to recovering potentially missed tracks.

Once matching is complete, ByteTrack updates its tracks. Tracks successfully matched with detections are updated with their new bounding box and state information. High-confidence detections that remain unmatched (D_{remain}) are used to initialize new tracks. Tracks that remain unmatched for more than a predefined threshold T_{lost} are removed from the tracking list.

In cases where appearance embeddings are available, ByteTrack can use cosine similarity to improve matching. The similarity between a track k and a detection d is computed as:

$$\text{CosSim}(e_k, e_d) = \frac{e_k \cdot e_d}{\|e_k\| \|e_d\|},$$

where e_k and e_d are the embeddings of the track and detection, respectively. This additional information enhances the algorithm's ability to distinguish between objects with similar motion.

ByteTrack also supports interpolation for tracks that are temporarily occluded. If a track is not visible in frame t, its bounding box can be interpolated using:

$$B_t = B_{t_1} + \frac{t - t_1}{t_2 - t_1}(B_{t_2} - B_{t_1}),$$

where B_{t_1} and B_{t_2} are the bounding boxes before and after the occlusion.

By combining these techniques, ByteTrack achieves robust and efficient multi-object tracking, handling low-confidence detections effectively and maintaining consistent trajectories over time. This approach significantly improves tracking accuracy, especially in scenarios where traditional methods fail due to discarded detections or fragmented trajectories.

Comparison of Algorithms

SORT is extremely fast and simple to implement, but its accuracy is limited as it relies only on motion information. It struggles with occlusions and identity switches, making it less reliable in complex scenarios.

Deep SORT improves tracking accuracy by adding appearance-based features through a re-identification (Re-ID) network. While it handles occlusions much better than SORT, it is slower due to the added computational cost of extracting appearance embeddings.

FairMOT takes accuracy to the next level by combining object detection and Re-ID in a single framework. This approach ensures fairness between the two tasks and achieves excellent results in dense and complex scenes, while maintaining reasonable speed.

ByteTrack pushes accuracy even further by utilizing both high-confidence and low-confidence detections, reducing the chance of missed objects. It is efficient and operates in real-time, while being relatively easy to implement compared to FairMOT.

Algorithm	**Accuracy**	**Speed**	**Implementation Complexity**
SORT	Moderate; struggles with occlusions/identity switches	Very high (up to 260 FPS)	Simple; uses Kalman filter and IoU
Deep SORT	High; robust with occlusions using Re-ID	Moderate (25 to 30 FPS)	Moderate; integrates deep Re-ID model
FairMOT	Very high; state-of-the-art in dense scenes	Competitive (25 to 30 FPS)	Complex; custom multi-branch network
ByteTrack	Superior; robust to low-confidence detections	High (30+ FPS)	Moderate; extends SORT logic

2 Application of Tracking Algorithms in Sports Videos

Tracking algorithms play a significant role in analyzing sports videos, where objects such as players, balls, and referees need to be monitored for performance evaluation, strategy analysis, or creating interactive experiences for viewers. The choice of an algorithm and approach depends on the complexity of the scene and the number of objects being tracked. Below is an explanation of how the algorithms discussed in the papersâSORT, Deep SORT, FairMOT, and ByteTrackâcan be applied in sports scenarios:

- Tracking Players and Referees: In team sports like football or basketball, where multiple players and referees are moving on the field, Algorithms like Deep SORT and FairMOT are particularly useful in such cases. Deep SORT introduces appearance features to improve tracking accuracy and reduce identity switches, making it reliable when players cross paths or occlude each other. FairMOT further enhances tracking by combining detection and re-identification (Re-ID) in a single framework, ensuring high accuracy even in dense scenes.

- Tracking the Ball: For tracking the ball in sports, where the primary focus is on a single object moving at high speed, Algorithms like SORT and ByteTrack are efficient choices. SORT is lightweight and fast, making it ideal for scenarios where only the ball needs to be tracked, and its motion is relatively predictable. ByteTrack adds robustness by considering low-confidence detections, which is beneficial for tracking fast-moving objects like balls that may appear blurry or partially occluded in some frames.
- General Use Cases: SORT and ByteTrack can be used for quick analysis in less crowded sports scenarios, like tennis or table tennis, where only one or two objects are moving simultaneously. FairMOT is better suited for analyzing team sports where maintaining the identities of multiple players is crucial, such as in tactical analysis or player heatmaps.

Determining Between MOT and SOT for Sports Scenarios

The choice between Single Object Tracking (SOT) and Multiple Object Tracking (MOT) depends on the specific use case and the nature of the objects being tracked.

- When to Use SOT: Tracking a single ball: In scenarios where the focus is solely on tracking the ball's motionâsuch as analyzing its speed, trajectory, or interactions with playersâSOT is sufficient. Since SOT focuses on one object at a time, it is simpler and faster to implement, making it ideal for scenarios where no other objects need to be tracked.
- When to Use MOT: Tracking players, referees, and the ball: In complex sports like football, basketball, or hockey, where the movement and interaction of multiple players and referees are equally important, MOT becomes necessary.

This tailored approach ensures that the selected algorithm is both efficient and effective for the specific task in sports video analysis.

3 Tracking Performance Metrics

- MOTA (Multiple Object Tracking Accuracy): MOTA is a widely used metric to evaluate the overall tracking accuracy. It combines three error components: false positives (FP), false negatives (FN), and identity switches (IDSW):

  $$\text{MOTA} = 1 - \frac{\sum(\text{FN} + \text{FP} + \text{IDSW})}{\sum \text{GT}},$$

  where GT is the total number of ground-truth objects. Strengths: Provides a high-level summary of tracking performance. Limitations: It does not account for the quality of object identification, which makes it less sensitive to identity consistency.

- IDF1 (Identity F1 Score): IDF1 measures the accuracy of identity preservation during tracking by computing the F1 score of correctly identified detections:

  $$\text{IDF1} = 2 \cdot \frac{\text{IDTP}}{\text{IDTP} + \text{IDFP} + \text{IDFN}},$$

  where:

  IDTP: Identity True Positives (correctly matched tracks to ground truth), IDFP: Identity False Positives, IDFN: Identity False Negatives.

  Strengths: Focuses on identity consistency, making it a complement to MOTA. Limitations: Sensitive to occlusions and short-term losses of track continuity.

- Recall: Recall measures how many ground-truth objects are correctly tracked:

  $$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

  where TP is True Positives, and FN is False Negatives. Strengths: Highlights the algorithm's ability to track objects. Limitations: Does not consider identity preservation or false positives.

- Precision: Precision measures how many tracked objects correspond to ground-truth objects:

  $$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

  Strengths: Highlights the algorithm's ability to avoid false detections. Limitations: Alone, it doesnât provide a complete picture of tracking performance.

- Track Fragmentation (Frag): Measures the number of times a single ground-truth trajectory is fragmented into multiple tracks. Strengths: Identifies issues with track continuity. Limitations: Does not directly quantify the effect of fragmentation on tracking accuracy.

Addressing Metric Limitations

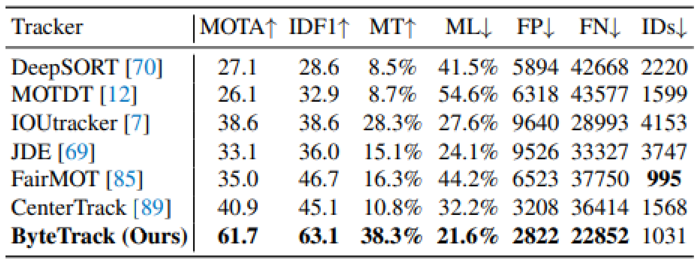
MOTA and IDF1 are often reported together to balance their strengths. While MOTA emphasizes overall tracking accuracy, IDF1 ensures identity consistency, providing a more comprehensive evaluation.

Metrics like IDF1 and Recall are sensitive to occlusions. Using Re-ID-based algorithms (e.g., Deep SORT, FairMOT) or robust matching strategies (e.g., ByteTrack) helps improve performance under occlusions by maintaining track continuity.

False positives and identity switches can be reduced by incorporating low-confidence detections (as ByteTrack does) or using appearance features for association (as in FairMOT and Deep SORT).

Algorithms like ByteTrack handle fragmentation effectively by associating low-confidence detections to extend tracks, reducing the number of fragmented tracks.

Overal, metrics like MOTA, IDF1, and Recall provide a detailed perspective on tracking performance. While MOTA summarizes tracking accuracy, IDF1 focuses on identity consistency, and Recall ensures



Tracker	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDs↓
DeepSORT [70]	27.1	28.6	8.5%	41.5%	5894	42668	2220
MOTDT [12]	26.1	32.9	8.7%	54.6%	6318	43577	1599
IOUtracker [7]	38.6	38.6	28.3%	27.6%	9640	28993	4153
JDE [69]	33.1	36.0	15.1%	24.1%	9526	33327	3747
FairMOT [85]	35.0	46.7	16.3%	44.2%	6523	37750	**995**
CenterTrack [89]	40.9	45.1	10.8%	32.2%	3208	36414	1568
ByteTrack (Ours)	**61.7**	**63.1**	**38.3%**	**21.6%**	**2822**	**22852**	1031

Figure 5: Comparison of the state-of-the-art methods using the metrics

that all objects are tracked. Combining these metrics addresses their individual limitations and offers a holistic evaluation of tracking algorithms.

4 Introduction to Sports Tracking Datasets

In the field of sports tracking, several datasets are available, each addressing specific challenges and use cases. These datasets vary in their focus, content, and suitability for sports scenarios. Below, we explore key datasets used for sports tracking and evaluate their strengths and limitations.

4.1 SportsMOT

The SportsMOT dataset is explicitly designed for multi-object tracking in sports scenarios. It includes 240 video sequences, over 150,000 frames, and 1.6 million bounding boxes. The dataset focuses on tracking players in basketball, volleyball, and football, providing diverse and challenging scenarios. Each player is assigned a unique ID, and bounding boxes are carefully annotated to ensure high quality.

SportsMOT captures the unique challenges of sports environments, such as fast and unpredictable player movements, frequent occlusions, and visually similar appearances due to uniforms. Additionally, it employs metrics like HOTA, IDF1, MOTA, and AssA to evaluate tracking performance comprehensively. This makes SportsMOT the most suitable dataset for testing and improving tracking algorithms in dynamic and densely populated sports scenarios.

4.2 MOT17

MOT17 is a general-purpose multi-object tracking dataset widely used for evaluating tracking algorithms. It focuses on pedestrian tracking in urban environments and includes crowded scenes with varying levels of complexity. However, MOT17 does not address the specific challenges of sports tracking, such as high-speed motion, frequent occlusions, or dense interactions between players. While it is a valuable benchmark for general tracking tasks, it is less suitable for sports-related applications.

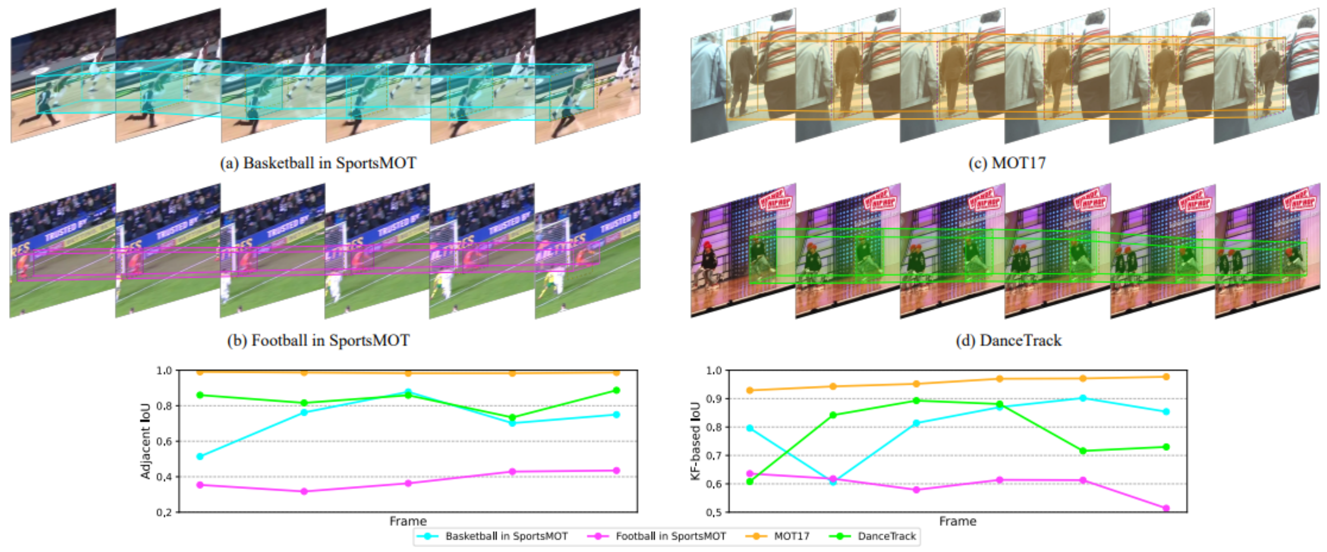


Figure 6: Sampled sequences of SportsMOT, MOT17 and DanceTrack

4.3 DanceTrack

DanceTrack provides a unique perspective on tracking appearance-similar objects. The dataset focuses on tracking dancers in coordinated movements, highlighting the challenge of distinguishing between objects with near-identical appearances. However, DanceTrack lacks the dynamic and fast-paced interactions that are typical in sports scenarios. While it is useful for evaluating appearance-based tracking methods, it does not provide the motion complexity required for sports tracking.

4.4 PoseTrack

PoseTrack is a dataset focused on human pose estimation and tracking. It provides keypoint annotations, making it ideal for analyzing individual player movements and postures in sports. This level of detail is particularly useful for tasks requiring fine-grained motion analysis, such as studying player biomechanics. However, PoseTrack is less suited for multi-object tracking tasks that rely on bounding box-level annotations.

Comparison and Recommendations

For sports-specific applications, SportsMOT stands out as the most relevant and comprehensive dataset. Its scale, diversity, and detailed annotations make it ideal for evaluating tracking algorithms in dynamic and densely populated sports environments.

If the task involves fine-grained analysis of player movements, such as joint motion or postures, PoseTrack is a good alternative due to its focus on pose estimation. However, for general multi-object tracking in sports, SportsMOT provides the most relevant benchmark. MOT17 and DanceTrack, while useful for other tracking challenges, lack the specific complexities and dynamics of sports scenarios.

5 Challenges in Tracking

The challenges of tracking are comprehensively addressed in various scenarios, with emphasis on sports tracking. These challenges significantly impact the performance and evaluation metrics of tracking algorithms.

5.1 Occlusion

Definition and Impact: Occlusion happens when an object being tracked is obscured, either partially or fully, by another object or environmental factors. In sports, this is common due to player interactions or overlapping movements. Occlusions primarily affect metrics like *IDF1* (identity consistency) and *Track Fragmentation (Frag)*, as maintaining object identities becomes challenging when occlusions break tracks.

Algorithm Adaptations: Algorithms like Deep SORT and FairMOT mitigate occlusion effects by using appearance-based models, such as re-identification (Re-ID) embeddings, to match objects after they reappear. ByteTrack extends tracking capabilities by associating low-confidence detections during occlusion, recovering tracks more effectively than simpler motion-based methods like SORT.

5.2 Scale Variation

Definition and Impact: Scale variation refers to the changing size of objects as they move closer to or farther from the camera, or due to camera zoom. This poses challenges for detection-based tracking, where bounding box sizes need to adapt dynamically. Metrics like *Recall* and *Precision* are directly affected as algorithms may miss objects or produce false positives due to inaccurate bounding box predictions.

Algorithm Adaptations: Algorithms like FairMOT use high-resolution feature maps to improve the detection of objects at varying scales. ByteTrack also adapts well to scale variation by leveraging its two-stage matching strategy, associating detections across scales effectively. SORT, being a simpler motion-based tracker, often struggles with rapid scale changes.

5.3 Illumination Change

Definition and Impact: Illumination changes occur due to dynamic lighting conditions, such as shadows, glare, or inconsistent lighting across the playing field. These variations can cause detection errors, leading to false positives or missed detections, which impact metrics like *MOTA* and *IDF1*.

Algorithm Adaptations: Motion-based algorithms like SORT are particularly sensitive to illumination changes, as they lack the robustness of appearance-based features. Deep SORT, FairMOT, and ByteTrack incorporate deep learning models that are better equipped to handle these variations by learning invariant features during training.

Impact on Metrics

Each of these challenges affects tracking metrics differently: *IDF1* is most impacted by occlusions, as identity preservation becomes difficult. *MOTA* summarizes overall tracking accuracy but does not capture identity-related issues. *Precision* and *Recall* are affected by scale variation and illumination change due to detection errors. *Track Fragmentation (Frag)* is a direct measure of the impact of occlusion and motion continuity.

These challenges are intrinsic to tracking scenarios, particularly in sports. Advanced algorithms provide robust solutions by integrating motion and appearance-based features, addressing these challenges effectively.

6 Dataset Availability and Usage

The complete SportsMOT dataset, which is designed for multi-object tracking in sports scenarios, is available for download. You can download it by clicking on the following text:

– Complete Dataset Download (33 GB)

The full dataset is approximately 33 GB in size, making it impractical for download on some personal computers. For such cases, an example dataset is provided to facilitate development and testing. You can download the example dataset by clicking the text below:

– Example Dataset Download

The example dataset is a smaller subset of the full SportsMOT dataset and is ideal for setting up and testing algorithms locally before scaling up to the complete dataset.

The dataloader implementation is in the jupyter notebook.

7 Conclusion and Recommendation

7.1 Challenges in the Reviewed Algorithms

The reviewed tracking algorithmsâSORT, Deep SORT, FairMOT, and ByteTrackâface several challenges in sports scenarios. These include occlusions, scale variation, high-speed motion, and lighting changes, which affect performance metrics like IDF1, MOTA, and Track Fragmentation.

One significant challenge is occlusion, where objects are hidden temporarily by other players or obstacles. This disrupts track continuity and identity preservation. While Deep SORT and FairMOT use Re-ID features to address occlusion, their performance declines in dense scenes. ByteTrack partially resolves this by utilizing low-confidence detections.

Scale variation is another issue caused by dynamic camera angles and zoom levels. SORT and Deep SORT often fail to adjust to these changes, while FairMOT and ByteTrack are more robust but require optimization for extreme cases. High-speed motion, common in sports, leads to identity switches and fragmented tracks, which are difficult for motion-based methods like SORT to handle. ByteTrackâs two-stage matching improves robustness but struggles in scenarios with rapid directional changes.

Lighting changes further challenge detection accuracy. Variations in shadows, glare, and field lighting increase false positives and missed detections. While appearance-based algorithms mitigate this issue to some extent, they still require improved feature extraction for extreme conditions.

7.2 Recommendations for Improvement

-Implement attention-based transformers to predict object trajectories during occlusions and recover identities effectively. Additionally, incorporating multi-camera setups in datasets like SportsMOT can provide alternative views, reducing the impact of occlusions in dense scenes.

-Develop multi-scale detection frameworks that automatically adjust to objects of varying sizes. Vision transformers or feature pyramid networks can dynamically handle scale variations, ensuring consistent accuracy even when object sizes change rapidly.

-Optimize models using techniques like pruning and quantization to maintain computational efficiency while preserving accuracy. Lightweight architectures, combined with hardware acceleration (e.g., GPUs), can ensure real-time performance for high-frame-rate sports videos.