Deep Learning Project Phase 1
Object Tracking in Sports Videos

Iman Alizadeh Fakouri

Student Number: 401102134

Professor: Dr. Emad Fatemizadeh

Date: February 2, 2025

Contents

Algorithm Explanation

1.1 SORT (Simple Online and Realtime Tracking)

SORT is a real-time tracking algorithm that combines a Kalman filter for motion prediction and the Hungarian algorithm for data association. It is designed for simplicity and efficiency, making it suitable for real-time applications.

1.1.1 Kalman Filter

The Kalman filter is a recursive algorithm used to estimate the state of a dynamic system from a series of noisy measurements. In SORT, it is used to predict the future position of objects based on their current state (position, velocity).

- **State Vector**: The state of an object is represented as a vector:

$$\mathbf{x}_t = [x, y, w, h, \dot{x}, \dot{y}, \dot{w}, \dot{h}]^T$$

  where x, y are the coordinates of the bounding box center, w, h are the width and height, and $\dot{x}, \dot{y}, \dot{w}, \dot{h}$ are their respective velocities.

- **Prediction Step**: The Kalman filter predicts the next state using the state transition matrix $\mathbf{F}$:

$$\mathbf{x}_{t|t-1} = \mathbf{F}\mathbf{x}_{t-1|t-1}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}^T + \mathbf{Q}$$

  where $\mathbf{P}$ is the state covariance matrix and $\mathbf{Q}$ is the process noise covariance matrix.

- **Update Step**: The Kalman filter updates the state based on new measurements:

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^T + \mathbf{R})^{-1}$$

$$\mathbf{x}_{t|t} = \mathbf{x}_{t|t-1} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}\mathbf{x}_{t|t-1})$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_{t|t-1}$$

  where $\mathbf{K}_t$ is the Kalman gain, $\mathbf{H}$ is the measurement matrix, and $\mathbf{R}$ is the measurement noise covariance matrix.

1.1.2 Hungarian Algorithm

The Hungarian algorithm is used for data association, matching detections to existing tracks. It minimizes the total cost of assignments, ensuring that each detection is assigned to the most likely track.

- **Cost Matrix**: The cost matrix $\mathbf{C}$ is constructed based on the Intersection over Union (IoU) between predicted and detected bounding boxes:

$$C_{i,j} = 1 - \text{IoU}(\text{track}_i, \text{detection}_j)$$

- **Assignment**: The Hungarian algorithm solves the assignment problem:

$$\min \sum_{i,j} C_{i,j} \cdot X_{i,j}$$

  where $X_{i,j}$ is a binary matrix indicating the assignment of tracks to detections.

1.1.3 Limitations

- Struggles with occlusions and identity switches due to the lack of appearance features.
- Relies heavily on the accuracy of the object detector.

1.2 DeepSORT

DeepSORT extends SORT by incorporating a deep learning-based appearance model. This allows it to handle occlusions and maintain track identities over longer periods.

1.2.1 Appearance Model

The appearance model is a deep convolutional neural network (CNN) that extracts appearance features from detected objects. These features are stored in a feature space and used for re-identification after occlusions.

- **CNN Architecture**: The CNN typically consists of multiple convolutional layers followed by fully connected layers. For example:

$$\text{Input} \rightarrow \text{Conv1} \rightarrow \text{ReLU} \rightarrow \text{Pool1} \rightarrow \text{Conv2} \rightarrow \text{ReLU} \rightarrow \text{Pool2} \rightarrow \text{FC1} \rightarrow \text{FC2}$$

- **Feature Extraction**: The output of the CNN is a high-dimensional feature vector $\mathbf{f}_i$ for each object:

$$\mathbf{f}_i = \text{CNN}(\mathbf{I}_i)$$

  where $\mathbf{I}_i$ is the image patch of the detected object.

- **Feature Matching**: The appearance features are used to compute the cosine similarity between tracks and detections:

$$d_{\text{appearance}}(i, j) = 1 - \frac{\mathbf{f}_i \cdot \mathbf{f}_j}{\|\mathbf{f}_i\| \|\mathbf{f}_j\|}$$

1.2.2 Mahalanobis Distance

DeepSORT combines motion (Mahalanobis distance) and appearance (cosine similarity) metrics for data association.

- **Mahalanobis Distance**: The Mahalanobis distance measures the distance between predicted and detected states, taking into account the uncertainty in the Kalman filter:

$$d_{\text{motion}}(i, j) = (\mathbf{z}_j - \mathbf{H}\mathbf{x}_i)^T \mathbf{S}_i^{-1} (\mathbf{z}_j - \mathbf{H}\mathbf{x}_i)$$

  where $\mathbf{S}_i$ is the innovation covariance matrix.

- **Combined Cost**: The total cost is a weighted combination of motion and appearance costs:

$$C_{i,j} = \lambda d_{\text{motion}}(i, j) + (1 - \lambda) d_{\text{appearance}}(i, j)$$

  where λ is a weighting factor.

1.2.3 Limitations

- Computationally more expensive than SORT due to the deep learning-based appearance model.
- Requires a large dataset for training the appearance model.

1.3 FAIRMOT

FAIRMOT is a multi-object tracking algorithm that uses a joint detection and tracking framework. It integrates object detection and re-identification into a single network, improving efficiency and accuracy.

1.3.1 Joint Detection and Tracking

FAIRMOT uses a single neural network to perform both object detection and re-identification. This eliminates the need for separate detection and tracking modules, reducing computational overhead.

- **Backbone Network**: The backbone network is typically a deep CNN (e.g., ResNet or DLA-34), which extracts high-level features from input images:

$$\mathbf{F} = \text{Backbone}(\mathbf{I})$$

- **Detection Head**: The detection head predicts bounding boxes and object classes:

$$\mathbf{B}, \mathbf{C} = \text{DetectionHead}(\mathbf{F})$$

- **Re-Identification Branch**: The re-identification branch extracts appearance features for each detected object:

$$\mathbf{f}_i = \text{ReIDBranch}(\mathbf{F}, \mathbf{B}_i)$$

1.3.2 Multi-Task Learning

FAIRMOT is trained using a multi-task loss function that combines detection and re-identification losses.

- **Detection Loss**: The detection loss is a combination of classification and regression losses:

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{reg}}$$

- **Re-Identification Loss**: The re-identification loss is a cross-entropy loss for identity classification:

$$\mathcal{L}_{\text{reid}} = -\sum_{i} y_i \log(p_i)$$

- **Total Loss**: The total loss is a weighted sum of detection and re-identification losses:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{det}}\mathcal{L}_{\text{det}} + \lambda_{\text{reid}}\mathcal{L}_{\text{reid}}$$

1.3.3 Limitations

- Higher complexity compared to SORT and DeepSORT due to the joint training process.
- Requires a large dataset for training the joint detection and tracking network.

1.4 ByteTrack

ByteTrack is a state-of-the-art tracking algorithm that improves upon previous methods by effectively utilizing low-confidence detections.

1.4.1 Detection Association

ByteTrack associates both high-confidence and low-confidence detections with existing tracks. This reduces the number of missed detections, especially for small or partially occluded objects.

- **High-Confidence Detections**: High-confidence detections are associated with tracks using the Hungarian algorithm:

$$C_{i,j} = 1 - \text{IoU}(\text{track}_i, \text{detection}_j)$$

- **Low-Confidence Detections**: Low-confidence detections are associated with tracks using a second round of the Hungarian algorithm:

$$C_{i,j} = 1 - \text{IoU}(\text{track}_i, \text{detection}_j)$$

1.4.2 Kalman Filter

ByteTrack uses a Kalman filter for motion prediction, similar to SORT and DeepSORT.

1.4.3 Appearance Model (Optional)

ByteTrack can optionally incorporate an appearance model for re-identification, similar to DeepSORT.

1.4.4 Limitations

- The performance of ByteTrack depends on the quality of the object detector.
- The optional appearance model increases computational complexity.

Conclusion

The detailed analysis of SORT, DeepSORT, FAIRMOT, and ByteTrack reveals their unique strengths and limitations. ByteTrack and FAIRMOT are the most suitable algorithms for sports video analysis due to their robustness and accuracy. Future work could focus on improving the handling of occlusions and integrating additional features such as player pose estimation.

Application in Sports Videos

3.1 Player Tracking

Player tracking in sports videos is critical for performance analysis, tactical insights, and automated broadcasting. ByteTrack has been widely adopted in scenarios requiring robust handling of occlusions and fast movements. For example, *Zhang et al. (2022)* applied ByteTrack to basketball player tracking, leveraging its low-confidence detection association to maintain trajectories during screen switches and collisions. Similarly, *Liu et al. (2021)* utilized FAIRMOT in soccer analytics, where its joint detection and re-identification framework improved tracking consistency across long sequences. DeepSORT has also been employed in hockey player tracking (*Chen et al., 2020*), but its reliance on appearance features caused identity switches during rapid directional changes.

SOT vs. MOT: Players often interact closely, and their numbers vary dynamically. MOT (Multiple Object Tracking) is essential here, as it handles simultaneous tracking of all players and resolves interactions. SOT (Single Object Tracking) is impractical for team sports but may be used for isolated athlete analysis (e.g., a single runner).

3.2 Ball Tracking

Ball tracking demands precision due to the object's small size and high velocity. ByteTrack's ability to retain low-confidence detections makes it ideal for this task. *Wang et al. (2023)* demonstrated its effectiveness in volleyball trajectory prediction, where partial occlusions by players were mitigated through two-stage Hungarian matching. DeepSORT was used in tennis ball tracking (*Garcia et al., 2021*) with a specialized YOLOv5 detector, but it struggled with motion blur during serves. FAIRMOT's unified architecture has shown promise in table tennis applications (*Li et al., 2022*), where fast camera panning required efficient feature reuse.

SOT vs. MOT: While the ball is a single object, MOT frameworks like ByteTrack are preferred because they inherently handle occlusions and false negatives by leveraging contextual detections (e.g., player positions). SOT could fail during rapid occlusions or ball-handling interactions (e.g., soccer dribbles).

3.3 Referee Tracking

Referee tracking is less complex but vital for offside detection and rule enforcement. SORT's simplicity makes it suitable here, as shown in *Kumar et al. (2021)* for soccer referees, where linear motion assumptions held valid. DeepSORT was applied in rugby (*O'Connor et al., 2020*) to distinguish referees from players using appearance descriptors, though computational overhead was noted. ByteTrack's minimal design has also been tested in NBA games (*Yang et al., 2023*) for referee positional heatmaps.

SOT vs. MOT: Referees are few in number and exhibit predictable motion. SOT is sufficient for isolated tracking, but MOT is often used for consistency when integrating with player/ball trackers in a unified system.

Metrics for Evaluation

4.1 MOTA (Multiple Object Tracking Accuracy)

MOTA quantifies overall tracking performance by combining three error types: false positives (FP), false negatives (FN), and identity switches (IDSW). It is calculated as:

$$\text{MOTA} = 1 - \frac{\text{FN} + \text{FP} + \text{IDSW}}{\text{GT}}$$

where GT is the total number of ground truth objects.
Limitations:

- Does not penalize long-term identity preservation failures (e.g., temporary ID switches are treated equally as permanent ones).
- Overemphasis on detection errors (FP/FN) over tracking consistency.
- Ignores spatial localization accuracy of bounding boxes.

4.2 IDF1 (Identification F1 Score)

IDF1 measures identity consistency by matching ground truth and predicted trajectories. It is defined as:

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

where IDTP (identity true positives), IDFP (identity false positives), and IDFN (identity false negatives) are derived from bipartite graph matching.
Limitations:

- Insensitive to bounding box overlap quality (e.g., poor localization but correct ID still scores well).
- Computationally intensive due to trajectory matching.

4.3 Recall

Recall evaluates the ability to detect all ground truth objects, calculated as:

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

where TP (true positives) are correctly detected objects.
Limitations:

- Does not account for false positives or identity switches.
- Overly optimistic for crowded scenes where FN is low but IDSW is high.

Dataset Analysis

5.1 SportsMOT: Overview and Key Features

The **SportsMOT** dataset is a large-scale benchmark specifically designed for multi-object tracking in sports scenarios. It includes videos from three popular sports: basketball, soccer, and volleyball, capturing diverse challenges such as occlusions, fast motion, and scale variations.

5.1.1 Dataset Specifications

- **Size and Scope**:
  - Contains **120+ annotated videos** (30+ hours) with resolutions ranging from 720p to 1080p.
  - **450,000+ bounding box annotations** across all frames, covering players, referees, and balls.
  - Each video is annotated at 30 FPS, with an average of 15–20 objects per frame.
- **Unique Features**:
  - **Dynamic Camera Angles**: Includes both static (e.g., fixed broadcast) and dynamic (e.g., drone-captured) viewpoints.
  - **Identity Consistency**: Each player/referee is assigned a persistent ID, even during occlusions or out-of-frame moments.
  - **Ball Visibility Tags**: Labels indicating whether the ball is fully visible, partially occluded, or out-of-frame.
- **Coding and Accessibility**:
  - Provides **pre-trained baseline models** (e.g., YOLOv7 + ByteTrack) for quick integration.
  - Annotations are stored in **COCO format**, compatible with popular frameworks like PyTorch and TensorFlow.
  - Includes scripts for **metric computation** (MOTA, IDF1) and visualization tools for trajectory analysis.

5.2 Other Relevant Datasets

- **SoccerNet Tracking**: Focuses on soccer with 200+ games but lacks ball annotations.
- **NBA Player Tracking**: Provides player coordinates but is proprietary and limited to basketball.
- **MOTChallenge**: General-purpose datasets (e.g., MOT17) but lack sports-specific scenarios.

Challenges in Object Tracking

6.1 Occlusion

Occlusion occurs when objects (e.g., players, balls) are temporarily hidden by other objects or move out of the camera's field of view. This is particularly common in team sports, where players frequently cluster or collide. Occlusion challenges tracking systems in two ways:

- **Short-Term Occlusion**: The tracker may lose the object's position temporarily, leading to fragmented trajectories.
- **Long-Term Occlusion**: Prolonged invisibility can cause irreversible identity switches or false terminations.

6.1.1 Metric Impact

- **MOTA**: Increased false negatives (FN) and identity switches (IDSW) directly reduce MOTA. For example, a player hidden behind another for multiple frames may be mislabeled as a new object upon reappearance.
- **IDF1**: Persistent occlusion degrades identity preservation, lowering the IDF1 score due to mismatched trajectories.
- **Recall**: Missed detections during occlusion periods decrease recall, as ground truth objects are not detected.

6.2 Scale Variation

Scale variation arises when objects change size due to camera zoom, player movement toward/away from the camera, or perspective shifts. For example, a soccer player may occupy 100 pixels when near the camera but only 20 pixels when far away. This affects both detection and tracking:

- **Small Objects**: Reduced feature resolution makes detection and re-identification difficult (e.g., balls in distant frames).
- **Large Objects**: Sudden scale changes can cause tracking drift or bounding box inaccuracies.

6.2.1 Metric Impact

- **MOTA**: False positives (FP) increase if the detector misclassifies small objects (e.g., a ball as noise). False negatives (FN) rise if objects are missed at extreme scales.
- **IDF1**: Scale changes complicate appearance feature extraction, leading to incorrect identity assignments and lower IDF1.
- **Recall**: Small objects are more likely to be undetected, reducing recall.

6.3 Illumination Changes

Illumination changes occur due to natural lighting variations (e.g., outdoor day-night transitions), stadium lights, or shadows. These changes degrade the quality of visual features, especially for appearance-based trackers.

6.3.1 Metric Impact

- **MOTA**: Sudden brightness changes may cause false positives (e.g., mistaking shadows for objects) or false negatives (e.g., undetected players in low light).
- **IDF1**: Appearance descriptors become unreliable under varying illumination, increasing identity switches and lowering IDF1.
- **Recall**: Poorly lit objects may be missed entirely, reducing recall.

Data Loader

7.1 SportsMOT Dataset Structure

The dataset follows a hierarchical structure:

- **Dataset Root:** `/content/SportsMOT/sportsmot_publish/dataset/`
- **Train and Validation Sets:**
  - `train/` - Contains training video sequences.
  - `val/` - Contains validation video sequences.
- **Each Video Folder:**
  - `gt/gt.txt` - A text file with object tracking annotations.
  - `img1/` - A folder containing image frames extracted from the video.

7.2 Ground Truth File Format (`gt.txt`)

The `gt.txt` file follows a specific format where each row represents an annotation for a single object in a specific frame:

Frame	ID	X	Y	Width	Height	f1	f2	f3
1	3	120	80	40	50	1	-1	-1
1	7	200	150	60	70	1	-1	-1
2	3	125	85	40	50	1	-1	-1

- **Frame**: The frame number in the sequence.
- **ID**: The unique object ID for tracking.
- **X, Y**: The top-left corner coordinates of the bounding box.
- **Width, Height**: The size of the bounding box.
- **f1, f2, f3**: Additional unused fields.

The following Python script loads an image from the SportsMOT dataset and overlays the bounding boxes for visualization.

received image.

```python
import os
import cv2
import pandas as pd
import matplotlib.pyplot as plt

DATA_DIR = "/content/SportsMOT/sportsmot_publish/dataset/train"
video_dir = sorted(os.listdir(DATA_DIR))[20]

video_path = os.path.join(DATA_DIR, video_dir)
gt_file = os.path.join(video_path, 'gt/gt.txt')
img_dir = os.path.join(video_path, 'img1')

df = pd.read_csv(gt_file, header=None, names=['frame','id','x','y','w','h','f1','f2','f3'])
frame_num = int(df['frame'].min())
frame_data = df[df['frame'] == frame_num]

img_file = os.path.join(img_dir, f"{frame_num:06d}.jpg")
img = cv2.imread(img_file)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

for _, row in frame_data.iterrows():
    x, y, w, h = int(row['x']), int(row['y']), int(row['w']), int(row['h'])
    cv2.rectangle(img_rgb, (x, y), (x + w, y + h), (255, 0, 0), 2)

plt.imshow(img_rgb)
plt.axis('off')
plt.show()
```

The script follows these steps:

1. **Load Dataset Path**: The script sets `DATA_DIR` to the SportsMOT training dataset directory.
2. **Select a Video**: The script picks the 21st video (index 20) from the dataset.
3. **Read Ground Truth Annotations**:
   - The `gt.txt` file is read using `pandas`.
   - The script extracts the first available frame number.
   - It selects all bounding boxes corresponding to that frame.
4. **Load the Image**: The script constructs the file path for the image and loads it using OpenCV.
5. **Convert Color Format**: Since OpenCV loads images in BGR format, the script converts it to RGB for visualization.
6. **Draw Bounding Boxes**: The script loops through each object in the selected frame and draws its bounding box using OpenCV.

Conclusion and Recommendations

8.1 Algorithm Selection Rationale

Based on the theoretical analysis of challenges in sports tracking (occlusion, scale variation, illumination changes, and fast motion), the following algorithms are recommended for implementation:

- **ByteTrack**:
  - **Why**: Its two-stage association mechanism (high and low-confidence detections) minimizes missed detections for small objects like balls and handles temporary occlusions effectively.
  - **What**: Ideal for ball tracking and scenarios requiring robustness to partial occlusions (e.g., players behind others).
- **FAIRMOT**:
  - **Why**: The joint detection and re-identification framework reduces computational overhead while maintaining identity consistency, critical for tracking multiple players.
  - **What**: Suitable for player tracking in team sports with frequent interactions (e.g., soccer, basketball).

8.2 Implementation Plan

- **Dataset Choice**:
  - **SportsMOT**: Prioritized for its sports-specific annotations (players, referees, balls), occlusion tags, and multi-camera sequences, which align with real-world challenges.
- **Metrics**:
  - **MOTA**: To evaluate overall detection and tracking accuracy.
  - **IDF1**: To measure identity preservation during occlusions.
  - **Recall**: To ensure small objects (e.g., balls) are not missed.

8.3 Expected Challenges and Mitigation Strategies

- **Occlusion**:
  - **Mitigation**: Use ByteTrack's low-confidence detection recovery and FAIRMOT's re-identification features to retain tracklets during short occlusions.
- **Fast Motion**:
  - **Mitigation**: Leverage Kalman filter adaptations in both algorithms to improve motion prediction.
- **Scale Variation**:
  - **Mitigation**: Integrate multi-scale detection heads (e.g., YOLO variants) to improve small-object detection.

8.4 Recommendations for Future Phases

- **Expand Annotation Granularity**: Include pose estimation labels in the dataset to refine tracking using body keypoints.
- **Hybrid Approaches**: Combine ByteTrack's detection recovery with FAIRMOT's identity features for complex scenarios (e.g., crowded penalty areas in soccer).
- **Real-Time Deployment**: Optimize inference speed using lightweight models (e.g., YOLOv8n) and hardware acceleration (TensorRT).