Multiple Object Tracking and Forecasting: Jointly Predicting Current and Future Object Locations

Oluwafunmilola Kesa, Olly Styles, Victor Sanchez
Dept. of Computer Science, University of Warwick, UK
{funmi.kesa, o.c.styles, v.f.sanchez-silva}@warwick.ac.uk

Abstract

This paper introduces a joint learning architecture (JLA) for multiple object tracking (MOT) and multiple object forecasting (MOF) in which the goal is to predict tracked objects' current and future locations simultaneously. MOF is a recent formulation of trajectory forecasting where the full object bounding boxes are predicted rather than trajectories alone. Existing works separate multiple object tracking and multiple object forecasting. Such an approach can propagate errors in tracking to forecasting. We propose a joint learning architecture for multiple object tracking and forecasting (MOTF). Our approach reduces the chances of propagating tracking errors to the forecasting module. In addition, we show, through a new data association step, that forecasting predictions can be used for tracking objects during occlusion. We adapt an existing MOT method to simultaneously predict current and future object locations and confirm that JLA benefits both the MOT and MOF tasks.

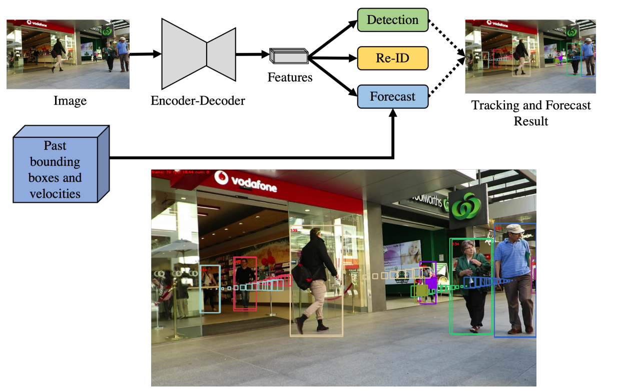
Image
Encoder-Decoder
Features
Detection
Re-ID
Forecast
Tracking and Forecast Result
Past bounding boxes and velocities

Figure 1. **JLA:** We propose a joint learning architecture for tracking multiple objects and forecasting their trajectories. The forecasting branch takes the past bounding boxes and velocities, and an embedding as input to predict the objects' locations into the future. JLA performs three tasks: detection, re-ID, and trajectory forecasting. The network is trained end-to-end. Full bounding boxes are predicted but are reduced in the image for clarity.

1. Introduction

Tracking and forecasting are two important tasks in computer vision. Tracking aims to estimate the locations of unique objects in a video [28, 48, 39, 47]. Meanwhile, forecasting aims to predict future short- and long-term locations of objects in a video using the objects' past location information [36, 32]. Tracking and forecasting are critical components for several applications, including autonomous driving [16], video surveillance [18] and smart elderly care [17]. Researchers have studied tracking and forecasting independently, and learning-based models exist for these two tasks. However, the two tasks share similar properties as both require the objects of interest to be detected and re-identified across frames.

A recent formulation of trajectory forecasting called Multiple Object Forecasting (MOF) extends the traditional MOT to predict objects' future coordinates and scale in terms of their bounding boxes [36]. While MOF exploits the advantages of an object-based architecture, MOF requires pre-computed trajectories from an object tracker to estimate future locations. Separating tracking and forecasting poses some challenges in MOF, including high computational cost, which can limit its real-time application.

In MOT methods, issues such as identity (ID) switches and incorrect predictions are still dominant. To this end, motion prediction is used to rectify wrong estimations caused by similar appearance embeddings or occlusion [39, 48]. More concretely, a Kalman Filter is typically used to provide short-term motion estimations, which are used to refine the bounding box estimations [41, 48]. However, such a motion prediction method cannot predict non-linear trajectories. In contrast, MOF can provide non-linear predictions that can benefit MOT methods.

Motivated by the above observations, we introduce the multiple object tracking and forecasting (MOTF) task, in which the aim is to simultaneously detect, track, and predict objects' current and future locations. We implement the MOTF task through a joint learning architecture (JLA) that models non-linear trajectories by using trajectory forecasts generated by an embedded forecasting network. By using these forecasts to refine bounding box estimations, our JLA can predict objects' locations during occlusion, an important advantage over previous works that rely on linear motion predictions. As a result, JLA can reduce ID switches considerably. Our contributions can be summarized as follows:

- We introduce MOTF, a new task for multiple object tracking and forecasting (Section 3).
- We propose a joint learning architecture that predicts current and future object locations simultaneously (Section 4).
- We employ the trajectory forecasts to refine objects' locations in lieu of the Kalman Filter during data association. This reduces ID switches caused by similar appearance embeddings (Section 6.2).
- We introduce a new step for data association of objects during occlusion in which the trajectory forecasts are used to estimate the location of occluded objects in the current frame (Section 6.2).
- We evaluate our model on the MOTChallange benchmarks. Our proposed method reduces ID switches on the MOT20 benchmark by 47% compared to FairMOT [48] (Section 7.4).

2. Related Work

Tracking and trajectory forecasting methods are vital to real-world applications such as video surveillance [15, 22, 19]. It is crucial that objects of interest can be accurately detected and tracked over a period of time. Also, these applications might predict objects' movements and intentions into the future. In this section, we review existing works on MOT and trajectory forecasting.

Multiple Object Tracking. Recent MOT methods leverage deep neural networks' representational power to learn the identity, appearance and pose of several objects to associate targets across several frames. Several of these methods follow the tracking-by-detection paradigm, in which objects are first detected as targets and then associated with subsequent detections, in the form of the bounding boxes, to form trajectories [41, 5, 42, 48, 28]. Specifically, these methods use bounding box estimations from an external detector and focus on improving the association of these estimations to form trajectories. Clearly, this approach can benefit from a strong object detector and a re-identification (re-ID) method. However, the high computational cost of training an object detector and a re-ID model separately, and their slow inference time, limit the real-time application of such an approach. To address this issue, one approach is to train the object detector and re-ID model simultaneously in an end-to-end manner. The works in [38, 39, 48] show that it is possible to design models that can simultaneously detect and predict identity embeddings by adding a re-ID head to existing object detectors. Our work takes this approach a step further to simultaneously detect, track, and forecast objects' locations by adding a trajectory forecast head to a tracking method.

An important part of an MOT method is the task of association, which can be performed in an online or offline manner. Online methods [41, 37, 48, 3, 25, 39, 27] associate bounding box estimations sequentially up to the current frame. In offline methods [7, 13, 29, 44, 6, 12], the order of association does not apply and future frame estimations can be used in data association. Offline methods can interpolate missing objects' locations by using the past, current and future estimations to generate more accurate trajectory predictions than those given by online methods. However, offline methods cannot be used in real-time applications [20]. Our method uses trajectory forecasts to estimate occluded objects' locations in an online fashion.

Trajectory Forecasting. Trajectory forecasting methods predict the future location of an object identified in a video. Datasets for trajectory forecasting typically consist of footage of pedestrians or vehicles captured from a birds-eye perspective [33]. Methods use past location information in addition to various categories of features such as the interactions between objects [1, 31] and the estimated final destination [23, 9]. A smaller number of works have also considered trajectory forecasting from an object-level viewpoint [43, 45, 36, 2], where visual information such as human pose estimations or optical flow can be incorporated more easily. All the previous works mentioned before use either ground truth or pre-computed object tracking results prior to forecasting.

Joint Tracking and Trajectory Forecasting. A small number of works have attempted joint tracking and trajectory forecasting [40, 21]. However, these works predict trajectories as 2D positions on the ground from a top-down view. [35] uses lidar for trajectory forecasting. Our work predicts trajectories as full bounding boxes from an object-level viewpoint.

3. Multiple Object Tracking and Forecasting

MOTF draws from MOT and MOF and follows a similar formulation to these two tasks. In this section, we formalize the problem and explain the evaluation metrics.

3.1. Problem Formulation

Consider a video with n frames $f_0, f_1, \dots, f_{n-1}$. Given frame f_s at timestep s,

- let I be a set of identifiable objects in the frame such that object $i \in I$, and
- let $\boldsymbol{b}_s = \{b_s^i\}_{i=1}^{I}$ be the set of bounding boxes for all

identifiable objects in frame f_s. Each bounding box $b_s^i = (x, y, w, h)$ is represented by the location of its centroid, (x, y), and its width and height, (w, h).

The aim of MOT is to associate all the frame-wise bounding boxes, $\{b_0^i\}, \{b_1^i\}, \ldots, \{b_{n-1}^i\}$ for all $i \in I$, to a unique identifier, $k \in 1, 2 \ldots K$, where K is the total number of unique objects across all the frames, such that a set of tracks, $\mathcal{T} = \{t^k\}_{k=1}^K$, is computed for the entire video sequence, where t^k represents the k^{th} unique track. This association task can be formulated as a bipartite or linear assignment problem where only one bounding box is linked to another bounding box in a subsequent frame. Therefore, an object in frame f_s is not associated with any other object in the same frame.

Similarly, given a sequence of frames $f_{s-p}, f_{s-p+1}, \ldots, f_s$, with their respective set of tracks $\{t_{s-p}^k\}, \{t_{s-p+1}^k\}, \ldots, \{t_s^k\}$ for all $k \in K$, the task of MOF is to predict the future set of bounding boxes $\{b_{s+1}^k\}, \{b_{s+2}^k\}, \ldots, \{b_{s+q}^k\}$ for all $k \in K$, for future frames $f_{s+1}, f_{s+2}, \ldots, f_{s+q}$, where p is the number of past frames used as input and q is the length of predictions into the future [36]. Note that in MOF, the tracks are pre-determined before forecasting.

Our goal is to create a JLA that jointly tracks and forecasts objects' locations. Thus, we formulate the MOTF task as a joint problem of MOT and MOF. Given frame f_s and a sequence of p past bounding boxes $\{b_{s-p}^k\}, \{b_{s-p+1}^k\}, \ldots, \{b_{s-1}^k\}$ for all $k \in K$, MOTF aims to compute tracks $\{t_s^k\}$ for all $k \in K$ at frame f_s and forecast each track's current and future bounding boxes, *i.e.*, $\{b_s^k\}, \{b_{s+1}^k\}, \{b_{s+2}^k\}, \ldots, \{b_{s+q}^k\}$ for all $k \in K$. Unlike MOF, the current frame bounding boxes, *i.e.*, $\{b_s^k\}$ for all $k \in K$, are also predicted to evaluate the accuracy of the jointly trained forecasting model in predicting the location of the detected objects. In this work, we set $p = 10$ and $q = 60$, which correspond to less than 1 second in the past and predicting 2 seconds into the future, respectively, at a frame rate of 30Hz.

3.2. Evaluation Metrics

MOTF is a joint task of tracking and forecasting. We employ the evaluation metrics of both tracking and forecasting. We use the *CLEAR* metrics [4], and *IDF1* values [30] to evaluate the trajectory tracking performance. We use *ADE/FDE* [1], and *AIOU/FIOU* [36] metrics to evaluate the trajectory forecasting performance.

4. Proposed JLA

In this section, we present JLA, a joint learning architecture for MOTF. JLA draws from existing architectures for tracking and forecasting [48, 36, 2]. We use the FairMOT model [48] as our base model because this architecture already performs detection and tracking. We add a forecasting branch to the network as shown in Figure 1 and train the resulting architecture end-to-end. FairMOT consists of a backbone network called DLA-34 [50], an object detection head, and a re-ID head. More details about the FairMOT architecture can be found in [48].

The design of the trajectory forecasting branch is shown in Figure 2. The goal of this branch is to predict future bounding boxes of objects using the past bounding box information. The trajectory forecasting network consists of recurrent neural networks (RNN) used to encode and decode the past bounding boxes and predict future bounding boxes. The components of the network are listed below:

(i) An RNN to encode past bounding boxes and velocities, which are denoted by $\{B_{s-p}^k\}, \ldots, \{B_{s-1}^k\}$, for all $k \in K$.

(ii) A fully-connected layer to encode DLA-34 feature embeddings retrieved from the tracking network.

(iii) An RNN to decode past bounding boxes and velocities.

(iv) An RNN to decode future velocities, which are denoted by $\{\hat{V}_s^k\}, \ldots, \{\hat{V}_{s+q}^k\}$, for all $k \in K$.

(v) A trajectory concatenation layer to convert future velocities to bounding boxes.

4.1. Past Bounding Box and Velocity Encoder

An RNN (PastEncoder in Figure 2) is used to extract features from past bounding boxes. This encoder captures the velocity of each object by iterating over historical information.

Given frame f_s and past bounding boxes $\{b_{s-p}^k\}, \{b_{s-p+1}^k\}, \ldots, \{b_{s-1}^k\}$ for all $k \in K$, we construct a sequence of p sets of 8-dimensional vectors $\boldsymbol{B} \in \mathbb{R}^{p \times 8}$. This sequence of 8-dimensional vectors can be written as $\boldsymbol{B} \equiv \{\{B_j^k\}_{k=1}^K\}_{j=s-p}^{s-1} \equiv \{\{B_{s-p}^k\}, \{B_{s-p+1}^k\}, \ldots, \{B_{s-1}^k\}\}$, where for each object k in frame f_j, $B_j^k = (x_j^k, y_j^k, w_j^k, h_j^k, \Delta x_j^k, \Delta y_j^k, \Delta w_j^k, \Delta h_j^k)$ and (x_j^k, y_j^k) represents the location of the centroid of the corresponding bounding box, (w_j^k, h_j^k) represents the width and height of the bounding box, $V_j^k = (\Delta x_j^k, \Delta y_j^k, \Delta w_j^k, \Delta h_j^k)$ represents the velocity, and Δ represents the change between consecutive timesteps computed as:

$$\Delta u_j^k = u_j^k - u_{j-1}^k \quad \forall u \in \{x, y, w, h\}. \tag{1}$$

As shown in Figure 2, an RNN (PastEncoder) takes the sequence of past bounding boxes and velocities, $\boldsymbol{B}$, and generates a final hidden state vector h_p^e, that summarizes the sequence. The final hidden state vector is achieved by repeatedly updating the previous hidden state vector with

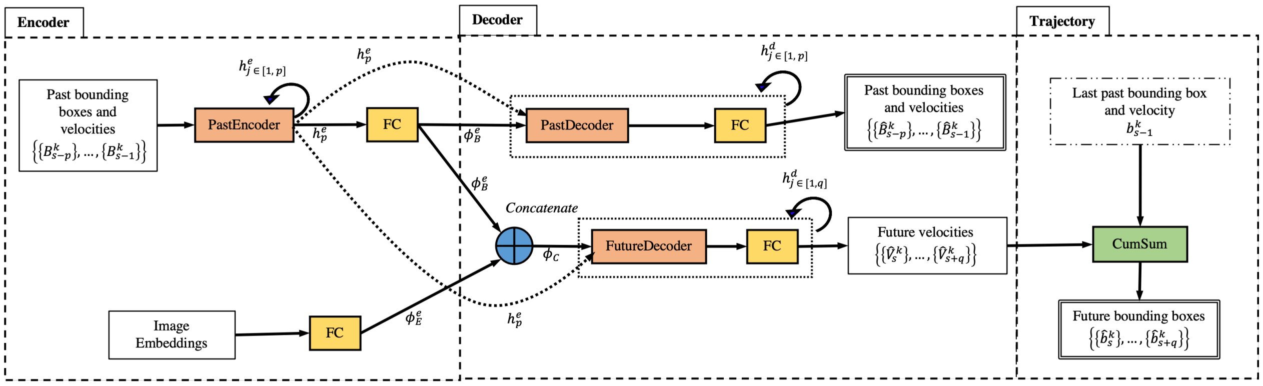
Encoder
Decoder
Trajectory
Past bounding boxes and velocities $\{\{B^k_{s-p}\},\dots,\{B^k_{s-1}\}\}$
PastEncoder
$h^e_{j\in[1,p]}$
h^e_p
FC
ϕ^e_B
PastDecoder
$h^d_{j\in[1,p]}$
Past bounding boxes and velocities $\{\{\hat{B}^k_{s-p}\},\dots,\{\hat{B}^k_{s-1}\}\}$
Concatenate
ϕ_C
FutureDecoder
$h^d_{j\in[1,q]}$
Future velocities $\{\{\hat{V}^k_s\},\dots,\{\hat{V}^k_{s+q}\}\}$
Last past bounding box and velocity b^k_{s-1}
CumSum
Future bounding boxes $\{\{\hat{b}^k_s\},\dots,\{\hat{b}^k_{s+q}\}\}$
Image Embeddings
ϕ^e_E

Figure 2. **Trajectory Forecast Architecture:** The trajectory forecasting branch of JLA takes past bounding boxes, velocities, and DLA-34 feature embeddings (image embeddings) as input. It uses a Recurrent Neural Network (RNN) and a fully connected (FC) ReLU layer to encode the past bounding boxes, and an FC ReLU layer to encode the image embeddings. Two different decoders are used to decode past bounding boxes and velocities, and predict future velocities. Finally, a trajectory concatenation layer transforms the predicted future velocities into future bounding boxes.

the input $\{B^k_j\}$ for p timesteps. The hidden state vector is initialized to zero. The resulting final hidden state vector is then passed through a fully connected layer with ReLU activations to generate a 256-dimensional feature vector ϕ^e_B.

4.2. Embedding Encoder

The embedding encoder is used to capture features from the DLA-34 backbone network. The DLA-34 network provides visual context for the predicted objects' bounding boxes in the current frame. The use of visual features in the trajectory forecasting module improves the accuracy of the bounding box estimations. The DLA-34 features are shared across the detection, re-ID, and forecast branches.

Given an input frame with dimension $H_s \times W_s$, we append a forecast head to the DLA-34 network to generate a feature map $\boldsymbol{D} \in \mathbb{R}^{256\times H\times W}$ where $H = H_s/4$ and $W = W_s/4$. The top N features of $\boldsymbol{D}$ are selected resulting in a $256 \times N$ embedding, where N is the maximum number of objects for multi-scale learning and set to a default value $N = 500$. This allows us to learn a high-dimensional vector representation of the input frame in the forecasting network.

We then pass the feature map $\boldsymbol{D}$ to a fully connected layer to generate a 256-dimensional vector ϕ^e_E. The resulting encoding is concatenated to the past bounding box and velocity encoding ϕ^e_B, similar to STED [36]. However, STED uses optical flow to pass visual information to the forecasting network instead of the DLA-34 feature embeddings. The ablation study (Table 4) shows that using DLA-34 features embeddings in the forecasting model helps to improve the performance of the entire JLA.

4.3. Past Bounding Box and Velocity Decoder

Another RNN (PastDecoder in Figure 2) is used to reproduce the past bounding boxes and velocities. The purpose of using the decoder to reproduce the input is to ensure that the model learns the correct input representation [2]. Thus, we can define an objective function to penalize the decoder when it deviates from the input.

At each timestep, the decoder first uses the encoding ϕ^e_B to update a previous hidden state vector and then passes the updated hidden state vector through a fully connected layer to generate an 8-dimensional vector. The hidden state vector is initialized to the final hidden state vector h^e_p of the encoder. The output of the decoder is a set of predicted past bounding boxes and velocities. Similar to the ground truth bounding boxes and velocities, $\boldsymbol{B}$, the predicted past bounding boxes and velocities can be represented as a sequence of p sets of 8-dimensional vectors $\hat{\boldsymbol{B}} \in \mathbb{R}^{p\times 8}$ where $\hat{\boldsymbol{B}} \equiv \{\{\hat{B}^k_j\}^K_{k=1}\}^{s-1}_{j=s-p} \equiv \{\{\hat{B}^k_{s-p}\},\{\hat{B}^k_{s-p+1}\},\dots,\{\hat{B}^k_{s-1}\}\}$ and $\hat{B}^k_j = (\hat{x}^k_j, \hat{y}^k_j, \hat{w}^k_j, \hat{h}^k_j, \Delta\hat{x}^k_j, \Delta\hat{y}^k_j, \Delta\hat{w}^k_j, \Delta\hat{h}^k_j)$. We use an L1 loss to penalize the decoder as follows:

$$L_{past} = \frac{1}{(K \times p \times 8)} \sum_{k=1}^{K} \sum_{j=s-p}^{s-1} \|B^k_j - \hat{B}^k_j\|_1. \tag{2}$$

4.4. Future Velocity Decoder

A third RNN (FutureDecoder in Figure 2) is used to predict q future velocities for the identified K objects. The past bounding boxes encoding ϕ^e_B is concatenated with the embedding ϕ^e_E, resulting in a 512-dimensional vector ϕ_C. As shown in Figure 2, ϕ_C is passed through a fully connected ReLU layer q times while updating the previous hidden state vector at each timestep. The hidden state vector is initialized to the final hidden state vector h^e_p of the encoder. The decoder generates predicted future velocities $\hat{\boldsymbol{V}} \in \mathbb{R}^{q\times 4}$ where $\hat{\boldsymbol{V}} \equiv \{\{\hat{V}^k_j\}^K_{k=1}\}^{s+q}_{j=s} \equiv \{\{\hat{V}^k_s\},\{\hat{V}^k_{s+1}\},\dots,\{\hat{V}^k_{s+q}\}\}$ and $\hat{V}^k_j = (\Delta\hat{x}^k_j, \Delta\hat{y}^k_j, \Delta\hat{w}^k_j, \Delta\hat{h}^k_j)$. We do not penalize this decoder directly based on recommendations in [2]. The trajectory concatenation layer, discussed next, is used to penalize the future bounding boxes instead.

4.5. Trajectory Concatenation Layer

The trajectory concatenation layer is used to transform the future velocities to bounding boxes [2]. This layer adds the last frame bounding boxes $\{b_{s-1}^k\}_{k=1}^K$, to the cumulative sum of the velocities to generate the predicted future bounding boxes $\hat{\boldsymbol{F}} \in \mathbb{R}^{q\times4}$, where $\hat{\boldsymbol{F}} \equiv \{\{\hat{b}_j^k\}_{k=1}^K\}_{j=s}^{s+q} \equiv \{\{\hat{b}_s^k\}, \{\hat{b}_{s+1}^k\}, \{\hat{b}_{s+2}^k\}, \dots, \{\hat{b}_{s+q}^k\}\}$. The cumulative sum of the velocities is computed using Eq. 3. The cumulative sum of the velocities is used to compute the predicted future bounding boxes (Eq. 4).

$$\hat{Z}_i^k = \begin{cases} \hat{V}_i^k & \text{if } i = 1 \\ \hat{V}_i^k + \hat{Z}_{i-1}^k & \text{for } i = 2\dots q \end{cases} \tag{3}$$

$$\hat{b}_{s+i-1}^k = b_{s-1}^k + (i \times \hat{Z}_i^k) \quad \text{for } i = 1\dots q. \tag{4}$$

Hence, we can define an objective function to penalize the predicted future locations. We use an L1 loss function defined as:

$$L_{future} = \frac{1}{(K\times q\times 4)} \sum_{k=1}^{K}\sum_{j=s}^{s+q} \|\hat{b}_j^k - \hat{b}_j^k\|_1. \tag{5}$$

We compute the forecast loss as:

$$L_{for} = L_{past} + L_{future}. \tag{6}$$

5. Training JLA

The entire network is trained end-to-end using a multi-task uncertainty loss [8]. The multi-task uncertainty loss performs a weighted linear sum of the losses for each task and the weights are learned automatically from the data [8]. Given the detection loss L_{det}, re-ID loss L_{id}, and forecast loss L_{for}, the total loss function is defined as:

$$\begin{aligned} L_{total} = \tfrac{1}{2}(&e^{-s_{det}}L_{det} + e^{-s_{id}}L_{id} + e^{-s_{for}}L_{for} \\ &+ s_{det} + s_{id} + s_{for}), \end{aligned} \tag{7}$$

where s_{det}, s_{id} and s_{for} are the weights for detection, re-ID, and forecast, respectively. We initialize the weights to values between -2.0 to 5.0 [8], which are then updated automatically by the model. The equations for L_{det} and L_{id} are defined in FairMOT [48].

The input to the model is the current frame and the past bounding boxes for the observed objects in previous frames. We use a Gated Recurrent Unit (GRU) for the PastEncoder, PastDecoder, and FutureDecoder implementation. However, any other type of RNN can be used for the implementation of the encoder and decoders. Although the past bounding boxes are supplied for the trajectory forecasting branch, we observe that the performance of the two other branches, *i.e.*, the detection and re-ID branches, improve greatly due to the shared image embedding (Section 7).

Different from existing trajectory forecasting methods, we propose using a variable length of past and future bounding boxes during training and a fixed length for evaluation. This implies that each object can have any number of past and future bounding boxes, less than or equal to the fixed values of p and q, respectively. This allows the network to learn the objects' historical information early during training without waiting for a complete set of past or future bounding boxes. Based on this idea, we re-formalize the past and future losses as:

$$L_{past} = \frac{1}{(\sum_{k=1}^{K} p_s^k \times 8)} \sum_{k=1}^{K}\sum_{j=s-p_s^k}^{s-1} \|B_j^k - \hat{B}_j^k\|_1, \tag{8}$$

$$L_{future} = \frac{1}{(\sum_{k=1}^{K} q_s^k \times 4)} \sum_{k=1}^{K}\sum_{j=s}^{s+q_s^k} \|b_j^k - \hat{b}_j^k\|_1. \tag{9}$$

where $p_s^k \le p$ and $q_s^k \le q$ represents the number of ground truth past and future bounding boxes available at frame f_s for each tracked object k, respectively. During training, we only use p_s^k past and q_s^k future bounding boxes for computing the loss despite predicting p past and q future bounding boxes.

6. Online Inference

In this section, we present the network inference and data association for JLA. The online association is similar to that used by FairMOT [48], however, we replace the Kalman Filter estimations with trajectory forecasts and add an extra step for estimating objects' location during occlusion. The algorithm for this step is explained in Section 6.2.

6.1. Network Inference

Our complete JLA network predicts trajectory forecasts in addition to the heatmap, bounding box offset, and bounding box size generated by the base model [48]. The size of the input frame is 1088×608, as in previous literature [39, 48]. We initialize the past bounding box and velocity information to zero in the first three frames. As new objects are detected and tracked, we store their previous locations. We require at least two previous locations to predict future locations for an object.

6.2. Data Association

We use the detections and trajectory forecasts for data association. The trajectory forecasts serve two purposes at inference time: (i) to generate short-term estimations used in lieu of a Kalman Filter to prevent associating detections with large motion and (ii) to generate bounding box estimation when an object is occluded.

Algorithm 1 SHORT-TERM FORECAST FOR MOTION FUSION

```
procedure FUSEMOTION($reidDist, tracks, detections, \lambda, l$)
    for $i \leftarrow 0$ to $length(tracks)$ do
        $track \leftarrow tracks[i]$
        $d \leftarrow IOUDistance(track, detections)$
        if $hasForecasts(track)$ then
            $forecasts \leftarrow getForecasts(track, l)$
            $dists \leftarrow IOUDistance(forecasts, detections)$
            $m \leftarrow minimum(dists, axis = 0)$
            $d \leftarrow d * m$
        end if
        $reidDist[i, d \geq 1] \leftarrow reidDist[i, d \geq 1] * 2$
        $costs[i] \leftarrow \lambda * reidDist[i] + (1 - \lambda) * d$
    end for
    return $costs$
end procedure
```

Algorithm 2 FORECAST FOR OCCLUSION

```
procedure FORECAST($track, frameCenter, \lambda, maxTime, thresh$)
    if $hasForecasts(track)$ then
        $cost \leftarrow track.lostTime/maxTime$
        $forecast \leftarrow getForecastAtLostTime(track)$
        $dist \leftarrow frameDistance(forecast, frameCenter)$
        $cost \leftarrow \lambda * dist + (1 - \lambda) * cost$
        if $cost < thresh$ then
            return $forecast$
        end if
    end if
    return $none$
end procedure
```

In the first frame, all detections above the confidence threshold are initialized as new tracks. Detections in subsequent frames are linked using three key steps discussed next. We set the state of tracks that are unmatched after the three steps to lost. If a track is lost a predetermined number of times, we remove the track. We set the maximum lost time to 30 frames.

Re-ID Features and Motion Fusion. The purpose of this step is to match tracked objects with new detections using re-ID features and bounding box overlap. A cosine distance matrix between tracks and detections computed on re-ID features is fused with short-term motion distance computed on trajectory forecasts. The motion distance is estimated using the IOU distance between a subset of the predicted trajectory forecasts and the detections. The trajectory forecast with the minimum IOU distance is selected and used to refine the re-ID distance matrix (Algorithm 1). If the minimum IOU distance between each detection and the predicted trajectory forecasts is too large, we increase its re-ID cosine distance by a factor of two. This helps to mitigate identity switches caused by incorrect re-ID features. A cost matrix is computed by calculating a weighted sum of the re-ID cosine distance and the minimum IOU distance (Algorithm 1). In Algorithm 1, λ is a weight to regularize the re-ID and IOU distances and l is the number of trajectory forecasts used to compute the IOU distance. We set $\lambda = 0.75$ and $l = 10$. Matches are found by passing the cost matrix to a Hungarian method [14] for linear assignment. The ablation study in Section 7.3 shows that the trajectory forecasts are more accurate than a Kalman Filter for short-term motion predictions.

IOU Association. In this step, we use the IOU distance to associate unmatched tracks and unmatched detections in situations where the re-ID features and short-term forecasts are not sufficient for data association. Matches are found by using a Hungarian method for linear assignment [14] on the IOU distance.

Forecast Association. We introduce a new step for data association in which trajectory forecasts are used to estimate objects' location during occlusion. At this stage, the unmatched tracks are either false positives or occluded. When an object is occluded, visual information to re-identify the object is not available. Trajectory forecasts can provide spatio-temporal information to estimate an object's location during occlusion. We compute a cost using the distance of the trajectory forecast to the centre of the frame and the lost time of the track (Algorithm 2). We assume that an unmatched track at the centre of the frame is likely to be occluded. The lost time of the track is incremented when a track is not associated with a new detection. The lost time avoids keeping an undetected object alive infinitely. We set $\lambda = 0.5$, $maxTime = 20$, and $thresh = 0.55$ in Algorithm 2.

7. Experiments

7.1. Datasets

We evaluate the performance of JLA using different amounts of training data. For the ablation studies (Section 7.3), we train JLA on 50% of the MOT17 training set and evaluate the model on the remaining 50%. For evaluation on the MOTChallenge server, we use the same training data as FairMOT [48]. The training data is described below:

- We use ETH [11] and CityPersons [46] to train the detection branch because these datasets provide only bounding box annotations.
- We use PRW [49], MOT17 [24], and CalTech [10] to train the detection, re-ID, and trajectory forecast branches because these datasets provide both bounding box and identity annotations.
- We use CUHK-SYS [42] to train the detection and re-ID branches. CUHK-SYS provides both bounding box and identity annotations but the frames are not sequential and are therefore not suitable for forecasting.

We evaluate the tracking result of JLA on the tests sets of the MOT15, MOT16, MOT17, and MOT20 benchmarks (Sec-

	IDF1 ↑	MT ↑	IDs ↓	MOTA ↑
FairMOT	70.9	140	441	67.1
FairMOT_CV	72.8	156	357	67.5
FairMOT_KF	72.5	149	279	68.1
JLA	**75.3**	**169**	**262**	**69.1**

Table 1. Results of multiple object tracking (MOT) on the MOT17 dataset.

	AIOU ↑	FIOU ↑	ADE ↓	FDE ↓
FairMOT	-	-	-	-
FairMOT_CV	33.7	14.9	112.3	205.1
FairMOT_KF	38.1	18.9	104.9	195.3
JLA	**39.1**	**22.1**	**97.0**	**177.5**

Table 2. Results of multiple object forecasting (MOF) evaluation on the MOT17 dataset. We set the number of past bounding boxes used for prediction to $p = 10$ and the number of predictions to $q = 60$.

App + Forecast	Box IOU	Occlusion Forecast	IDF1 ↑	MT ↑	IDs ↓	MOTA ↑
✓	✗	✗	72.7	139	381	67.7
✓	✓	✗	72.8	139	366	67.8
✗	✓	✗	60.4	145	1185	64.3
✗	✓	✓	65.0	145	976	64.8
✓	✗	✓	74.7	168	301	68.9
✓	✓	✓	**75.3**	**169**	**262**	**69.1**

Table 3. Evaluation of the data association components in JLA on the MOT17 dataset. We set the number of past bounding boxes used for prediction to $p = 10$ and the number of predictions to $q = 60$.

DLA34 Embedding	IDF1 ↑	MT ↑	IDs ↓	MOTA ↑
✗	72.8	150	**262**	65.8
✓	**75.3**	**169**	**262**	**69.1**

Table 4. Evaluation of the DLA34 feature embedding in JLA on the MOT17 dataset. We set the number of past bounding boxes used for prediction to $p = 10$ and the number of predictions to $q = 60$.

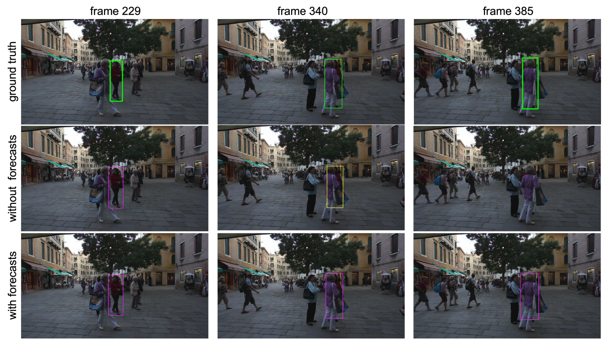

Figure 3. **Example tracking results during occlusion (best viewed in color)**. The same color represents the same tracked person across the sample frames. Mixed colors represent identity switches. The first row is the ground truth which shows an occluded person in frame 340 and frame 385. The second row contains results from the base model without forecasts trained on half the MOT17 train dataset and validated on the other half. The occluded target is re-identified as another person in frame 340 and lost in frame 385. The third row contains sample frames from our JLA model with forecasts trained on half the MOT17 dataset and validated on the other half. The occluded target is tracked up to frame 385.

tion 7.4).

7.2. Implementation Details

We finetune JLA on a model pre-trained on the CrowdHuman dataset [34]. The pre-training process uses a self-supervised approach in which a unique identity label is assigned to each bounding box and the model is trained without any tracking information [48].

We train JLA on six datasets (Section 7.1) with Adam optimizer for 30 epochs with a starting learning rate of 10^{-4}, which decays to 10^{-5} after 20 epochs. The training takes about 60 hours on two GeForce RTX 2080 GPUs with a batch size of 8. We use the standard data augmentation techniques used in the literature [48] including color jittering, rotation, and scaling. Finally, we finetune JLA for 20 epochs on the training datasets of MOT20 and MOT15 to evaluate their respective test datasets. The results on the MOTChallenge benchmarks are discussed in Section 7.4.

7.3. Ablation Studies

We first compare the performance of JLA with a constant velocity and the Kalman Filter baseline methods. We also perform ablation studies to evaluate the impact of the various components of JLA. We discuss these evaluations next.

Comparison with Baselines. A constant velocity model and the Kalman Filter are often used as baseline methods in trajectory forecasting. For a fair comparison of the JLA with the baseline methods, we modify the data association step in FairMOT [48] to be the same as in JLA. Then, for each detected object, we generate q trajectory forecasts using a constant velocity method (FairMOT_CV in Table 1) and a Kalman Filter (FairMOT_KF in Table 1). We set $q = 60$. We evaluate the tracking and trajectory forecasting performance using the metrics mentioned in Section 3.2. We do not compare our work with STED because STED requires an external detector to precompute object trajectories, which will result in an unfair comparison.

The results in Table 1 and Figure 3 show that using our proposed data association steps improves the performance of the base method, FairMOT [48]. The number of ID switches (IDs) is reduced considerably compared to both cases, constant velocity and the Kalman Filter. Also, the

Dataset	Tracker	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	IDs ↓	FPS ↑
MOT15	TubeTK [26]	58.4	53.1	39.3	18.0	854	5.8
	FairMOT [48]	**60.6**	**64.7**	**47.6**	**11.0**	**591**	**30.5**
	JLA (Ours)	55.8	63.2	42.7	16.1	644	22.4
MOT16	TubeTK [26]	64.0	59.4	33.5	19.4	1117	1.0
	JDE [39]	64.4	55.8	35.4	20.0	1544	18.5
	CTrackerV1 [27]	67.6	57.2	32.9	23.1	1897	6.8
	FairMOT [48]	**74.9**	72.8	44.7	**15.9**	1074	**25.9**
	JLA (Ours)	73.8	**75.0**	**44.9**	22.8	**719**	19.7
MOT17	TubeTK [26]	63.0	58.6	31.2	19.9	4137	3.0
	CTrackerV1 [27]	66.6	57.4	32.2	24.2	5529	6.8
	FairMOT [48]	73.7	72.3	43.2	**17.3**	3303	**25.9**
	JLA (Ours)	**74.0**	**74.0**	**45.1**	20.5	**2292**	19.0
MOT20	FairMOT [48]	**61.8**	67.3	68.8	**7.6**	5243	**25.9**
	JLA (Ours)	60.2	**68.7**	59.8	10.5	**2780**	14.3

Table 5. Comparison of state-of-the-art methods under the "private" category on the MOTChallenge benchmarks. These methods are categorized under the *private detections* because they use more datasets for training.

number of mostly tracked objects and accuracy increases. This shows that we can detect some occluded objects using trajectory forecasts.

In addition, Table 2 shows that the trajectory forecasting branch in JLA performs better than the constant velocity and Kalman Filter predictions with an increase in AIOU and FIOU, and a decrease in ADE and FDE.

Analysis of Data Association Components in JLA. As discussed in Section 6.2, JLA uses three components for data association: appearance embedding fused with short-term forecasts, bounding box IOU, and forecasts during occlusion. We study the impact of these individual components on the performance of the model. Where applicable, one or more components are turned off and the model is evaluated without the component(s). The results of the study are shown in Table 3.

The best performance is achieved when all the components are turned on. When *trajectory forecast during occlusion* is turned off, the number of ID switches (IDs) increases from 262 to 366; the number of mostly tracked decreases from 169 to 139; IDF1 and MOTA decrease from 75.3% to 72.8% and from 69.1% to 67.7% respectively. Associating data based on bounding box IOU alone gives the worst performance.

Image Embedding in Trajectory Forecasting. As explained in Section 4.2, the image embedding from the DLA34 network provides visual features for the trajectory forecasting network. Previous literature concatenates optical flow encoding to the previous bounding box encoding [36] to provide visual information to the trajectory forecasting network. Optical flow is computationally expensive and requires to be computed separately [36]. In our work, we concatenate the DLA34 output features with the previous bounding box encoding. This approach is less expensive and provides high-dimensional features for the trajectory forecasting network. We train JLA without this image embedding and compare its performance against the case of JLA trained with the embedding.

The result in Table 4 shows that including DLA34 features in the trajectory forecasting network improves the performance of JLA. MOTA increases from 65.8% to 69.1%, IDF1 increases from 72.8% to 75.3%, and the number of mostly tracked increases from 150 to 169.

7.4. Results on MOTChallenge Benchmarks

We compare our method with the top methods in the MOTChallenge benchmarks under the private category. The private category uses an external detector or datasets for training. As shown in Table 5, JLA reduces the ID switches (IDs) compared to FairMOT by 33%, 31%, and 47% for MOT16, MOT17, and MOT20, respectively. JLA does not perform as well on MOT15 because the performance of the trajectory forecast is dependent on the availability of ground truth information. MOT15 has missing tracking information during occlusion, which can result in a high number of false positives.

8. Conclusion

We have introduced a joint learning architecture for multiple object tracking and forecasting. We have shown that trajectory forecasts can be used in lieu of Kalman Filters to model non-linear trajectories. Also, we have shown that future predictions can be used to estimate objects' locations during occlusion. Our evaluations on the MOTChallenge benchmarks show that our architecture reduces the ID switches within an MOT context considerably. Our work shows promises for future research to develop more sophisticated architectures that improve multiple object tracking and forecasting.

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, pages 961–971. IEEE Computer Society, 12 2016.

[2] Junaid Ahmed Ansari and Brojeshwar Bhowmick. Simple means Faster: Real-time human motion forecasting in monocular first person videos on CPU. Technical report, 2020.

[3] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. Technical report, 2019.

[4] Keni Bernardin and Rainer Stiefelhagen. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):1–10, 5 2008.

[5] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. Technical report, 2016.

[6] Jiahui Chen, Hao Sheng, Yang Zhang, and Zhang Xiong. Enhancing detection model for multiple hypothesis tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2143–2152, 2017.

[7] Wongun Choi and Silvio Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. In *European Conference on Computer Vision*, pages 553–567. Springer, 2010.

[8] Roberto Cipolla, Yarin Gal, and Alex Kendall. Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7482–7491. IEEE Computer Society, 12 2018.

[9] Nachiket Deo and Mohan M. Trivedi. Trajectory Forecasts in Unknown Environments Conditioned on Grid-Based Plans. *arXiv preprint arXiv:2001.00735*, 2020.

[10] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009*, pages 304–311. Institute of Electrical and Electronics Engineers (IEEE), 3 2010.

[11] Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc Van Gool. A mobile vision system for robust multi-person tracking. In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.

[12] Joao F Henriques, Rui Caseiro, and Jorge Batista. Globally optimal solution to multi-object tracking with merged measurements. In *2011 International Conference on Computer Vision*, pages 2470–2477. IEEE, 2011.

[13] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M. Rehg. Multiple hypothesis tracking revisited. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704, 2015.

[14] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 3 1955.

[15] K. Susheel Kumar, Shitala Prasad, Pradeep K. Saroj, and R. C. Tripathi. Multiple cameras using real time object tracking for surveillance and security system. *Proceedings - 3rd International Conference on Emerging Trends in Engineering and Technology, ICETET 2010*, pages 213–218, 2010.

[16] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H.S. Torr, and Manmohan Chandraker. DESIRE: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 2165–2174, 2017.

[17] Young-Sook Lee and HoonJae Lee. Multiple object tracking for fall detection in real-time surveillance system. In *2009 11th International Conference on Advanced Communication Technology*, volume 03, pages 2308–2312, 2009.

[18] Xufeng Lin, Chang Tsun Li, Victor Sanchez, and Carsten Maple. On the detection-to-track association for online multi-object tracking. *Pattern Recognition Letters*, 146:200–207, 6 2021.

[19] Joachim Lohn-Jaramillo, Khari Elijah Jarrett, Laura Ray, Richard Granger, and Elijah Bowen. Time-first tracking: An efficient multiple-object tracking architecture for dynamic surveillance environments. *ICPRAM 2021 - Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods*, pages 602–611, 2021.

[20] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[21] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[22] Chandrajit M, Girisha R, and Vasudev T. Multiple Objects Tracking in Surveillance Video Using Color and Hu Moments. *Signal & Image Processing : An International Journal*, 7(3):15–27, 2016.

[23] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *European Conference on Computer Vision*, 2020.

[24] Anton Milan, Laura Leal-Taixe, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A Benchmark for Multi-Object Tracking. Technical report, 2016.

[25] Bo Pang, Yizhuo Li, Yifan Zhang, Muchen Li, and Cewu Lu. Tubetk: Adopting tubes to track multi-object in a one-step training model. Technical report, 2020.

[26] Bo Pang, Yizhuo Li, Yifan Zhang, Muchen Li, and Cewu Lu. Tubetk: Adopting tubes to track multi-object in a one-step training model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[27] Jinlong Peng, Changan Wang, Fangbin Wan, Yang Wu, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue

Huang, and Yanwei Fu. Chained-Tracker: Chaining Paired Attentive Regression Results for End-to-End Joint Multiple-Object Detection and Tracking. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12349 LNCS, pages 145–161. Springer Science and Business Media Deutschland GmbH, 7 2020.

[28] Zhixiong Pi, Huai Qin, Changxin Gao, and Nong Sang. Jointly detecting and multiple people tracking by semantic and scene information. *Neurocomputing*, 412:244–251, 2020.

[29] Zhen Qin and Christian R Shelton. Improving multi-target tracking via social grouping. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1972–1978. IEEE, 2012.

[30] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9914 LNCS, pages 17–35. Springer Verlag, 2016.

[31] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European Conference on Computer Vision*, 2016.

[32] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M. Kitani, Dariu M. Gavrila, and Kai O. Arras. Human motion trajectory prediction: a survey. *International Journal of Robotics Research*, 39(8):895–935, 2020.

[33] Amir Sadeghian, Vineet Kosaraju, Agrim Gupta, Silvio Savarese, and Alexandre Alahi. TrajNet: Towards a Benchmark for Human Trajectory Prediction. *arXiv preprint*, 2018.

[34] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. CrowdHuman: A Benchmark for Detecting Human in a Crowd. *arXiv*, 4 2018.

[35] Shashank Srikanth, Junaid Ahmed Ansari, R Karnik Ram, Sarthak Sharma, J Krishna Murthy, and K Madhava Krishna. INFER: INtermediate representations for FuturE pRediction.

[36] Olly Styles, Tanaya Guha, and Victor Sanchez. Multiple object forecasting: Predicting future object locations in diverse environments. Technical report, 2020.

[37] Shijie Sun, Naveed Akhtar, Huansheng Song, Ajmal Mian, and Mubarak Shah. Deep Affinity Network for Multiple Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):104–119, 1 2021.

[38] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:7934–7943, 2 2019.

[39] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards Real-Time Multi-Object Tracking. Technical report, 2020.

[40] Xinshuo Weng, Ye Yuan, and Kris Kitani. PTP: Parallelized Tracking and Prediction with Graph Neural Networks and Diversity Sampling. *IEEE Robotics and Automation Letters*, 6(3):4640–4647, 3 2020.

[41] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. Technical report, 2018.

[42] Tong Xiao, Shuang Li, Bochao Wang, Liang Lin, and Xiaogang Wang. Joint detection and identification feature learning for person search. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 3376–3385. Institute of Electrical and Electronics Engineers Inc., 11 2017.

[43] Takuma Yagi, Karttikeya Mangalam, Ryo Yonetani, and Yoichi Sato. Future Person Localization in First-Person Videos. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7593–7602, 2018.

[44] Bo Yang, Chang Huang, and Ram Nevatia. Learning affinities and dependencies for multi-target tracking using a crf model. In *CVPR 2011*, pages 1233–1240. IEEE, 2011.

[45] Yu Yao, Mingze Xu, Chiho Choi, David J. Crandall, Ella M. Atkins, and Behzad Dariush. Egocentric vision-based future vehicle localization for intelligent driving assistance systems. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2019-May, pages 9711–9717, 2019.

[46] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. CityPersons: A diverse dataset for pedestrian detection. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 4457–4465. Institute of Electrical and Electronics Engineers Inc., 11 2017.

[47] Yang Zhang, Hao Sheng, Yubin Wu, Shuai Wang, Weifeng Lyu, Wei Ke, and Zhang Xiong. Long-Term Tracking with Deep Tracklet Association. *IEEE Transactions on Image Processing*, 29:6694–6706, 2020.

[48] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. FairMOT: On the Fairness of Detection and Re-identification in Multiple Object Tracking. Technical report, 2021.

[49] Liang Zheng, Hengheng Zhang, Shaoyan Sun, Manmohan Chandraker, Yi Yang, and Qi Tian. Person re-identification in theWild. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 3346–3355. Institute of Electrical and Electronics Engineers Inc., 11 2017.

[50] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as Points. Technical report, 2019.