# MULTI-OBJECT TRACKING FROM THE CLASSICS TO THE MODERN

A Dissertation Proposal
Presented to
The Academic Faculty

By

Chanho Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2020

# MULTI-OBJECT TRACKING FROM THE CLASSICS TO THE MODERN

To my parents.

# ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Prof. James Rehg, for introducing me to computer vision research and providing continuous support. Over the course of my Ph.D. program, he has inspired me by always being very sharp and precise in his thinking as a researcher. It has been a great joy and honor to learn how to become an independent researcher from him. This dissertation would not have been completed without his long time support and encouragement.

I would also like to thank Prof. Fuxin Li who has collaborated with me from the very early stages of my Ph.D. studies. He has been my go-to-person whenever I have had any questions pertaining to my daily research tasks. He has always patiently helped me learn and has also helped me focus on the important matters in my research. I appreciate his long time support which has continued even after he left Georgia Tech.

I would also like to thank my committee members, Prof. Mark Clements, Prof. Patricio Vela, Prof. James Hays, and Prof. Bernt Schiele, for joining my committee and providing me with invaluable feedback during my Ph.D. proposal and dissertation defense.

I would like to thank all of my awesome colleagues whom I have met in the lab. Specially, I would like to thank Arri who, as a senior student, helped my research a lot when I first joined the lab. I would like to thank Ahmad who made many nights in the lab much more enjoyable with his humorous jokes and interesting research discussions. I would like to thank Hyeokhyen for introducing me to motorsports, a new exciting interest. I would like to thank Audrey for helping me write and speak better in English by answering a lot of my random English questions.

I would like to thank Sangkyu, Kimoon, Eunhwan, and Daewon for their friendship and all the good memories that we created while traveling and studying together at Georgia Tech. I would like to thank Sehoon and Woohyuk who always welcomed me and made time for me whenever I visited Korea. I would like to thank Hansol, Kyungsu, Jinho, and Hyojung for their friendship and all the good memories that we created in a college together.

I would like to thank Eric, Alima, Scott, Rachel, Gordon, Brandi, India, and Donna for their friendship and kindness. Thanks to them, I have many good memories associated with Atlanta, which has become my second hometown. I will never forget what they have done for me.

Finally, I would like to thank my parents for their endless support. Mom and Dad, this dissertation is not just my work, but also yours. Thank you for always being there for me and supporting every decision that I make. I wouldn't have been able to complete my Ph.D. studies without your sacrifice and love. I owe you everything for this. Thank you.

## TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

Visual object tracking is one of the computer vision problems that has been researched extensively over the past several decades. Many computer vision applications, such as robotics, autonomous driving, and video surveillance, require the capability to track multiple objects in videos. The most popular solution approach to tracking multiple objects follows the tracking-by-detection paradigm in which the problem of tracking is divided into object detection and data association. In data association, track proposals are often generated by extending the object tracks from the previous frame with new detections in the current frame. The association algorithm then utilizes a track scorer or classifier in evaluating track proposals in order to estimate the correspondence between the object detections and object tracks.

The goal of this dissertation is to design a track scorer and classifier that accurately evaluates track proposals that are generated during the association step. In this dissertation, I present novel track scorers and track classifiers that make a prediction based on long-term object motion and appearance cues and demonstrate its effectiveness in tracking by utilizing them within existing data association frameworks. First, I present an online learning algorithm that can efficiently train a track scorer based on a long-term appearance model for the classical Multiple Hypothesis Tracking (MHT) framework. I show that the classical MHT framework achieves competitive tracking performance even in modern tracking settings in which strong object detector and strong appearance models are available. Second, I present a novel Bilinear LSTM model as a deep, long-term appearance model which is a basis for an end-to-end learned track classifier. The architectural design of Bilinear LSTM is inspired by insights drawn from the classical recursive least squares framework. I incorporate this track classifier into the classical MHT framework in order to demonstrate its effectiveness in object tracking. Third, I present a novel multi-track pooling module that enables the Bilinear LSTM-based track classifier to simultaneously consider all the objects in the scene in order to better handle appearance ambiguities between different objects. I utilize this track classifier in a simple, greedy data association algorithm and achieve real-time, state-of-the-art tracking performance. I evaluate the proposed methods in this dissertation on public multi-object tracking datasets that capture challenging object tracking scenarios in urban areas.

# CHAPTER 1
# INTRODUCTION

Visual object tracking is the area of research in computer vision where an algorithm is aimed to detect and track objects of interest in video. It has a long history in computer vision with earliest works dating back to the 80's [1] due to its importance in understanding dynamic objects in videos. It has many important applications in various domains such as surveillance [2], representation learning [3, 4], scene understanding [5, 6], and autonomous driving [7]. For example, in autonomous driving, an autonomous driving agent needs to be able to detect and follow other nearby objects in the real world coordinate system in order to safely navigate on the road. In a surveillance setting, object tracking can be performed in surveillance videos to detect and follow suspicious individuals or activities in the scene.

In general, there are three types of problems within visual object tracking. Firstly, when object classes of interest are known in advance, one can train an object detector that identifies and localizes objects of interest in images. A visual tracking algorithm then takes in object detections as input and assign an object ID number to each of the detections such that the assigned object ID is consistent over time for each object. This assignment process is also called data association, and a solution approach that solves tracking in these two steps (i.e. detection and data association) is called a tracking-by-detection approach. Designing effective tracking-by-detection algorithms has been the most active research area in Multi-Object Tracking (MOT) for the past several decades [8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. Secondly, when a user defines an object of interest in the first frame of video by drawing a bounding box, a visual tracking algorithm takes in the bounding box as input and tracks that object by detecting it in the following video frames. In this setting, the object classes of interest are assumed to be not known in advance so a pre-trained object detector is not available. Thus, the goal is to build and maintain a good object representation for the target object which is robust to appearance changes using the appearance information that the tracker obtains during tracking. This type of problem has been researched in the context of single object tracking [18, 19, 20, 21]. Lastly, all moving objects within videos regardless of the object classes are assumed to be objects of interest so a visual tracking algorithm identifies them and track them over time [22, 23, 24]. In this dissertation, I focus on solving the first type of visual tracking problem.

## 1.1 Objective

The goal of this dissertation is to develop novel discriminative track scorers and classifiers that are trained online (i.e. during tracking) or trained offline (i.e. before tracking) to identify correct or wrong track proposals in data association. Such a track scorer or classifier enables a data association algorithm to select a set of tracks among the track proposals that can explain the observed scene. Figure 1.1 shows a general pipeline for tracking-by-detection approaches. In the tracking-by-detection setting, it is assumed that an object detector is available for objects of interest. The tracker takes the output of the detector as input and generates track proposals by grouping the input detections into tracks over time. These track proposals are then either ranked by a track scorer or accepted or rejected by a track classifier. The best set of track proposals are selected by a data associaiton algorithm such as the Multiple Hypothesis Tracking (MHT) algorithm, the Hungarian algorithm or the greedy association algorithm. Traditionally, the track scorer or classifier were mostly hand-engineered via feature engineering, but recent developments in deep learning have enabled us to learn such a track scorer/classifier from the data in an end-to-end fashion [25, 26, 27]. Strong track scorers or classifiers based on deep neural networks have allowed us to achieve better tracking performance with a simpler tracking pipeline.

Designing an effective track classifier is challenging due to the following factors. First, object appearances change over time due to lighting changes, occlusion, and changes in

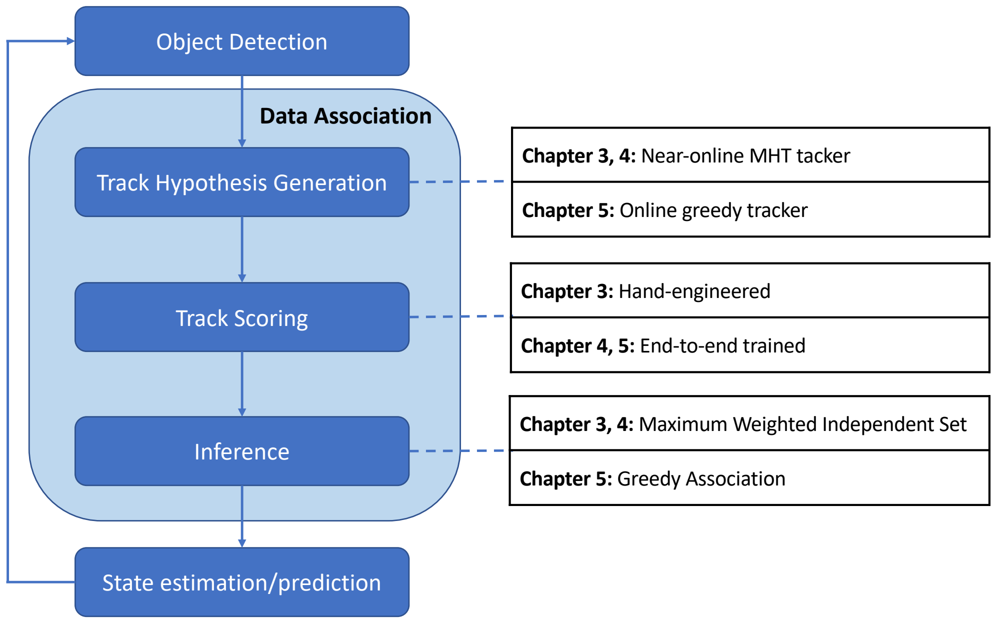

Object Detection
Data Association
Track Hypothesis Generation
Track Scoring
Inference
State estimation/prediction
**Chapter 3, 4:** Near-online MHT tacker
**Chapter 5:** Online greedy tracker
**Chapter 3:** Hand-engineered
**Chapter 4, 5:** End-to-end trained
**Chapter 3, 4:** Maximum Weighted Independent Set
**Chapter 5:** Greedy Association

Figure 1.1. An overview of a multi-object tracking algorithm

object poses or shapes. Thus, the track scorer or classifier needs to be based on an object appearance representation which is robust to such appearance changes while still being able to discriminate the target object from other objects in the scene. Thus, object appearance modeling in which such an appearance representation is built and updated over time is crucial in visual object tracking. Object motion provides important cues for data association as well since object motion is generally constrained by other objects in the scene, the environment, and the object class. Thus, the track classifier also needs to be based on an object motion representation that reflects realistic object movements in the scene. Thus, object motion modeling in which such a motion representation is built and updated over time is another key to success in designing a track scorer or classifier. In this thesis, I explore different methods for building effective object appearance and motion models for a track proposal scorer and classifier and demonstrate its effectiveness on public MOT datasets that capture challenging object tracking scenarios in urban scenes.

## 1.2 Contributions

In this thesis, I make the following contributions.

**Chapter 3 -** I present a novel incremental online learning algorithm that efficiently trains online-learned, discriminative appearance models for the classical Multiple Hypothesis Tracking (MHT) framework. I show that the proposed online learning algorithm is suitable for the MHT framework due to its computational efficiency. The online-learned appearance model utilizes all the object appearances in the scene over time and is thus cable of modeling long-term appearance information. I first show that the classical MHT framework still holds its effectiveness on the MOT 15 Challenege benchmark and then present a new MHT track scoring function based on the proposed appearance model. I demonstrate that the MHT framework with the new MHT track scoring function achieves a significant [1] performance improvement over the classical MHT framework.

**Chapter 4 -** I present an end-to-end learned track proposal classifier which is based on a long-term appearance and motion model. The appearance model in the classifier is based on a novel Bilinear Long Short-Term Memory (LSTM) model. The design of the Bilinear LSTM is inspired by the classical recursive least square framework that I introduce in Chapter 3 for learning long-term appearance models. The motion model in the classifier is based on the vanilla LSTM model. I train the track classifier in an end-to-end fashion and utilize the resulting classifier in the MHT framework. I demonstrate that the proposed approach achieves competitive performance on near-online multiple object tracking on the MOT benchmarks.

**Chapter 5 -** Unlike the appearance model presented in Chapter 3, the Bilinear LSTM-based appearance model presented in Chapter 4 stores information about the target object only in its memory. In Chapter 5, I present a novel multi-track pooling (MTP) module for the Bilinear LSTM-based track classifier in order to solve the problem of simultaneously

[1] Significance tests are not part of the current MOT evaluation protocol, so the term ”significant” used in this dissertation does not mean ”statistically significant.”

considering all tracks during memory updating. The multi-track pooling module enables the classifier to adaptively adjust its prediction based on all the objects' appearances in the scene in order to better handle appearance ambiguities between different objects. I also present a new training strategy that is adapted to train the proposed multi-track pooling module. I utilize the track classifier in a simple, greedy data association algorithm and demonstrate real-time, state-of-the-art performance on the MOT benchmarks.

## 1.3 List of Publications

This dissertation is based on the following publications.

- **Chanho Kim**, Fuxin Li, Mazen Alotaibi, James M. Rehg. Discriminative Appearance Modeling with Multi-track Pooling for Real-time Multi-object Tracking. **Under review for CVPR** 2021.
- **Chanho Kim**, Fuxin Li, James M. Rehg. Multi-object Tracking with Neural Gating Using Bilinear LSTM. **ECCV** 2018.
- **Chanho Kim**, Fuxin Li, Arridhana Ciptadi, James M. Rehg. Multiple Hypothesis Tracking Revisited. **ICCV** 2015.

# CHAPTER 2
# LITERATURE SURVEY

## 2.1 Classical and Graph-based Approaches

Network flow-based methods [8, 9, 10, 11] have recently become a standard approach to visual multi-target tracking due to their computational efficiency and optimality. In recent years, efficient inference algorithms to find the globally optimal solution [10, 9] or approximate solutions [8] have been introduced. However, the benefits of flow-based approaches come with a costly restriction: the cost function can only contain unary and pairwise terms. Pairwise costs are very restrictive in representing motion and appearance. In particular, it is difficult to represent even a linear motion model with those terms.

An alternative is to define pairwise costs between tracklets – short object tracks that can be computed reliably [28, 29, 30, 31]. Unfortunately the availability of reliable tracklets cannot be guaranteed, and any mistakes propagate to the final solution. In Brendel et al. [31], data association for tracklets is solved using the Maximum Weighted Independent Set (MWIS) method. I also adopt MWIS in Chapter 3 and 4, but follow the classical formulation in [15] and focus on the incorporation of appearance modeling.

Collins [32] showed mathematically that the multidimensional assignment problem is a more complete representation of the multi-target tracking problem than the network flow formulation. Unlike network flow, there is no limitation in the form of the cost function, even though finding an exact solution to the multidimensional assignment problem is intractable.

Classical solutions to multidimensional assignment are MHT [12, 13, 14, 15] and Markov Chain Monte Carlo (MCMC) data association [16, 17]. While MCMC provides asymptotic guarantees, MHT has the potential to explore the solution space more thoroughly, but has traditionally been hindered by the exponential growth in the number of hypotheses and had to resort to aggressive pruning strategies, such as propagating only the $M$-best hypotheses [13]. In Chapter 3, I show that this limitation can be addressed through discriminative appearance modeling.

Andriyenko [33] proposed a discrete-continuous optimization method to jointly solve trajectory estimation and data association. Trajectory estimation is solved by spline fitting and data association is solved via MRF inference. These two steps are alternated until convergence. Segal [34] proposed a related approach based on a message passing algorithm. These methods are similar to MHT in the sense that they directly optimize a global energy with no guarantees on solution quality. But in practice, MHT is more effective in identifying high quality solutions.

There have been a significant number of prior works that exploit appearance information to solve data association. In the network flow-based method, the pairwise terms can be weighted by offline trained appearance templates [35] or a simple distance metric between appearance features [10]. However, these methods have limited capability to model the complex appearance changes of a target. In [14], a simple fixed appearance model is incorporated into a standard MHT framework. In contrast, I show that MHT can be extended to include online learned discriminative appearance models for each track hypothesis in Chapter 3.

Online discriminative appearance modeling is a standard method for addressing appearance variation [36]. In tracklet association, several works [37, 38, 39, 40] train discriminative appearance models of tracklets in order to design a better affinity score function. However, these approaches still share the limitations of the tracklet approach. Other works [41, 42] train a classifier for each target and use the classification score for greedy data association or particle filtering. These methods only keep one online learned model for each target, while my method trains multiple online appearance models via multiple track hypotheses, which is more robust to model drift.

## 2.2 Deep Learning-based Approaches

A deep neural network-based tracking approach has been shown to be very effective in learning features and track scoring functions, and currently produces the best performance on standard benchmarks. Various approaches utilizing convolutional networks [43, 44, 45] or recurrent neural networks [25, 26, 27] for handling multi-object tracking data (i.e. sequential data) have been proposed. In multi-object tracking, the tracker maintains in its memory the appearance and motion information for each object in the scene. In deep learning-based approaches, the tracker utilizes deep neural networks to obtain the memory representation for each target object. The memory is then utilized for finding matches between tracks and detections, and is updated based on the matching results. In this section, I review two types of deep learning-based approaches. The first type of approach focuses on modeling each target object in isolation and thus lacks the capability to consider all the objects in the scene simultaneously. The work in Chapter 4 belongs to this line of approach. The second type of approach focuses on updating the target object memory representations based on other objects in the scene by utilizing object interactions or jointly considering other objects' appearances during memory updating. The work in Chapter 5 is related to this line of approach. The comparison between the proposed works and the related works is also summarized in Table 2.1.

### 2.2.1 Approaches that utilize the target object only during model updating

The prior work that is closest to my work in Chapter 4 uses RNNs as a track proposal classifier in the Markov Decision Process (MDP) framework [25]. Three different RNNs that handle appearance, motion, and social information are trained separately for track proposal classification and then combined for joint reasoning over multiple cues to achieve the best performance. My method is different from this approach both in terms of the network architecture and training sequence generation from ground truth tracks. Also, I present the first incorporation of deep learned track model into an MHT framework in Chapter 4.

Other recent approaches [44, 43] adopt siamese networks that learn the matching function for a pair of images. The network is trained for the binary classification problem where the binary output represents whether or not the image pair comes from the same object. The matching function can be utilized in a tracking framework to replace any previous matching function. Approaches in this category are limited to only modeling the information between a pair of the detections, whereas my approach can model the interaction between a track and a detection, thereby exploiting long-term appearance and motion information.

Milan et al. [46] presented a deep learning framework that solves the multi-object tracking problem in an end-to-end trainable network. Unlike my approach, they attempted to solve state estimation and data association jointly in one framework. While this was highly innovative, an advantage of MHT is the ability to use highly optimized combinatoric solvers.

RNN has been applied in single-object tracking [47, 48], however multi-target tracking is a more challenging problem due to the amount of occlusion and problem of ID switches, which is much more likely to happen in a multi-object setting.

### 2.2.2 Approaches that utilize all the objects in the scene during model updating

Two groups of prior works have explored the incorporation of positive and negative samples during on-the-fly testing, and these are the closest related works to my work presented in Chapter 5. One line of work incorporates these samples by fine-tuning a pretrained CNN using positive examples (target object) and negative examples (background in [18], other objects in [54]) during testing. While these approaches share my interest in utilizing scene-specific information during tracking, the need to fine-tune during testing adds an additional source of complexity and is a barrier to efficient online performance. In contrast, my model automatically adjusts its prediction based on scene-specific information without the need

Table 2.1. Comparison between the proposed approaches and the related works

| | Association | Sequence Length | Network | Input | Output |
|---|---|---|---|---|---|
| [25] | MDP, Hungarian | 6 frames | LSTM | A track-detection pair and an occupancy map | Binary class |
| [26] | Hungarian | 9 frames | GRU | A track-detection pair | Parameters of AR model |
| [49] | A variant of MHT | Training: up to 3 seconds<br>Testing: up to 6 seconds | Bi-directional LSTM | A pair of tracks | Approximated IDF1 |
| [50] | Hungarian | 10 frames | Relational Networks | A track-detection pair and all detections up to the current frame | Binary class |
| [51] | ECO, Greedy | 8 frames | Bi-directional LSTM | A track-detection pair | Binary class |
| [52] | Greedy, Post-processing | 25 frames<br>(=12 message passing steps) | GCN | All detections in selected frames | Binary class |
| [53] | Hungarian | 5 frames (motion)<br>2 frames (appearance) | GCN | All tracks in the previous frame and all detections in the current frame | Binary class |
| Ch. 4 | MHT | Training: up to 40 frames<br>Testing: track length | Bilinear LSTM (Appearance)<br>LSTM (Motion) | A track-detection pair | Binary class |
| Ch. 5 | Greedy | Track length | Bilinear LSTM with MTP (Appearance)<br>LSTM (Motion) | All tracks in the previous frame and a detection in the current frame | Binary class |

for fine-tuning. The second line of work uses relational networks or graph convolutional networks (GCNs) to incorporate the appearance of other detections in the same frame [50] and in neighboring frames [52] when computing the appearance features of each detection. However, these works operate in a batch setting where the entire video is available, whereas my approach is online. In addition, my multi-track pooling method is significantly simpler and faster than graph convolutional networks, which require multiple iterations of message passing.

I utilize the Bilinear LSTM architecture in Chapter 4 in developing the matching approach. I extend beyond this work in multiple ways, the primary difference being the introduction of a novel multi-object pooling approach which utilizes appearance information across tracks to significantly improve data association performance. I demonstrate that this makes it feasible to use a much simpler and more cost-effective matching algorithm following track scoring, achieving real-time multi-object tracking.

[25] also presented an LSTM-based track proposal classifier that integrates motion, appearance, and interaction cues. The primary difference between my work and this approach is that their interaction cues are the relative locations of other tracks with respect to the target object, whereas my model takes the appearances of other tracks in the scene into account when making a prediction.

Among the previous works that were not based on deep learning, [55] and [6] exploited interactions between tracks in solving the multi-object tracking problem. [55] incorporated a social behavior model into their tracking algorithm. The social behavior model is based on the assumption that each person moves in such a way as to avoid collisions with other people. [6] incorporated high-level human activity cues into their tracking algorithm by exploiting the fact that human activities can influence how people move in the scene. In contrast to these works, I focus on incorporating multiple appearances from all tracks into the model, in order to make it more discriminative.

# CHAPTER 3
# MULTIPLE HYPOTHESIS TRACKING REVISITED

In this chapter, I revisit the classical multiple hypotheses tracking (MHT) algorithm in a tracking-by-detection framework. The success of MHT largely depends on the ability to maintain a small list of potential hypotheses, which can be facilitated with the accurate object detectors that are currently available. I demonstrate that a classical MHT implementation from the 90's can still achieve competitive performance on standard benchmark datasets. In order to further utilize the strength of MHT in exploiting higher-order information, I introduce a method for training online appearance models for each track hypothesis. I show that appearance models can be learned efficiently via a regularized least squares framework, requiring only a few extra operations for each hypothesis branch. The work presented in this chapter has been published as [56].

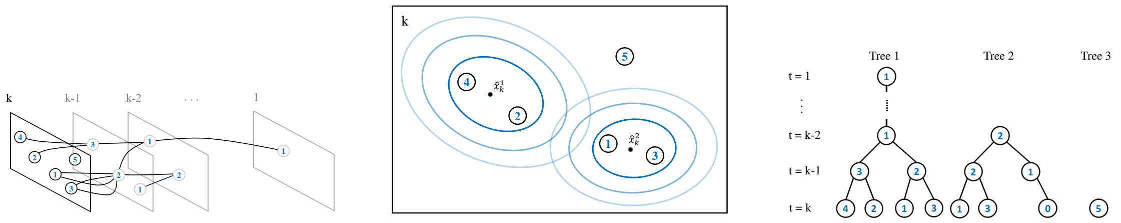
## 3.1 Introduction

Multiple Hypotheses Tracking (MHT) is one of the earliest successful algorithms for visual tracking. Originally proposed in 1979 by Reid [12], it builds a tree of potential track hypotheses for each candidate target, thereby providing a systematic solution to the data association problem. The likelihood of each track is calculated and the most likely combination of tracks is selected. Importantly, MHT is ideally suited to exploiting higher-order information such as long-term motion and appearance models, since the entire track hypothesis can be considered when computing the likelihood.

MHT has been popular in the radar target tracking community [57]. However, in visual tracking problems, it is generally considered to be slow and memory intensive, requiring many pruning tricks to be practical. While there was considerable interest in MHT in the vision community during the 90s, for the past 15 years it has not been a mainstream approach for tracking, and rarely appears as a baseline in tracking evaluations. MHT is in essence a breadth-first search algorithm, hence its performance strongly depends on the ability to prune branches in the search tree quickly and reliably, in order to keep the number of track hypotheses manageable. In the early work on MHT for visual tracking [13], target detectors were unreliable and motion models had limited utility, leading to high combinatoric growth of the search space and the need for efficient pruning methods.

We argue that the MHT approach is well-suited to the current visual tracking context. Modern advances in tracking-by-detection and the development of effective feature representations for object appearance have created new opportunities for the MHT method. First, we demonstrate that a modern formulation of a standard motion-based MHT approach gives comparable performance to state-of-the-art methods on popular tracking datasets. Second, and more importantly, we show that MHT can easily exploit high-order appearance information which has been difficult to incorporate into other tracking frameworks based on unary and pairwise energies. We present a novel MHT method which incorporates long-term appearance modeling, using features from deep convolutional neural networks [58, 59]. The appearance models are trained online for each track hypothesis on all detections from the entire history of the track. We utilize online regularized least squares [60] to achieve high efficiency. In our formulation, *the computational cost of training the appearance models has little dependency on the number of hypothesis branches,* making it extremely suitable for the MHT approach.

Our experimental results demonstrate that our scoring function, which combines motion and appearance, is highly effective in pruning the hypothesis space efficiently and accurately. Using our trained appearance model, we are able to cut the effective number of branches in each frame to about $50\%$ of all branches (Sec. 3.4.1). This enables us to make less restrictive assumptions on motion and explore a larger space of hypotheses. This also makes MHT less sensitive to parameter choices and heuristics (Fig. 3.3). Experiments on the PETS and the recent MOT challenge illustrate the state-of-the-art performance of our



(a) Tracks in Video Frames (b) Gating (c) Track Trees

Figure 3.1. Illustration of MHT. (a) Track hypotheses after the gating test at time $k$. Only a subset of track hypotheses is visualized here for simplicity. (b) Example gating areas for two track hypotheses with different thresholds $d_{\text{th}}$. (c) The corresponding track trees. Each tree node is associated with an observation in (a).

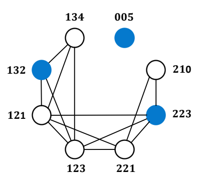
approach.

## 3.2 Multiple Hypotheses Tracking

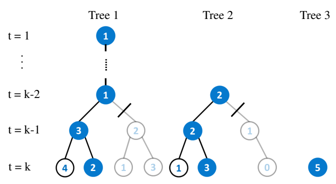
We adopt a tracking-by-detection framework such that our observations are localized bounding boxes obtained from an object detection algorithm. Let $k$ denote the most recent frame and $M_k$ denote the number of object detections (i.e. observations) in that frame. For a given track, let $i_k$ denote the observation which is selected at frame $k$, where $i_k \in \{0, 1, \ldots, M_k\}$. The observation sequence $i_1, i_2, \ldots, i_k$ then defines a **track hypothesis** over $k$ frames. Note that the dummy assignment $i_t = 0$ represents the case of a missing observation (due to occlusion or a false negative).[1] Let the binary variable $z_{i_1 i_2 \ldots i_k}$ denote whether or not a track hypothesis is selected in the final solution. A **global hypothesis** is a set of track hypotheses that are not in conflict, i.e. that do not share any measurements at any time.

A key strategy in MHT is to delay data association decisions by keeping multiple hypotheses active until data association ambiguities are resolved. MHT maintains multiple track trees, and each tree represents all of the hypotheses that originate from a single observation (Fig. 3.1c). At each frame, the track trees are updated from observations and each track in the tree is scored. The best set of non-conflicting tracks (the best global hypothesis)

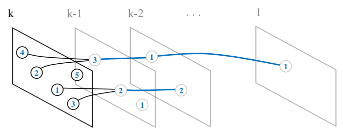
[1]For notational convenience, observation sequences can be assumed to be padded with zeros so that all track hypotheses can be treated as fixed length sequences, despite their varying starting and ending times.



(a) MWIS



(b) $N$-scan Pruning



(c) Remaining Track Hypotheses

Figure 3.2. (a) An undirected graph for the example of Fig. 3.1 in which each track hypothesis is a node and an edge connects two tracks that are conflicting. The observations for each hypothesis in the last three frames are indicated. An example of the Maximum Weighted Independent Set (MWIS) is highlighted in blue. (b) An $N$-scan pruning example ($N = 2$). The branches in blue contain the global hypothesis at frame $k$. Pruning at $t = k-2$ removes all branches that are far from the global hypothesis. (c) Track hypotheses after the pruning. The trajectories in blue represent the finalized measurement associations.

can then be found by solving a maximum weighted independent set problem (Fig. 3.2a). Afterwards, branches that deviate too much from the global hypothesis are pruned from the trees, and the algorithm proceeds to the next frame. In the rest of this section, we will describe the approach in more detail.

### 3.2.1 Track Tree Construction and Updating

A track tree encapsulates multiple hypotheses starting from a single observation. At each frame, a new track tree is constructed for each observation, representing the possibility that this observation corresponds to a new object entering the scene.

Previously existing track trees are also updated with observations from the current frame. Each track hypothesis is extended by appending new observations located within its gating area as its children, with each new observation spawning a separate branch. We also always spawn a separate branch with a dummy observation, in order to account for missing detection.

## 3.2.2 Gating

Based on the motion estimates, a gating area is predicted for each track hypothesis which specifies where the next observation of the track is expected to appear.

Let $\mathbf{x}_k^l$ be the random variable that represents the likely location of the $l^{\text{th}}$ track at time $k$. The variable $\mathbf{x}_k^l$ is assumed to be normally distributed with mean $\hat{\mathbf{x}}_k^l$ and covariance $\Sigma_k^l$ determined by Kalman filtering. The decision whether to update a particular trajectory with a new observation $i_k$ is made based on the Mahalanobis distance $d^2$ between the observation location $\mathbf{y}_{i_k}$ and the predicted location $\hat{\mathbf{x}}_k^l$:

$$d^2 = (\hat{\mathbf{x}}_k^l - \mathbf{y}_{i_k})^\top (\Sigma_k^l)^{-1} (\hat{\mathbf{x}}_k^l - \mathbf{y}_{i_k}) \leq d_{\text{th}}. \tag{3.1}$$

The distance threshold $d_{\text{th}}$ determines the size of the gating area (see Fig. 3.1b).

## 3.2.3 Track Scoring

Each track hypothesis is associated with a track score. The $l^{\text{th}}$ track's score at frame $k$ is defined as follows:

$$S^l(k) = w_{\text{mot}} S_{\text{mot}}^l(k) + w_{\text{app}} S_{\text{app}}^l(k) \tag{3.2}$$

where $S_{\text{mot}}^l(k)$ and $S_{\text{app}}^l(k)$ are the motion and appearance scores, and $w_{\text{mot}}$ and $w_{\text{app}}$ are the weights that control the contribution of the location measurement $\mathbf{y}_{i_k}$ and the appearance measurement $X_{i_k}$ to the track score, respectively.

Following the original formulation [57], we use the log likelihood ratio (LLR) between the target hypothesis and the null hypothesis as the motion score. The target hypothesis assumes that the sequence of observations comes from the same target, and the null hypothesis assumes that the sequence of observations comes from the background. Then the

$l^{\text{th}}$ track's motion score at time $k$ is defined as:

$$S^l_{\text{mot}}(k) = \ln \frac{p(\mathbf{y}_{i_{1:k}}|i_{1:k} \subseteq T_l)}{p(\mathbf{y}_{i_{1:k}}|i_{1:k} \subseteq \phi)} \tag{3.3}$$

where we use the notation $i_{1:k}$ for the sequence of observations $i_1, i_2, ..., i_k$. We denote by $i_{1:k} \subseteq T_l$ the target hypothesis that the observation sequence comes from the $l^{\text{th}}$ track and we denote the null hypothesis by $i_{1:k} \subseteq \phi$. The likelihood factorizes as:

$$\frac{p(\mathbf{y}_{i_{1:k}}|i_{1:k} \subseteq T_l)}{p(\mathbf{y}_{i_{1:k}}|i_{1:k} \subseteq \phi)} = \frac{\prod_{t=1}^{k} p(\mathbf{y}_{i_t}|\mathbf{y}_{i_{1:t-1}}, i_{1:t} \subseteq T_l)}{\prod_{t=1}^{k} p(\mathbf{y}_{i_t}|i_t \subseteq \phi)} \tag{3.4}$$

where we assume that measurements are conditionally independent under the null hypothesis.

The likelihood for each location measurement at time $t$ under the target hypothesis is assumed to be Gaussian. The mean $\hat{\mathbf{x}}^l_t$ and the covariance $\Sigma^l_t$ are estimated by a Kalman filter for the measurements $\mathbf{y}_{i_{1:t-1}}$. The likelihood under the null hypothesis is assumed to be uniform. The factored likelihood terms at time $t$ are then written as:

$$\begin{aligned} p(\mathbf{y}_{i_t}|\mathbf{y}_{i_{1:t-1}}, i_{1:t} \subseteq T_l) &= \mathcal{N}(\mathbf{y}_{i_t}; \hat{\mathbf{x}}^l_t, \Sigma^l_t), \\ p(\mathbf{y}_{i_t}|i_t \subseteq \phi) &= 1/V \end{aligned} \tag{3.5}$$

where $V$ is the measurement space [57, 13], which is the image area or the area of the ground plane for $2.5D$ tracking.

The appearance track score is defined as:

$$S^l_{\text{app}}(k) = \ln \frac{p(X_{i_{1:k}}|i_{1:k} \subseteq T_l)}{p(X_{i_{1:k}}|i_{1:k} \subseteq \phi)} = \ln \frac{p(i_{1:k} \subseteq T_l|X_{i_{1:k}})}{p(i_{1:k} \subseteq \phi|X_{i_{1:k}})} \tag{3.6}$$

where we obtain the posterior LLR under the assumption of equal priors. The posterior ratio factorizes as:

$$\frac{p(i_{1:k} \subseteq T_l | X_{i_{1:k}})}{p(i_{1:k} \subseteq \phi | X_{i_{1:k}})} = \frac{\prod_{t=1}^{k} p(i_t \subseteq T_l | i_{1:t-1} \subseteq T_l, X_{i_{1:t}})}{\prod_{t=1}^{k} p(i_t \subseteq \phi | X_{i_t})} \tag{3.7}$$

where we utilize $\{i_{1:k} \subseteq T_l\} = \bigcup_{t=1}^{k} \{i_t \subseteq T_l\}$ for the factorization. We assume that $i_t \subseteq T_l$ is conditionally independent of future measurements $X_{i_{t+1:k}}$ and the $i_t \subseteq \phi$ hypotheses are independent given the current measurement $X_{i_t}$.

Each term in the factored posterior comes from the online learned classifier (Sec. 3.3) at time $t$. Given prior observations $i_{1:t-1}$, we define the posterior of the event that observation $i_t$ is in the $l^{\text{th}}$ track as:

$$p(i_t \subseteq T_l | i_{1:t-1} \subseteq T_l, X_{i_{1:t}}) = \frac{e^{F(X_{i_t})}}{e^{F(X_{i_t})} + e^{-F(X_{i_t})}} \tag{3.8}$$

where $F(\cdot)$ is the classification score for the appearance features $X_{i_t}$ and the classifier weights are learned from $X_{i_{1:t-1}}$. We utilize the constant probability $c_1$ for the posterior of the background (null) hypothesis.

$$p(i_t \subseteq \phi | X_{i_t}) = c_1 \tag{3.9}$$

The track score expresses whether a track hypothesis is more likely to be a true target ($S^l(k) > 0$) or false alarm ($S^l(k) < 0$). The score can be computed recursively [57]:

$$S^l(k) = \quad S^l(k-1) + \Delta S^l(k), \tag{3.10}$$

$$\Delta S^l(k) = \begin{cases} \ln \frac{1-P_D}{1-P_{FA}} \approx \ln(1-P_D), & \text{if } i_k = 0 \\ w_{\text{mot}} \Delta S^l_{\text{mot}}(k) + w_{\text{app}} \Delta S^l_{\text{app}}(k), & \text{otherwise} \end{cases} \tag{3.11}$$

where $P_D$ and $P_{FA}$ (assumed to be very small) are the probabilities of detection and false alarm, respectively. $\Delta S^l_{\text{mot}}(k)$ and $\Delta S^l_{\text{app}}(k)$ are the increments of the track motion score and the track appearance score at time $k$ and are calculated using Eqs. (3.5), (3.8), and (3.9) as:

$$\begin{aligned} \Delta S^l_{\text{mot}}(k) &= \ln \frac{V}{2\pi} - \frac{1}{2} \ln |\Sigma^l_k| - \frac{d^2}{2}, \\ \Delta S^l_{\text{app}}(k) &= -\ln\left(1 + e^{-2F(X_{i_k})}\right) - \ln c_1. \end{aligned} \tag{3.12}$$

The score update continues as long as the track hypothesis is updated with detections. A track hypothesis which is assigned dummy observations for $N_{\text{miss}}$ consecutive frames is deleted from the hypothesis space.

### 3.2.4 Global Hypothesis Formation

Given the set of trees that contains all trajectory hypotheses for all targets, we want to determine the most likely combination of object tracks at frame $k$. This can be formulated as the following $k$-dimensional assignment problem:

$$\begin{aligned} &\max_{\mathbf{z}} \sum_{i_1=0}^{M_1} \sum_{i_2=0}^{M_2} \cdots \sum_{i_k=0}^{M_k} s_{i_1 i_2 \ldots i_k} z_{i_1 i_2 \ldots i_k} \\ &\text{subject to} \sum_{i_1=0}^{M_1} \cdots \sum_{i_{u-1}=0}^{M_{u-1}} \sum_{i_{u+1}=0}^{M_{u+1}} \cdots \sum_{i_k=0}^{M_k} z_{i_1 i_2 \ldots i_u \ldots i_k} = 1 \\ &\text{for} \quad i_u = 1,2,\ldots,M_u \quad \text{and} \quad u = 1,2,\ldots,k \end{aligned} \tag{3.13}$$

where we have one constraint for each observation $i_u$, which ensures that it is assigned to a unique track. Each track is associated with its binary variable $z_{i_1 i_2 \ldots i_k}$ and track score $s_{i_1 i_2 \ldots i_k}$ which is calculated by Eq. (4.1). Thus, the objective function in Eq. (3.13) represents the total score of the tracks in the global hypothesis. This optimization problem is known to be NP-hard when $k$ is greater than 2.

Following [15], the task of finding the most likely set of tracks can be formulated as a Maximum Weighted Independent Set (MWIS) problem. This problem was shown in [15] to be equivalent to the multidimensional assignment problem (3.13) in the context of MHT.

An undirected graph $G = (V, E)$ is constructed by assigning each track hypothesis $T_l$ to a graph vertex $x_l \in V$ (see Fig. 3.2a). Note that the number of track hypotheses needs to be controlled by track pruning (Sec. 3.2.5) at every frame in order to avoid the exponential growth of the graph size. Each vertex has a weight $w_l$ that corresponds to its track score $S^l(k)$. An edge $(l, j) \in E$ connects two vertices $x_l$ and $x_j$ if the two tracks cannot co-exist due to shared observations at any frame. An independent set is a set of vertices with no edges in common. Thus, finding the maximum weight independent set is equivalent to finding the set of compatible tracks that maximizes the total track score. This leads to the following discrete optimization problem:

$$\begin{aligned} &\max_{\mathbf{x}} \sum_l w_l x_l \\ \text{s.t.} \;\; &x_l + x_j \leq 1, \;\; \forall (l, j) \in E, \;\; x_l \in \{0, 1\}. \end{aligned} \tag{3.14}$$

We utilize either an exact algorithm [61] or an approximate algorithm [62] to solve the MWIS optimization problem, depending on its hardness (as determined by the number of nodes and the graph density).

### 3.2.5 Track Tree Pruning

Pruning is an essential step for MHT due to the exponential increase in the number of track hypotheses over time. We adopt the standard $N$-scan pruning approach. First, we identify the tree branches that contain the object tracks within the global hypothesis obtained from Eq. (3.14). Then for each of the selected branches, we trace back to the node at frame $k - N$ and prune the subtrees that diverge from the selected branch at that node (see Fig. 3.2b). In other words, we consolidate the data association decisions for old observations up to frame $k - (N - 1)$. The underlying assumption is that the ambiguities in data association for frames 1 to $k - N$ can be resolved after looking ahead for a window of $N$ frames [13]. A larger $N$ implies a larger window hence the solution can be more accurate, but makes the running time longer. After pruning, track trees that do not contain any track in the global hypothesis will be deleted.

Besides $N$-scan pruning, we also prune track trees that have grown too large. If at any specific time the number of branches in a track tree is more than a threshold $B_{\text{th}}$, then we prune the track tree to retain only the top $B_{\text{th}}$ branches based on its track score.

When we use MHT-DAM (see Table 3.1), the appearance model enables us to perform additional branch pruning. This enables us to explore a larger gating area without increasing the number of track hypotheses significantly. Specifically, we set $\Delta S_{\text{app}}(t) = -\infty$, preventing the tree from spawning a branch for observation $i_t$, when its appearance score $F(X_{i_t}) < c_2$. These are the only pruning mechanisms in our MHT implementation.

## 3.3 Online Appearance Modeling

Since the data association problem is ill-posed, different sets of kinematically plausible trajectories always exist. Thus, many methods make strong assumptions on the motion model, such as linear motion or constant velocity [34, 63, 11]. However, such motion constraints are frequently invalid and can lead to poor solutions. For example, the camera can move or the target of interest may also suddenly change its direction and velocity. Thus, motion-based constraints are not very robust.

When target appearances are distinctive, taking the appearance information into account is essential to improve the accuracy of the tracking algorithm. We adopt the multi-output regularized least squares framework [60] for learning appearance models of targets in the scene. As an online learning scheme, it is less susceptible to drifting than local appearance matching, because multiple appearances from many frames are taken into account.

We first review the Multi-output Regularized Least Squares (MORLS) framework and then explain how this framework fits into MHT.

### 3.3.1 Multi-output Regularized Least Squares

Multiple linear regressors are trained and updated simultaneously in multi-output regularized least squares. At frame k, the weight vectors for the linear regressors are represented by a $d \times n$ weight matrix $\mathbf{W}_k$ where $d$ is the feature dimension and $n$ is the number of regressors being trained. Let $\mathbf{X}_k = [X_{k,1}|X_{k,2}|...|X_{k,n_k}]^\top$ be a $n_k \times d$ input matrix where $n_k$ is the number of feature vectors (i.e. detections), and $X_{k,i}$ represents the appearance features from the $i$-th training example at time $k$. Let $\mathbf{V}_k = [V_{k,1}|V_{k,2}|...|V_{k,n}]$ denote a $n_k \times n$ response matrix where $V_{k,i}$ is a $n_k \times 1$ response vector for the $i^{\text{th}}$ regressor at time $k$. When a new input matrix $\mathbf{X}_{k+1}$ is received, the response matrix $\hat{\mathbf{V}}_{k+1}$ for the new input can be predicted by $\mathbf{X}_{k+1}\mathbf{W}_k$.

The weight matrix $\mathbf{W}_k$ is learned at time $k$. Given all the training examples $(\mathbf{X}_i, \mathbf{V}_i)$ for $1 \leq i \leq k$, the weight matrix can be obtained as:

$$\min_{\mathbf{W}_k} \sum_{t=1}^{k} \|\mathbf{X}_i\mathbf{W}_k - \mathbf{V}_i\|_F^2 + \lambda\|\mathbf{W}_k\|_F^2 \tag{3.15}$$

where $\|\cdot\|_F$ is the Frobenius norm. The optimal solution is given by the following system of linear equations:

$$(\mathbf{H}_k + \lambda\mathbf{I})\mathbf{W}_k = \mathbf{C}_k \tag{3.16}$$

where $\mathbf{H}_k = \sum_{t=1}^{k} \mathbf{X}_t^\top \mathbf{X}_t$ is the covariance matrix, and $\mathbf{C}_k = \sum_{t=1}^{k} \mathbf{X}_t^\top \mathbf{V}_t$ is the correlation matrix.

The model is online because at any given time only $\mathbf{H}_k$ and $\mathbf{C}_k$ need to be stored and updated. $\mathbf{H}_k$ and $\mathbf{C}_k$ can be updated recursively via:

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \mathbf{X}_{k+1}^\top \mathbf{X}_{k+1}, \tag{3.17}$$

$$\mathbf{C}_{k+1} = \mathbf{C}_k + \mathbf{X}_{k+1}^\top \mathbf{V}_{k+1} \tag{3.18}$$

which only requires the inputs and responses at time $k+1$.

### 3.3.2 Application of MORLS to MHT

We utilize each detected bounding box as a training example. Appearance features from all detection boxes at time $k$ form the input matrix $\mathbf{X}_k$. Each tree branch (track hypothesis) is paired with a regressor which is trained with the detections from the time when the track tree was born to the current time $k$. Detections from the entire history of the track hypothesis serve as positive examples and all other detections serve as negative examples. The response for the positive example is $1$, and the responses for the negative examples are set to $-1$. Note that a classification loss function (e.g. hinge loss) will be more suitable for this problem, but then the benefits of efficient updates and an analytic globally optimal solution would be lost.

The online nature of the least squares framework makes it efficient to update multiple regressors as the track tree is extended over time. Starting from one appearance model at the root node, different appearance models will be generated as the track tree spawns different branches. $\mathbf{H}$ and $\mathbf{C}$ in the current tree layer (corresponding to the current frame) are copied into the next tree layer (next frame), and then updates according to Eqs. (3.17) and (3.18) are performed for all of the tree branches in the next tree layer. Suppose we have $\mathbf{H}_{k-1}$ and $\mathbf{C}_{k-1}$ and are branching into $n$ branches at time $k$. Note that the update of $\mathbf{H}_k$ only depends on $\mathbf{X}_k$ and is done once, no matter how many branches are spawned at time $k$. $\mathbf{C}_k$ depends on both $\mathbf{X}_k$ and $\mathbf{V}_k$. Hence, for each new tree branch $i$, one matrix-vector multiplication $\mathbf{X}_k^\top V_{k,i}$ needs to be performed. The total time complexity for computing $\mathbf{X}_k^\top \mathbf{V}_k = [\mathbf{X}_k^\top V_{k,1}|\mathbf{X}_k^\top V_{k,2}|...|\mathbf{X}_k^\top V_{k,n}]$ is then $O(dnn_k)$ which is linear in both the number of tree branches $n$ and the number of detections $n_k$.

The most time-consuming operation in training the model is updating and decomposing $\mathbf{H}$ in solving Eq. (3.16). This operation is shared among all the track trees that start at the same frame and is independent of the branches on the track trees. Thus, one can easily spawn many branches in each track tree with minimal additional computation required for appearance updating. This property is unique to tree-based MHT, where all the branches have the same ancestry. If one is training long-term appearance models using other global methods such as [64] and [17], then such computational benefits disappear, and the appearance model would need to be fully updated for each target separately, which would incur substantial computational cost.

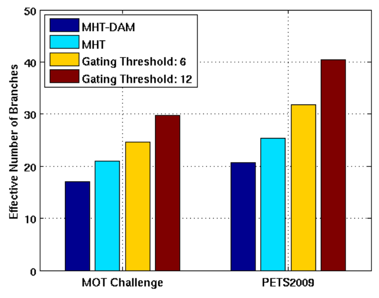
As for the appearance features, we utilize the convolutional neural network features trained on the ImageNet+PASCAL VOC dataset in [59]. We follow the protocol in [59] to extract the 4096-dimensional feature for each detection box. For better time and space complexity, a principal component analysis (PCA) is then performed to reduce the dimensionality of the features. In the experiments we take the first $256$ principal components.

## 3.4 Experiments

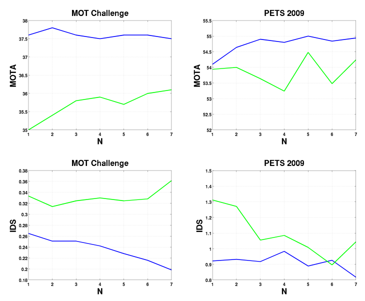
In this section we first present several experiments that show the benefits of online appearance modeling on MHT. We use 11 MOT Challenge [65] training sequences and 5 PETS 2009 [66] sequences for these experiments. These sequences cover different difficulty levels of the tracking problem. In addition to these experimental results, we also report the performance of our method on the MOT Challenge and PETS benchmarks for quantitative comparison with other tracking methods.

For performance evaluation, we follow the current evaluation protocols for visual multi-target tracking. The protocols include the multiple object tracking accuracy (MOTA) and multiple object tracking precision (MOTP) [67]. MOTA is a score which combines false positives, false negatives and identity switches (IDS) of the output trajectories. MOTP measures how well the trajectories are aligned with the ground truth trajectories in terms of the average distance between them. In addition to these metrics, the number of mostly tracked targets (MT), mostly lost targets (ML), track fragmentations (FM), and IDS are also reported. Detailed descriptions about these metrics can be found in [68].

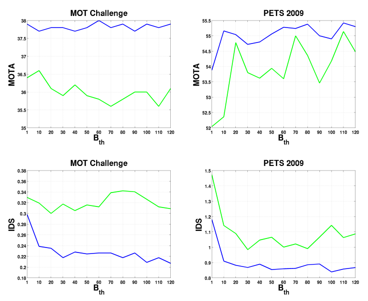
Table 3.1 shows the default parameter setting for all of the experiments in this section. In the table, our baseline method that only uses motion information is denoted as MHT. This



(a)



(b)



(c)

Figure 3.3. (a) Average effective number of branches per track tree for different pruning mechanisms. MHT-DAM uses a gating threshold $d_{\text{th}} = 12$ and MHT uses a gating threshold $d_{\text{th}} = 6$. Even with a larger gating area, the appearance model for MHT-DAM is capable of significantly reducing the number of branches. (b) Sensitivity analysis for $N$-scan parameter $N$. (c) Sensitivity analysis for the maximum number of branches $B_{th}$. The **Blue** lines are the results from MHT-DAM and the **Green** lines are the results from MHT. The first row shows the MOTA score (higher is better) and the second row shows the number of ID switches (averaged per target, lower is better) over different pruning parameters.

is a basic version of the MHT method described in Section 3.2 using only the motion score $S_{\text{mot}}(k)$. Our novel extension of MHT that incorporates online discriminative appearance modeling is denoted as MHT-DAM.

Table 3.1. Parameter Setting

| | N-scan | $B_{\text{th}}$ | $N_{\text{miss}}$ | $P_D$ | $d_{\text{th}}$ | $w_{\text{mot}}, w_{\text{app}}$ | $c_1, c_2$ |
|---|---|---|---|---|---|---|---|
| MHT-DAM | 5 | 100 | 15 | 0.9 | 12 | $0.1, 0.9$ | $0.3, -0.8$ |
| MHT | 5 | 100 | 15 | 0.9 | 6 | $1.0, 0.0$ | |

## 3.4.1 Pruning Effectiveness

As we explained earlier, pruning is central to the success of MHT. It is preferable to have a discriminative score function so that more branches can be pruned early and reliably. A measure to quantify this notion is the entropy:

$$H(B_k) = -\sum_{v} p(B_k = v) \ln p(B_k = v) \tag{3.19}$$

Table 3.2. Results from $2D$ MOT 2015 Challenge (accessed on $9/25/2015$)

| Method | MOTA | MOTP | FAF | MT | ML | FP | FN | IDS | FM | Hz |
|---|---|---|---|---|---|---|---|---|---|---|
| MHT-DAM | **32.4** | **71.8** | 1.6 | **16.0**% | **43.8**% | 9,064 | **32,060** | **435** | 826 | 0.7 |
| MHT | **29.2** | **71.7** | 1.7 | **12.1**% | 53.3% | 9,598 | **33,467** | **476** | **781** | 0.8 |
| LP_SSVM [69] | 25.2 | **71.7** | **1.4** | 5.8% | 53.0% | 8,369 | 36,932 | 646 | 849 | **41.3** |
| ELP [70] | 25.0 | 71.2 | **1.3** | 7.5% | **43.8**% | **7,345** | 37,344 | 1,396 | 1,804 | 5.7 |
| MotiCon [71] | 23.1 | 70.9 | 1.8 | 4.7% | 52.0% | 10,404 | 35,844 | 1,018 | 1,061 | 1.4 |
| SegTrack [72] | 22.5 | **71.7** | **1.4** | 5.8% | 63.9% | **7,890** | 39,020 | 697 | **737** | 0.2 |
| CEM [73] | 19.3 | 70.7 | 2.5 | 8.5% | 46.5% | 14,180 | 34,591 | 813 | 1,023 | 1.1 |
| RMOT [74] | 18.6 | 69.6 | 2.2 | 5.3% | 53.3% | 12,473 | 36,835 | 684 | 1,282 | 7.9 |
| SMOT [75] | 18.2 | 71.2 | 1.5 | 2.8% | 54.8% | 8,780 | 40,310 | 1,148 | 2,132 | 2.7 |
| TBD [76] | 15.9 | 70.9 | 2.6 | 6.4% | 47.9% | 14,943 | 34,777 | 1,939 | 1,963 | 0.7 |
| TC_ODAL [37] | 15.1 | 70.5 | 2.2 | 3.2% | 55.8% | 12,970 | 38,538 | 637 | 1,716 | 1.7 |
| DP_NMS [8] | 14.5 | 70.8 | 2.3 | 6.0% | **40.8**% | 13,171 | 34,814 | 4,537 | 3,090 | **444.8** |

where $p(B_k = v)$ is the probability of selecting $v^{\text{th}}$ tree branch at time $k$ for a given track tree and defined as:

$$p(B_k = v) = \frac{e^{\Delta S^v(k)}}{\sum_v e^{\Delta S^v(k)}}. \tag{3.20}$$

For the normalization, we take all the branches at time $k$ from the same target tree.

With the entropy, we can define the effective number of the branches $N_{\text{eff}}$ within each track tree as:

$$N_{\text{eff}} = e^{H(B_k)}. \tag{3.21}$$

When all the branches in the target tree have the same probability (i.e. when the features are not discriminative), $N_{\text{eff}}$ is equal to the actual number of branches, which means one would need to explore all the possibilities. In the opposite case where a certain branch has the probability of $1$, $N_{\text{eff}}$ is $1$ and it is only necessary to examine a single branch.

Fig. 3.3a shows the number of effective branches for different pruning mechanisms. For this experiment, we set the default gating threshold $d_{\text{th}}$ to $12$. The highest bar (dark red) in each PETS sequence in Fig. 3.3a shows the average number of tree branches generated per frame with the default gating parameter. A smaller gating area ($d_{\text{th}} = 6$) (yellow bar) only reduces the number of branches by a small amount but might prune out fast-moving hypotheses. Combined with the Kalman filter motion model, the reduction is more significant (cyan bar), but the algorithm still retains more than half of the effective branches compared to the full set with $d_{\text{th}} = 12$.

Incorporating the appearance likelihood significantly reduces the effective number of branches. In both the MOT Challenge and PETS sequences, the average effective number of branches in a tree becomes $\sim 50\%$ of the total number of branches. And this is achieved without lowering the size of the gating area, thereby retaining fast-moving targets. This shows that long-term appearance modeling significantly reduces the ambiguities in data association, which makes MHT search more effective and efficient.

**Analysis of Pruning Parameters.** MHT was known to be sensitive to its parameter settings [17]. In this section, we perform a sensitivity analysis of MHT with respect to its pruning parameters and demonstrate that our appearance model helps to alleviate this parameter dependency.

In our MHT implementation, there are two MHT pruning parameters. One is the $N$-scan pruning parameter $N$, the other is the maximum number of tree branches $B_{\text{th}}$. We tested MHT using 7 different values for $N$ and 13 different values for $B_{\text{th}}$. We assessed the number of errors in terms of the MOTA score and identity switches (IDS).

Fig. 3.3b shows the results from this analysis over different $N$-scan parameters. We fix the maximum number of tree branches to 300, a large enough number so that very few branches are pruned when $N$ is large. The results show that motion-based MHT is negatively affected when the $N$-scan parameter is small, while MHT-DAM is much less sensitive to the parameter change. This demonstrates that appearance features are more effective than motion features in reducing the number of look-ahead frames that are required to resolve data association ambiguities. This is intuitive, since many targets are capable of fast movement over a short time scale, while appearance typically changes more slowly.

Fig. 3.3c illustrates the change in the MOTA and IDS scores when the maximum number of branches varies from 1 to 120. We fix the $N$-scan pruning parameter to 5 which is the setting for all other experiments in this chapter. Note that appearance modeling is particularly helpful in preventing identity switches.

### 3.4.2 Benchmark Comparison

We test our method on the MOT Challenge benchmark and the PETS 2009 sequences. The MOT benchmark contains 11 training and 11 testing sequences. Users tune their algorithms on the training sequences and then submit the results on the testing sequences to the evaluation server. This benchmark is of larger scale and includes more variations than the PETS benchmark. Table 3.2 shows our results on the benchmark where MHT-DAM outperforms the best previously published method by more than $7\%$ on MOTA. In addition, $16.0\%$ of the tracks are mostly tracked, as compared to the next competitor at $8.5\%$. We also achieved the lowest number of ID switches by a large margin. This shows the robustness of MHT-DAM over a large variety of videos under different conditions. Also note that because MOT is significantly more difficult than the PETS dataset, the appearance model becomes more important to the performance.

Table 3.3 demonstrates the performance of MHT and MHT-DAM on the PETS sequences compared to one of the state-of-the-art tracking algorithms [64]. For a fair comparison, the detection inputs, ground truth annotations, and evaluation script provided by [64] were used. Our basic MHT implementation already achieves a better or comparable result in comparison to [64] for most PETS sequences and metrics. Cox's method is also surprisingly close in performance to [64] with $\sim 6\%$ lower MOTA on average with the exception of the S2L2 sequence where it is $\sim 20\%$ lower. However, considering that Cox's MHT implementation was done almost 20 years ago, and that it can run in real time due to the efficient implementation (40 FPS on average for PETS), the results from Cox's method are impressive. After adding appearance modeling to MHT, our algorithm MHT-DAM makes fewer ID switches and has higher MOTA and MOTP scores in comparison to previous methods.

## 3.5 Conclusion

Multiple Hypothesis Tracking solves the multidimensional assignment problem through an efficient breadth-first search process centered around the construction and pruning of hypothesis trees. Although it has been a workhorse method for multi-target tracking in general, it has largely fallen out-of-favor for visual tracking. Recent advances in object detection have provided an opportunity to rehabilitate the MHT method. Our results demonstrate that a modern formulation of a standard MHT approach can achieve comparable performance to several state-of-the-art methods on reference datasets. Moreover, an implementation of MHT by Cox [13] from the 1990's comes surprisingly close to state-of-the-art performance on 4 out of 5 PETS sequences. We have further demonstrated that the MHT framework can be extended to include on-line learned appearance models, resulting in substantial performance gains. The software and evaluation results are available from our project website.[2]

[2] `http://cpl.cc.gatech.edu/projects/MHT/`

Table 3.3. Tracking Results on the PETS benchmark

| Sequence | Method | MOTA | MOTP | MT | ML | FM | IDS |
|---|---|---|---|---|---|---|---|
| S2L1 | MHT-DAM | **92.6**% | **79.1**% | **18** | 0 | **12** | **13** |
| | MHT | **92.3**% | **78.8**% | **18** | 0 | **15** | **17** |
| | Cox's MHT [13] | 84.1% | 77.5% | 17 | 0 | 65 | 45 |
| | Milan [64] | 90.3% | 74.3% | **18** | 0 | **15** | 22 |
| S2L2 | MHT-DAM | **59.2**% | **61.4**% | **10** | **2** | 162 | **120** |
| | MHT | 57.2% | 58.7% | 7 | **1** | **150** | **134** |
| | Cox's MHT [13] | 38.0% | 58.8% | 3 | 8 | 273 | 154 |
| | Milan [64] | **58.1**% | **59.8**% | **11** | **1** | **153** | 167 |
| S2L3 | MHT-DAM | 38.5% | **70.8**% | **9** | 22 | **9** | **8** |
| | MHT | **40.8**% | **67.3**% | **10** | **21** | **19** | **18** |
| | Cox's MHT [13] | 34.8% | 66.1% | 6 | 22 | 65 | 35 |
| | Milan [64] | **39.8**% | 65.0% | 8 | **19** | 22 | 27 |
| S1L1-2 | MHT-DAM | **62.1**% | **70.3**% | **21** | **9** | **14** | **11** |
| | MHT | **61.6**% | **68.0**% | **22** | 12 | 23 | 31 |
| | Cox's MHT [13] | 52.0% | 66.5% | 17 | 14 | 52 | 41 |
| | Milan [64] | 60.0% | 61.9% | **21** | **11** | **19** | **22** |
| S1L2-1 | MHT-DAM | **25.4**% | **62.2**% | **3** | 24 | **30** | **25** |
| | MHT | 24.0% | 58.4% | **5** | **23** | **29** | **33** |
| | Cox's MHT [13] | 22.6% | 57.4% | 2 | **23** | 57 | 34 |
| | Milan [64] | **29.6**% | **58.8**% | 2 | **21** | 34 | 42 |

# CHAPTER 4
# BILINEAR LSTM

In recent deep online and near-online multi-object tracking approaches, a difficulty has been to incorporate long-term appearance models to efficiently score object tracks under severe occlusion and multiple missing detections. In this chapter, I propose a novel recurrent network model, the Bilinear LSTM, in order to improve the learning of long-term appearance models via a recurrent network. Based on intuitions drawn from recursive least squares, Bilinear LSTM stores building blocks of a linear predictor in its memory, which is then coupled with the input in a multiplicative manner, instead of the additive coupling in conventional LSTM approaches. Such coupling resembles an online learned classifier/regressor at each time step, which I have found to improve performance in using LSTM for appearance modeling. I also propose novel data augmentation approaches to efficiently train recurrent models that score object tracks on both appearance and motion. I train an LSTM that can score object tracks based on both appearance and motion and utilize it in the MHT framework. In experiments, I show that with the Bilinear LSTM model, the MHT framework achieves competitive performance on near-online multiple object tracking on the MOT 2016 and MOT 2017 benchmarks. The work presented in this chapter has been published as [77].

## 4.1 Introduction

With the improvement in deep learning based detectors [78, 79] and the stimulation of the MOT challenges [80], tracking-by-detection approaches for multi-object tracking have improved significantly in the past few years. Multi-object tracking approaches can be classified into three types depending on the number of lookahead frames: online methods that generate tracking results immediately after processing an input frame [46, 25, 81], near-online methods that look ahead a fixed number of frames before consolidating the decisions [82], and batch methods that consider the entire sequence before generating the decisions [83, 45]. For tracking multiple people, a recent state-of-the-art batch approach [45] relies upon person re-identification techniques which leverage a deep CNN network that can recognize a person that has left the scene and re-entered. Such an approach is able to thread together long tracks in which a person is not visible for dozens of frames, whereas the margin for missing frames in online and near-online approaches is usually much shorter.

A key challenge in online and near-online tracking is the development of deep appearance models that can automatically adapt to the diverse appearance changes of targets over multiple video frames. A few approaches based on Recurrent Neural Networks (RNNs) [46, 25] have been proposed in the context of multi-object tracking. [46] focuses on building a non-linear motion model and a data association solver using RNNs. [25] successfully adopted Long Short-Term Memory (LSTM) [84] to integrate appearance, motion, and interaction cues, but Figure 7. (b) in [25] reports results for sequences (tracks) of maximum length 10. In practice, object tracks are much longer than 10 frames, and it is unclear whether the method is equally effective for longer tracks.

Our own experience, coupled with the reported literature, suggests that it is difficult to use LSTMs to model object appearance over long sequences. It is therefore worthwhile to investigate the fundamental issues in utilizing LSTM for tracking, such as what is being stored in their internal memory and what factors result in them being either able or unable to learn good appearance models. Leveraging intuition from classical recursive least squares regression, we propose a new type of LSTM that is suitable for learning sequential appearance models. Whereas in a conventional LSTM, the memory and the input have a linear relationship, in our *Bilinear LSTM*, the LSTM memory serves as the building blocks of a *predictor* (classifier/regressor), which leads to the output being based on a multiplicative relationship between the memory and the input appearance. Based on this novel LSTM formulation, we are able to build a recurrent network for scoring object tracks that combines long-term appearance and motion information. This new track scorer is then utilized in conjunction with an established near-online multi-object tracking approach, multiple hypothesis tracking, which reasons over multiple track proposals (hypotheses). Our approach combines the benefits of deep feature learning with the practical utility of a near-online tracker.

Our second contribution is a training methodology for generating sequential training examples from multi-object tracking datasets that accounts for the cases where detections could be noisy or missing for many frames. We have developed systematic data augmentation methods that allow our near-online approach to take advantage of long training sequences and survive scenarios with detection noise and dozens of frames of consecutive missing detections.

With these two improvements, we are able to generate state-of-the-art multi-target tracking results for near-online approaches in the MOT challenge. In the future, our proposed Bilinear LSTM could be used in other scenarios where a long-term online predictor is needed.

## 4.2 Overview of MHT

In tracking-by-detection, multi-object tracking is solved through data association, which generates a set of tracks by assigning a track label to each detection. MHT solves the data association problem by explicitly generating multiple track proposals and then selecting the most promising ones. Let $T_l(t) = \{d_1^l, d_2^l, \ldots, d_{t-1}^l, d_t^l\}$ denote the $l^{\text{th}}$ track proposal at frame $t$ and let $d_t^l$ be a detection selected by the $l^{\text{th}}$ track proposal at frame $t$. The selected detection $d_t^l$ can be either an actual detection generated by an object detector or a dummy detection that represents a missing detection.

The track proposals for each object are stored in a track tree in which each tree node corresponds to one detection. For example, the root node represents the first detection of the object and the child nodes represent the detections in subsequent frames (i.e. tree nodes at the same depth represent detections in the same frame). Thus, multiple paths from the root to the leaf nodes correspond to multiple track proposals for a single object. The proposals are scored, and the task of finding the best set of proposals can be formulated as a Maximum Weighted Independent Set (MWIS) problem [15], with the score for each proposal being the weight of it. Once the best set of proposals is found, proposal pruning is performed. Only the surviving proposals are kept and updated in the next frame. More details about MHT can be found in chapter 3.

### 4.2.1 Gating in MHT

In MHT, track proposals are updated by extending existing track proposals with new detections. In order to keep the number of proposals manageable, exisiting track proposals are not updated with all of the new detections, but rather with a few selected detections. The selection process is called *gating*. Previous gating approaches rely on hand-designed track score functions [15, 85, 13]. Typically, the proposal score $S(T_l(t))$ is defined recursively as:

$$S(T_l(t)) = S(T_l(t-1)) + \Delta S(T_l(t)) \tag{4.1}$$

Gating is done by thresholding the score increment $\Delta S(T_l(t))$. New track proposals with score increments below a certain threshold are pruned instantly. Usually the proposal score includes an appearance term, which could be learned by recursive least squares, as well as a motion term which could be learned with Kalman filtering.

### 4.2.2 Recursive Least Squares as an Appearance Model

An important advantage of our MHT-DAM approach introduced in chapter 3 is the use of long-term appearance models that leverage all prior appearance samples from a given track and train a discriminative model to predict whether each bounding box belongs to each given track. Because we would like to be able to perform a similar task in our LSTM framework, we briefly review the recursive least squares appearance model. Given all the $n_t$ detections at frame $t$, one can extract appearance features (e.g. CNN fully-connected layer) for them and store them in an $n_t \times d$ matrix $\mathbf{X}_t$, where $d$ is the feature dimensionality. Then, suppose that we are tracking $k$ object tracks, an output vector can be created for each track as, e.g. the spatial overlap between the bounding box of each detection and each track (represented by one detection in the frame), with the set of output vectors denoted as an $n_t \times k$ matrix $\mathbf{Y}_t$. Then a regressor for each target can be found by least squares regression:

$$\min_{\mathbf{W}} \sum_{t=1}^{T} \|\mathbf{X}_t \mathbf{W} - \mathbf{Y}_t\|_F^2 + \lambda \|\mathbf{W}\|_F^2 \tag{4.2}$$

where $\|\cdot\|_F^2$ is a squared Frobenius norm and $\lambda$ is the regularization parameter. As is well-known, the solution can be written as:

$$\mathbf{W} = \left( \sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{X}_t + \lambda \mathbf{I} \right)^{-1} \left( \sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{Y}_t \right) \tag{4.3}$$

where $\mathbf{I}$ is the identity matrix. Notably, one can store $\mathbf{Q}_t = \sum_{i=1}^{t} \left( \mathbf{X}_i^\top \mathbf{X}_i \right)$ and $\mathbf{C}_t = \sum_{i=1}^{t} \left( \mathbf{X}_i^\top \mathbf{Y}_i \right)$ and update them online at frame $t+1$, by adding $\mathbf{X}_{t+1}^\top \mathbf{X}_{t+1}$ and $\mathbf{X}_{t+1}^\top \mathbf{Y}_{t+1}$ to $\mathbf{Q}_t$ and $\mathbf{C}_t$ respectively, while maintaining the optimality of the solution for $\mathbf{W}$. Moreover, the computation of $\mathbf{W}$ is only linear in the number of tracks $k$. The resulting model can train on all the positive examples (past detections in each track) and negative examples (past detections in other tracks not overlapping with a given track) and generate a regressor with good discriminative power. The computational efficiency of this approach and its optimality are the two keys to the success of the MHT-DAM framework.

## 4.3 RNN as a Gating Network

We use the term *gating network* to denote a neural network that performs gating. We utilize recurrent neural networks (RNNs) for training gating networks since track proposals constitute sequential data whose data size is not fixed. In this work, we adopt Long Short-

Term Memory (LSTM) as a recurrent layer due to its success in modeling long sequences on various tasks [86].

We formulate the problem of gating as a sequence labeling problem. The gating network takes track proposals as inputs and performs gating by generating a binary output for every detection in the track proposal. In this section, we describe network inputs and outputs and its utilization within the MHT framework. More details about the network architecture can be found in Sec. 5.5.

**Input.** Track proposals contain both motion and appearance information. We use the bounding box coordinates $(x, y, w, h)$ over time as motion inputs to the network. The coordinates are normalized with respect to the frame resolution $(\frac{x}{\text{width}}, \frac{y}{\text{height}}, \frac{w}{\text{width}}, \frac{h}{\text{height}})$ to make the range of the input values fixed regardless of the frame resolution. We also calculate sample mean and standard deviation from track proposals (see Sec. 4.4 for more details on how to generate track proposals from multi-object tracking datasets) and perform another normalization in order to make the input data zero-centered and normalized across different dimensions.

We use object images cropped to detection bounding boxes as appearance inputs to the network. RGB cropped images are first converted to convolutional features by Convolutional Neural Networks (CNN) before the gating networks process them. We use the ImageNet pretrained ResNet-50 [87] as our CNN.

**Output.** Given a current detection, the network makes a binary decision about whether or not it belongs to the proposal based on its compatibility with the appearance and motion of the other detections assigned to the proposal. Thus, the gating networks solve a binary classification task using cross-entropy loss. Note that we have multiple binary labels for each track sequence since gating is done on every frame.

**Track Scorer in MHT.** We use the softmax probability $p$ of the positive output (i.e. current detection belongs to the same object in the proposal) for calculating the score increment $\Delta S(T_l(t))$ as shown in Eq.(4.4). A higher score increment implies a higher matching quality between the track proposal $T_l(t-1)$ and the detection $d_t^l$.

$$\Delta S(T_l(t)) = p(d_t^l \in T_l(t-1) | T_l(t-1)) \tag{4.4}$$

This is a simple aggregation scheme that combines the per-frame predictions from the gating network in order to score tracks. Our assumption is that proposals that generate higher per-frame matching scores are more likely to be correct than proposals with lower per-frame matching scores. New track proposals with score increment below a threshold are pruned instantly by gating. In MHT, every track proposal in the track trees has a unique detection sequence, which is represented as a unique LSTM memory state in the gating network. The memory state for surviving proposals is stored for further gating and scoring in the next frame. The weights of the gating network are shared across all track proposals.

## 4.3.1 Bilinear LSTM

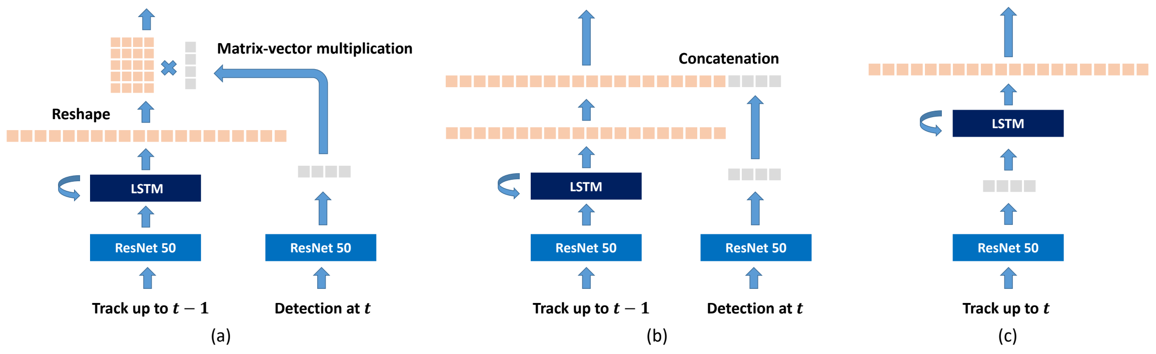


Figure 4.1. Motion gating network and appearance gating network are trained separately before training the full model. We evaluate multiple network architectures for each module. (a) The Bilinear LSTM network with a multiplicative relationship between the memory and the input CNN features. The LSTM memory is reshaped into a matrix and multiplied with the input appearance feature vector; (b) Input appearance is concatenated with the LSTM memory output before a fully-connected layer; (c) A conventional LSTM architecture.

Our experience suggests that conventional LSTMs are far more effective at modeling motion than appearance. This led us to ask, "what information about object appearance is being stored in the internal memory of a standard LSTM, and what would be an ideal memory representation for this task?".

Conventional LSTMs utilizes the following update rule:

$$\begin{aligned}
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t, & \mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t]), & \mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t]), \\
\mathbf{g}_t &= \sigma(\mathbf{W}_g[\mathbf{h}_{t-1}, \mathbf{x}_t]), & \mathbf{o}_t &= \tanh(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t])
\end{aligned} \tag{4.5}$$

where $\circ$ represents the Hadamard product. $\mathbf{x}_t$ is the current input. $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ are the forget gate, the input gate, and the output gate. $\mathbf{c}_t$ and $\mathbf{h}_t$ are the cell state and the hidden state that are repeatedly updated throughout the sequence. $\mathbf{g}_t$ is the new update values for the cell state.

When building an appearance model for multi-object tracking (i.e. data association), $\mathbf{x}_t$ represents the current appearance of an object candidate. For LSTM to solve the tracking task, one intuition is that $\mathbf{h}_t$ may represent some information about the acceptance/rejection of an object candidate. $\mathbf{c}_t$ can roughly be thought of as representing a stored template of the object appearance, and then the output gate $\mathbf{o}_t$ compares the previous stored appearance $\mathbf{c}_{t-1}$ and the new appearance $\mathbf{x}_t$ in order to decide the current output $\mathbf{h}_t$. Experiments in [25] where LSTM performance seems to saturate with a sequence length of $2-4$ frames, seems to suggest that the aforementioned intuition might be partially correct.

However, the main appeal of a long-term appearance model in previous work is the capability of using a classifier/regressor that learns from *all* the previous appearances of the object as shown in the previous chapter. Such a model trained from multiple different appearances could generalize better than one or a few stored templates and could potentially interpolate and extrapolate among different previous appearances of the model. An example would be the recursive least squares model in Eq. (4.2). But if we imagine the $\mathbf{W}$ in Eq. (4.2) as the memory output $\mathbf{h}_t$, it seems that a multiplicative form $\mathbf{x}^\top\mathbf{W}$ as in Eq. (4.2) is difficult to obtain from the additive forms in Eq. (4.5), no matter from $\mathbf{o}_t, \mathbf{c}_t$ or $\mathbf{h}_t$.

Thus, we would like to propose a new LSTM that can realize the multiplicative between the memory $\mathbf{h}_t$ and the input $\mathbf{x}$. We note that the solution of recursive least squares is dependent on the matrix $\mathbf{Q}_t = \sum_{i=1}^{t} \mathbf{X}_i^\top \mathbf{X}_i$ that is updated at each time linearly. It is difficult for LSTM to store a positive-definite matrix as memory, but a common approach to simplify such a positive-definite matrix is to use a low-rank approximation, e.g. assuming $\mathbf{Q}_t^{-1} = \sum_{i=1}^{r} \mathbf{q}_{ti}\mathbf{q}_{ti}^\top$. With this assumption and considering Eq. (4.3), the regressor output becomes:

$$\mathbf{w}^\top \mathbf{x} \;=\; \mathbf{C}_t^\top \mathbf{Q}_t^{-1}\mathbf{x} = \sum_{i=1}^{r} \mathbf{C}_t^\top \mathbf{q}_{ti}\mathbf{q}_{ti}^\top \mathbf{x} \tag{4.6}$$

Note that when there is only $1$ track, $\mathbf{C}_t$ is of the dimensionality $d \times 1$, and hence $\mu_i = \mathbf{C}_t^\top \mathbf{q}_i$ is a scalar. We have:

$$\mathbf{w}^\top \mathbf{x} = \sum_{i=1}^{r} \mu_i \mathbf{q}_{ti}^\top \mathbf{x} \tag{4.7}$$

Here $\mu_i$ is dependent on both $\mathbf{y}$ and $\mathbf{q}$, hence without loss of generality it could be a standalone variable that is separately estimated. With this derivation, it seems that the approach to emulate a linear regressor is to have several vectors $\mathbf{h}_{ti}$ to be learnable and gradually changing with time (in other words, serve as the memory in the LSTM), and a layer of learnable $\mu_i$ on top of a multiplicative relationship between $\mathbf{h}_{ti}$ and $\mathbf{x}$.

In this spirit, we propose the Bilinear LSTM (bLSTM) which utilizes the following forward pass that enables the multiplicative interaction between the input and the memory:

$$\begin{aligned} \mathbf{h}_{t-1} &= [\mathbf{h}_{t-1,1}^\top|\mathbf{h}_{t-1,2}^\top|...|\mathbf{h}_{t-1,r}^\top]^\top = \mathbf{o}_{t-1} \circ \tanh(\mathbf{c}_{t-1}) \\ \mathbf{H}_{t-1}^{\text{reshaped}} &= [\mathbf{h}_{t-1,1}|\mathbf{h}_{t-1,2}|...|\mathbf{h}_{t-1,r}]^\top, \quad \mathbf{m}_t = f(\mathbf{H}_{t-1}^{\text{reshaped}}\mathbf{x}_t) \end{aligned} \tag{4.8}$$

where $f(\cdot)$ is a non-linear activation function and $\mathbf{m}_t$ is the new hidden state for bLSTM.

$\mathbf{x}_t$ denotes the features from a box at frame $t$. Basically, we utilize a long vector as the LSTM memory which contains the concatenation of all the $\mathbf{h}_{t-1,i}$s. When it comes to the time to multiply $\mathbf{h}_{t-1,i}$ with $\mathbf{x}_t$, the $rd$ dimensional vector $\mathbf{h}_{t-1}$ is reshaped into the $r \times d$ matrix $\mathbf{H}_{t-1}^{\text{reshaped}}$ so that we could utilize matrix-vector multiplication between $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$.

The new hidden state $\mathbf{m}_t$ can then be used as input to other fully-connected layers (resembling the $\mu_i$ in eq. (4.7)) to generate the final prediction. Note that in online recursive least squares $\mu_i$ should be trained for each tracked object separately, however in our network the fully-connected layers after bLSTM are trained globally and fixed during testing time. Implementing a dynamic $\mu_i$ which is dependent on each object track did not result in significant improvement in performance. We believe that since the system is trained end-to-end, the LSTM updates of $\mathbf{h}$ should be able to encompass the potential changes in $\mu_i$, hence we can keep the fully-connected layers fixed without additional issues.

Intuitively, by saving a matrix-valued memory that resembles a low-rank decomposition of the matrix, at least r templates (as well as combinations of the r templates) can be used for the prediction. Hence bLSTM can store longer-term appearance models than traditional LSTMs and improve on maintaining track identity over many frames.

We have three types of gating networks based on the network input: Motion, Appearance, and Motion + Appearance. We test three different architectures in Fig. 4.1 for the motion gating and appearance gating networks. We select the best architecture for each type of input among the three and combine them for motion+appearance gating networks. Experimental results that we used for selecting the architecture are included in Sec. 4.5.3.

**Motion Gating.** For motion gating, the vanila version of LSTM eq. (4.5) works the best. Thus, we adopt LSTM as a sequence labeler where LSTM reads motion input recursively and store the sequence information in its hidden state. The FC layers are built on top of the hidden state to produce the final output. Architectures that we used for the comparison are shown in Table 4.1.

**Appearance Gating.** We propose to use Bilinear LSTM as appearance gating network where LSTM's hidden state becomes a weight vector for the appearance model of the current object. Details about the network architecture and other two baseline architectures are shown in Table 4.2.

Table 4.1. Different experimented architectures for motion gating. (a) Bilinear LSTM (b) LSTM as a feature extractor for the previous track (c) Vanila LSTM (LSTM as a sequence labeler)

| | | | |
|---|---|---|---|
| Soft-max | | | |
| Matrix-vector Multiplication-tanh 4 | | | |
| Reshape | $4 \times 64$ | Reshape | $64 \times 1$ |
| LSTM | 256 | | |
| FC-relu | 64 | FC-relu | 64 |
| Input at $t-1$ | 4 | Input at $t$ | 4 |

(a)

| | | | |
|---|---|---|---|
| Soft-max | | | |
| FC-tanh $64$ | | | |
| Concatenation $64+64$ | | | |
| LSTM | 64 | | |
| FC-relu | 64 | FC-relu | 64 |
| Input at $t-1$ | 4 | Input at $t$ | 4 |

(b)

| | |
|---|---|
| Soft-max | |
| FC-tanh | 8 |
| LSTM | 64 |
| FC-relu | 64 |
| Input at $t$ | 4 |

(c)

Table 4.2. Different experimented architectures for appearance gating: (a) Bilinear LSTM (b) LSTM as a feature extractor for the previous track (c) Vanila LSTM

| | | | |
|---|---|---|---|
| Soft-max | | | |
| Matrix-vector Multiplication-relu 8 | | | |
| Reshape | $8 \times 256$ | Reshape | $256 \times 1$ |
| LSTM | 2048 | | |
| FC-relu | 256 | FC-relu | 256 |
| ResNet-50 | 2048 | ResNet50 | 2048 |
| Input at $t-1$ | $128 \times 64 \times 3$ | Input at $t$ | $128 \times 64 \times 3$ |

(a)

| | | | |
|---|---|---|---|
| Soft-max | | | |
| FC-relu 512 | | | |
| Concatenation $2048+256$ | | | |
| LSTM | 2048 | | |
| FC-relu | 256 | FC-relu | 256 |
| ResNet-50 | 2048 | ResNet50 | 2048 |
| Input at $t-1$ | $128 \times 64 \times 3$ | Input at $t$ | $128 \times 64 \times 3$ |

(b)

| | |
|---|---|
| Soft-max | |
| FC-relu | 512 |
| LSTM | 2048 |
| FC-relu | 256 |
| ResNet-50 | 2048 |
| Input at $t$ | $128 \times 64 \times 3$ |

(c)

**Motion + Appearance Gating.** In order to enable a joint reasoning over both motion and appearance for object tracking, we construct a motion+gating network based on our analysis of different baseline architectures. We use Bilinear LSTM to process appearance data and vanila LSTM to process motion data. Then motion and appearance representations (i.e. outputs before soft-max) from both gating networks are concatenated after L2 normalization is applied to each representation separately. Prediction layers are built upon the concatenated features. We first train motion gating and appearance gating networks separately. Then we load all the pretrained layers before the concatenation layer from both gating networks and fine-tune them jointly.

#### 4.3.2 Handling Missing Detections

In tracking-by-detection, it is important to handle missing detections while keeping the correct track identity over time. In traditional Kalman filter-based motion tracking, the diagonal of the noise covariance matrix keeps increasing over time in the case of missing detections, resulting in accepting more detections from gating with a gradually larger gating area.

In the case of recurrent gating networks, the occurrences of missing detections should also modulate the gating network. For instance, one can imagine a gating network applying a stricter gating policy when all the detections are available for the current object than the case where detections are missing in recent frames. In order to encode such information inside the LSTM hidden states, we propose to input to the recurrent network all-zero input vectors in the case of missing detections. By doing so, the LSTM internal (both cell and hidden) states will be updated solely based on its previous states, which is different from normal LSTM updates where both the input data and the previous state are utilized. The gating network does not need to make any prediction for the missing detection but only need to update the LSTM internal memory. In Sec. 4.5.3, we show the effectivenss of such explicit missing detection handling for the motion gating networks.

### 4.4 Generating Training Sequences

Artificial track proposals are generated from ground truth track annotations as training data for training our LSTM network. First, we randomly pick one ground truth track annotation from which we sample track proposals. The starting frame and the ending frame are randomly selected. Due to the memory limit of GPUs, we select them in a way that the length of the track proposal does not exceed $N_{\text{max}}$. Let $N$ be the selected length $(2 \leq N \leq N_{\text{max}})$ of the proposal. Then we collect first $N-1$ bounding boxes of the selected object and pick the last $N^{\text{th}}$ detection from a different object. Positive labels are assigned for the first

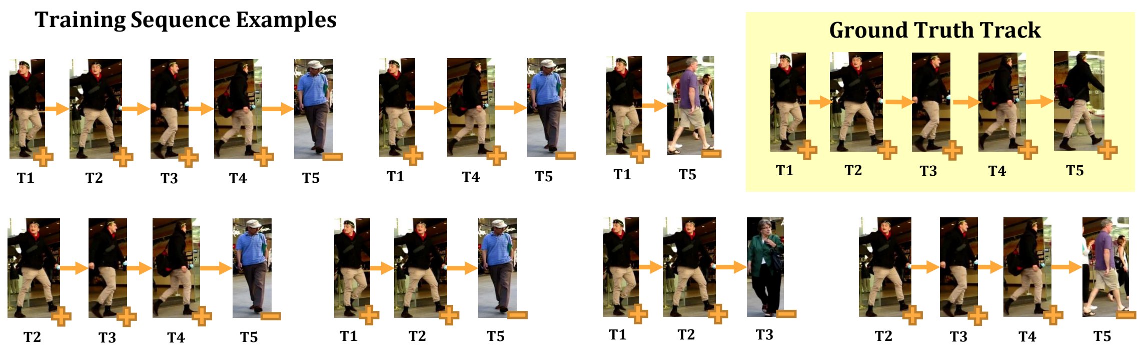


Figure 4.2. Training sequences are generated from the public MOT dataset. Each training sequence has detections from the same object throughout the track and one detection from different object at the end. Training sequences are generated in the manner that they reflect actual track proposals that MHT generates during tracking.

$N-1$ detections representing the correct object and a negative label is assigned for the $N^{\text{th}}$ detection representing a different object. Thus, each proposal is associated with a binary label vector where only the last element is a negative label as presented in Fig. 4.2. The maximum length $N_{\max}$ needs to be large enough so that the network learns the gating mechanism regardless of its input length. We show experimental results with different $N_{\max}$ values in Sec. 4.5.3.

**Data Augmentation.** If ground truth tracks are used without any augmentation to generate track proposals, each track proposal will consist of bounding boxes perfectly aligned with the object in consecutive frames, which may poorly represent actual track proposals consisting of noisy detections. Thus, it is important to perform proper data augmentation so that the track proposals reflect actual detection noise. There are two types of detection errors which need to be considered: localization error and missing detections.

In order to reflect the localization noise, we jitter the bounding boxes in the training track proposals using a noise model estimated from the training data. For estimating this noise model, given a set of detections and ground truth annotations, we first assign each detection to its closest ground truth bounding box, and then calculate localization error from each detection to its assigned ground truth bounding box. We then fit a normal distribution to these localization errors for all true positive detections. Since the MOT Challenge Benchmark [80] provides three public detectors (DPM [88], FRCNN [89], SDP [79]) which have different accuracy and noise levels, we estimate a different normal distribution for each public detector. Thus, before the training data generator samples random localization errors for the proposal, it first chooses a normal distribution based on the detector. Then for each bounding box in each track proposal, a different localization error is sampled from the estimated normal distribution.

In order to simulate missing detections, for $50\%$ of the tracks, we randomly pick a missing detection rate $p_{\text{miss}}(0.0 \sim 0.5)$ and drop the bounding boxes in the track proposal according to the selected missing rate except for the first bounding box (current object) and the last bounding box (differerent object). Example track proposals with this missing detection augmentation are shown in Fig. 4.1. The other $50\%$ of the tracks are retained without missing detections.

## 4.5 Experiments

We report all our experimental results on the validation set except for the final benchmark result which was evaluated on MOT 16/17 test sequences.

### 4.5.1 Training Data

In order to generate track proposals, we use MOT17 (MOT16) and MOT15 sequences [80, 90] and a few other tracking sequences [7, 91, 92] where pedestrian annoatations are available. All the training, validation, and testing sequences are shown in Table 4.3. In addition to the MOT sequences, we also use two public person re-identification datasets, Market1501 [93] and CUHK03 [94], in order to pre-train the appearance gating networks.

Table 4.3. Training/Val/Test Splits

| Training Set | Validation Set | Test Set |
|---|---|---|
| MOT17 - {02, 04, 05, 11, 13}, MOT15 - {PETS09-S2L1, ETH - {Sunnyday, Bahnhof}, TUD - {Campus, Stadtmitte}, KITTI-{17, 13}}, ETH - {Jelmoli, Seq01}, KITTI - {16, 19}, PETS09-S2L2, TUD-Crossing, AVG-TownCentre | MOT17 - {09, 10} | MOT17 - {01, 03, 06, 07, 08, 12, 14} |

### 4.5.2 Pre-training on Person Re-identification

In the person re-identification task, a pair of images is given to the learner and the learner decides whether two images come from the same person or not. One can treat the pair of two images as a track proposal with a temporal length 2. Such training examples can be also generated from multi-object tracking dataset. Thus, we utilize a person re-identification dataset in addition to the training set shown in Table 4.3 to pre-train our appearance gating network for person re-identification. Similar pre-training was also done in [25, 45]. Table 4.5 shows the effect of pre-training for person re-identification on the performance of gating networks.

### 4.5.3 Ablation Study

We conduct an ablation study for different network architectures and training settings on our validation sequences (MOT17-09 and MOT17-10). The MOT17 Benchmark provides three different public detectors. We used the Faster R-CNN detector for the experimental results in this section.

**Metrics.** Among many different tracking metrics, we choose the Multiple Object Tracking Accuracy (MOTA) [67], identity switches (IDS), and IDF1 [95] for this study. MOTA is calculated by object detection mistakes (false positive and false negative) and tracking mistakes (identity switches). MOTA is often dominated by object detection mistakes since the number of false positive/negative is typically much higher than IDS. IDS counts the number of track ID changes for all the objects. IDF1 is a tracking metric which measures how often objects are correctly identified by the same track ID.

Table 4.4. Ablation Study for Appearance Gating Networks. Baseline1 and Baseline2 are the networks shown in Table 4.2 (b) and (c) resepectively. **(Left)** State dim. $= 2048$, $N_{\text{max}} = 40$ **(Middle)** LSTM: Bilinear, $N_{\text{max}} = 40$, **(Right)** LSTM: Bilinear, State dim. $= 2048$

| LSTM | MOTA | IDF1 | IDS |
|---|---|---|---|
| Bilinear | **52.33** | **59.07** | **233** |
| Baseline1 | 50.43 | 51.28 | 412 |
| Baseline2 | 50.97 | 51.49 | 462 |

| State dim. | MOTA | IDF1 | IDS |
|---|---|---|---|
| 512 | 52.14 | 56.66 | 283 |
| 1024 | 52.36 | 55.85 | **222** |
| 2048 | 52.33 | **59.07** | 233 |

| $N_{\text{max}}$ | MOTA | IDF1 | IDS |
|---|---|---|---|
| 10 | 51.96 | 54.36 | 271 |
| 20 | 52.27 | 58.38 | 228 |
| 40 | 52.33 | **59.07** | 233 |
| 80 | 52.32 | 57.21 | 239 |
| 160 | 52.41 | 55.19 | **222** |

Table 4.5. Pre-training vs Random initialization **(Left)** LSTM: Bilinear, State dim. $= 2048$, $N_{\text{max}} = 40$ **(Right)** LSTM: Baseline2 (Motion) + Bilinear (Appearance), State dim. $=$ 64 (Motion), 2048 (Appearance), $N_{\text{max}} = 40$

| Input type | MOTA | IDF1 | IDS |
|---|---|---|---|
| (A) Random | 52.00 | 57.46 | 268 |
| (A) Pre-training | 52.33 | **59.07** | **233** |

| Pre-training | MOTA | IDF1 | IDS |
|---|---|---|---|
| (M)+(A) Random | 50.31 | 50.39 | 499 |
| (M)+(A) Pre-training | **52.63** | **58.08** | **197** |

**Network Architectures.** We test the three deep architectures shown in Fig. 4.1 for MHT gating. The results in Table 4.4 are generated by MHT with different appearance gating networks. The left table in Table 4.4 shows the tracking performance of different deep architectures as gating networks. Bilinear LSTM works best as the appearance gating network. In terms of the network sizes (LSTM state dimensions), 2048 state dimension is a good choice for Bilinear LSTM as an appearance gating network.

**Training Settings.** We also try differerent training settings such as different maximum sequence lengths, missing detection augmentations, and network pre-training. The results are included in Table 4.4, 4.5, and 4.6. We used the (M)+(A) model (in Table 4.6 (Middle)) which balances well between IDF1 and IDS as our final model for the comparison with MHT and the MOT benchmark in Sec. 4.5.4.

We used the Adam optimizer [96] for training motion gating networks and set the initial learning rate to 0.01 with the batch size of 64. We used the stochastic gradient optimizer for training appearance and motion+appearance gating networks and set the initial learning rate to 0.005 with the batch size of 16. In all cases, we let the learning rate decrease every

Table 4.6. **(Left)** Missing Detection Augmentation On/Off. $N_{\text{max}} = 40$. We update LSTM states with zero input vectors (as described in Sec. 4.3.2) for the models which are trained with the missing detection augmentation. The results in this table show that such LSTM state update is beneficial for the motion gating network, but not for the appearance gating network. Thus, we utilize the missing detection handling only for the motion gating network and the motion part in the motion+appearance gating network. **(Middle and Right)** Different input types with different maximum lengths of training sequences (Middle) $N_{\text{max}} = 40$ (Right) $N_{\text{max}} = 80$.

| Missing Det. | MOTA | IDF1 | IDS |
|---|---|---|---|
| (M) Yes | 52.47 | **50.22** | 229 |
| (M) No | 52.58 | 47.71 | **203** |
| (A) Yes | 52.29 | 41.37 | 244 |
| (A) No | 52.33 | **59.07** | **233** |

| Input type | MOTA | IDF1 | IDS |
|---|---|---|---|
| Motion (M) | 52.47 | 50.22 | 229 |
| Appearance (A) | 52.33 | **59.07** | 233 |
| (M) + (A) | 52.63 | 58.08 | **197** |

| Input type | MOTA | IDF1 | IDS |
|---|---|---|---|
| Motion (M) | 52.30 | 51.14 | 255 |
| Appearance (A) | 52.32 | **57.21** | 239 |
| (M) + (A) | 52.69 | 54.63 | **208** |

Table 4.7. Comparison with MHT-DAM on our val split (MOT17-09 and MOT17-10). $N_{\text{max}} = 80$. Tracks are interpolated through smoothing. **(Left)** DPM **(Middle)** Faster R-CNN **(Right)** SDP.

| Method | MOTA | IDF1 | IDS |
|---|---|---|---|
| MHT-DAM | **47.6** | 48.2 | **72** |
| Ours | 43.8 | **52.9** | 91 |

| Method | MOTA | IDF1 | IDS |
|---|---|---|---|
| MHT-DAM | 53.7 | 54.8 | **136** |
| Ours | **54.8** | **60.5** | 140 |

| Method | MOTA | IDF1 | IDS |
|---|---|---|---|
| MHT-DAM | **69.4** | 62.7 | **128** |
| Ours | **69.7** | **68.6** | 137 |

5000 iterations by exponential decay with the decay rate 0.9 until we observe a decrease in performance on the validation set.

### 4.5.4 MOT Challenge Benchmark

In this section, we report the performance comparison with MHT-DAM and our tracking results on the MOT Challenge 17/16 Benchmark.

**Comparison with MHT-DAM.** In order to see whether our trained models work well with MHT, we first compare the tracking performance with MHT-DAM presented in chapter 3 on the validation split. Unlike bLSTM, MHT-DAM does not benefit from any off-line training using multi-object tracking datasets. It rather builds appearance models for multiple objects in an online manner. Table 4.7 shows the comparison in results. Our new MHT with bLSTM works well when Faster RCNN and SDP provide input detections. However, for the case of DPM, MOTA score is lower compared to MHT-DAM, although our new method still shows stronger performance on IDF1. We believe that this is because DPM produces quite noisy detections while we use ground truth tracks to generate training sequences for our model. Thus, there could be still some gap between our training data (even after the data augmentation) and track proposals constructed from DPM.

**MOT16/17 Challenge Benchmarks.** We used the same model and setting as shown in Table 4.7 as our final method for evaluating on the MOT test sequences. The results are included in Table 4.8. we grouped previous methods that are closely related to our method separately in order to see the performance difference among these methods. The left table is the MOT 17 Challenge result where DPM, Faster RCNN, and SDP were used as public detectors. The right table is the MOT 16 Challenge result where only DPM was used as a public detector. As we described in the comparison with MHT-DAM above, our model does not seem to perform well when DPM provides input detections to the tracker. Thus, our tracker achieved more favorable performance on MOT 17 than on MOT 16 compared to other state-of-the-art trackers.

Table 4.8. Results from MOT $2017/2016$ Challenge (accessed on $7/26/2018$)

| Method | MOTA | IDF1 | IDS | Hz |
|---|---|---|---|---|
| JCC [97] | **51.2** | **54.5** | **1,802** | 1.8 |
| MOTDT17* [98] | 50.9 | 52.7 | 2,474 | **18.3** |
| PHD-GSDL17 [99] | 48.0 | 49.6 | 3,998 | 6.7 |
| FWT* [100] | **51.3** | 47.6 | 2,648 | 0.2 |
| MHT Methods | | | | |
| EDMT17* [101] | 50.0 | 51.3 | 2,264 | 0.6 |
| MHT-DAM | **50.7** | 47.2 | 2,314 | 0.9 |
| MHT-bLSTM* | 47.5 | **51.9** | **2,069** | **1.9** |

* indicates the use of additional training data

| Method | MOTA | IDF1 | IDS | Hz |
|---|---|---|---|---|
| NOMT [82] | 46.4 | **53.3** | **359** | 2.6 |
| MCjoint [97] | 47.1 | 52.3 | 370 | 0.6 |
| LMP* [45] | **48.8** | 51.3 | 481 | 0.5 |
| STAM16 [102] | 46.0 | 50.0 | 473 | 0.2 |
| RAR16pub [26] | 45.9 | 48.8 | 648 | 0.9 |
| NLLMPa [103] | 47.6 | 47.3 | 629 | **8.3** |
| JMC [83] | 46.3 | 46.3 | 657 | 0.8 |
| LINF1 [104] | 41.0 | 45.7 | 430 | 4.2 |
| CDA-DDALv2* [105] | 43.9 | 45.1 | 676 | 0.5 |
| MHT and LSTM-based Methods | | | | |
| EDMT* [101] | 45.3 | **47.9** | 639 | **1.8** |
| AMIR* [25] | **47.2** | 46.3 | 774 | 1.0 |
| MHT-DAM | 45.8 | 46.1 | **590** | 0.8 |
| MHT-bLSTM* | 42.1 | **47.8** | 735 | **1.8** |

## 4.6 Conclusion

In this chapter, we proposed using an LSTM network to score track proposals in a near-online multiple hypothesis tracking framework. In order to properly take into account multiple past appearances, we proposed a Bilinear LSTM algorithm that slices the LSTM memory into several vectors and uses a matrix vector multiplication between the memory output and the appearance input to simulate a discriminatively trained predictor model. Such an algorithm is shown to be better than traditional LSTM in modeling the appearance of each track, especially in terms of maintaining track identities. Jointly using appearance and motion LSTM gating networks in an MHT framework, we have achieved state-of-the-art performances in the MOT challenges for near-online methods. We believe the proposed Bilinear LSTM is general and could be applicable in many other problems that require learning an online sequential discriminative model using an end-to-end approach and will explore those as future work.

# CHAPTER 5
# BILINEAR LSTM WITH MULTI-TRACK POOLING

The Bilinear LSTM model presented in Chapter 4 models each target in isolation and lack the ability to use all the targets in the scene to jointly update the memory. This can be problematic when there are similar looking objects in the scene. In this chapter, I solve the problem of simultaneously considering all tracks during memory updating, with only a small spatial overhead, via a novel multi-track pooling module. I additionally propose a training strategy adapted to multi-track pooling which generates hard tracking episodes online. I show that the combination of these innovations results in a strong discriminative appearance model, enabling the use of greedy data association to achieve online tracking performance. My experiments demonstrate real-time, state-of-the-art performance on public multi-object tracking (MOT) datasets.

## 5.1 Introduction

In the typical tracking-by-detection setting of multi-object tracking, a trained target detector is assumed to exist, and the goal of the tracking algorithm is to solve the *data association* problem: associating detections from different frames into tracks. The standard approach involves building an appearance model and a motion model for each target being tracked. The appearance model stores features from past appearances, which are compared against detections in each new frame, and the motion model predicts where the target will be located in the next frame. Because of the ambiguity in appearance, in the past, many high-performance trackers utilized sophisticated data association schemes that relied on evidence from multiple frames to determine the correct track. This has included frames from the future, sometimes with a significant look-ahead. These choices result in a tracker which is not capable of *real-time online* performance, i.e. it cannot generate a result *without delay*

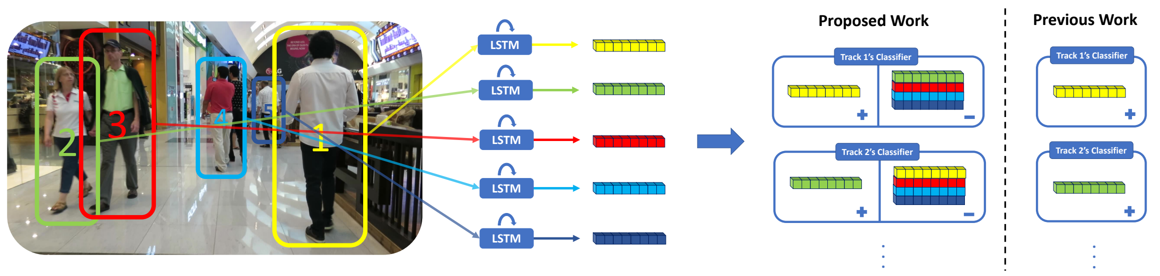


Figure 5.1. Existing recurrent neural network-based track classifiers used only matched detections for updating its appearance memory during tracking. This does not consider other objects in the scene (i.e. negative examples), which may have similar appearances. We propose to improve the predicted likelihood of such a classifier by augmenting its memory with appearance information about other tracks in the scene with multi-track pooling, leveraging the appearance information from the full set of tracks in the scene. The resulting classifier learns to adapt its prediction based on the information from other tracks in the scene.

after processing a new frame in the video. This is unfortunate, because real-time online tracking is critically important for numerous practical applications in AI. For example, in robotics applications such as autonomous driving, online processing is critical to support real-time decision-making.

Simple online tracking solutions based on matching appearance features between adjacent frames do not yield good performance. A better strategy is to use multiple frames to build up a memory [25, 26] which has the ability to store multiple appearances and match them to new detections automatically. However, when matching the current detection and object tracks, most current approaches look at tracks one at a time without considering them jointly. This approach does not have sufficient discriminative power to address the ambiguities arising when several similar targets move adjacent to each other in the video, e.g. multiple men in black suits. In this situation, either more subtle features need to be utilized for target discrimination, or the likelihood of the matching needs to be decreased to accommodate this uncertainty. This in turn requires the matching approach to utilize the appearance information *from other nearby tracks* in making a determination. The pairwise appearance models used in prior tracking methods lack the ability to make this comparison.

This chapter introduces a novel approach to appearance modeling that takes *all* of the tracked objects into account when matching detections. Suppose each track has a stored online memory about all of its past appearances, we propose a novel *multi-track pooling module* which stores a max-pooled version of the memory from all other tracks. This extension allows the appearance model to take into account **online** negative examples coming from other objects within the same video (see Fig. 5.1). We show that multi-track pooling greatly enhances the discriminative power of the appearance model for track scoring and improves overall performance.

We leverage the additional discriminative power of our matching approach to achieve online tracking performance by means of a simple, greedy data association method: Each track is matched with the detection that has the highest likelihood of belonging to the track. Such an association strategy is extremely efficient, resulting in fast performance. In this work we test the hypothesis that our discriminative appearance and motion modeling can enable this simple data association mechanism to achieve strong tracking performance.

Our online tracking framework additionally incorporates four components from recent tracking works. First, we utilize the Bilinear LSTM framework in Chapter 4 as the basis for our matching approach, due to its effective memory construction. Second, we incorporate training strategies that handle long tracks using truncated backpropagation through time [106], in contrast to our work in Chpater 4 and the prior work [25] in which the models were trained with short tracks. Third, we extend object tracks to frames without detections using a motion model, thereby compensating for missing detections. Fourth, we trained a bounding box coordinate corrector to correct bounding box coordinates, which is especially helpful for the extended tracks.

In summary, this chapter makes the following contributions:

1. A novel multi-track pooling module which enables a track classifier to take online negative examples into account during testing and thus adjust its prediction adaptively depending on the objects in the scene.

2. A training strategy that is adapted to train the proposed track pooling module by

utilizing within-batch dependencies among tracks and that enables long sequence training with the original full tracks instead of short, random tracks used in Chapter 4.

3. A real-time, online multi-object tracking algorithm that achieves state-of-the-art performance on standard tracking benchmarks.

## 5.2 Track Proposal Classifier

Tracking-by-detection approaches in multi-object tracking evaluate multiple track proposals [82, 45, 25] when finding correspondences between tracks and detections. Track proposals are typically constructed by extending existing tracks from the previous frame with new detections in the current frame. Let us denote $T_l(t)$ as the $l^{\text{th}}$ track at time $t$. Let $s(l)$ be the starting frame of the $l^{\text{th}}$ track and $d_t^l$ be the detection selected by the $l^{\text{th}}$ track at time $t$. We then write the $l^{\text{th}}$ track at time $t$ as $T_l(t) = \{d_{s(l)}^l,\ d_{s(l)+1}^l,\ \dots,\ d_{t-1}^l,\ d_t^l\}$. Recurrent networks are trained to output the following conditional probability:

$$f(d_t, T_l(t-1); \theta) = p(d_t \in T_l(t) | T_l(t-1)) \tag{5.1}$$

where $f(\cdot)$ and $\theta$ represent a neural network and its learnable parameters respectively. The inputs to the neural network are the detection-track pair $(d_t, T_l(t-1))$.

### 5.2.1 Bilinear LSTM

Vanilla LSTM equations are defined as follows [84]:

$$\begin{aligned}
&\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}; \mathbf{x}_t]), \quad \mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}; \mathbf{x}_t]), \\
&\mathbf{g}_t = \sigma(\mathbf{W}_g[\mathbf{h}_{t-1}; \mathbf{x}_t]), \quad \mathbf{o}_t = \tanh(\mathbf{W}_o[\mathbf{h}_{t-1}; \mathbf{x}_t]), \\
&\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t, \quad \mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)
\end{aligned} \tag{5.2}$$

where $[;]$ denotes concatenation of 2 column vectors. In the sequence labeling problem, $\mathbf{h}_t$ stores the information about the sequence and is fed into additional fully-connected or convolutional layers to generate the output. The input $\mathbf{x}_t$ and the LSTM memory $\mathbf{h}_{t-1}$ are combined by additive interactions in the above equations.

There are two issues with this formulation. First, the *matching* operation is usually more easily represented by multiplicative relationships instead of additive, an intuitive example being the inner product as a correlation metric. Second, it is difficult to store and clearly distinguish multiple different appearances in the same LSTM memory vector, but a track exhibiting multiple different appearances in different frames is very common in multi-target tracking. Bilinear LSTM solves these issues by introducing a new memory representation based on the multiplicative interaction between the input and the LSTM memory:

$$\begin{aligned} \mathbf{h}_{t-1} &= [\mathbf{h}_{t-1,1}^\top, \mathbf{h}_{t-1,2}^\top, ..., \mathbf{h}_{t-1,r}^\top]^\top \\ \mathbf{H}_{t-1}^{\text{reshaped}} &= [\mathbf{h}_{t-1,1}, \mathbf{h}_{t-1,2}, ..., \mathbf{h}_{t-1,r}]^\top \\ \mathbf{m}_t &= g(\mathbf{H}_{t-1}^{\text{reshaped}} \mathbf{x}_t) \end{aligned} \tag{5.3}$$

where $g(\cdot)$ is a non-linear activation function. A long vector $\mathbf{h}_{t-1}$ from LSTM is reshaped into a matrix $\mathbf{H}_{t-1}^{\text{reshaped}}$ before it being multiplied to the input $\mathbf{x}_t$. Hence, multiple memory vectors can be matched with the feature from the detection with an inner product. The new memory $\mathbf{m}_t$ is then used as input to the final layers to generate the output instead of $\mathbf{h}_t$. Note that the way that the LSTM memory $\mathbf{h}_{t-1}$ interacts with the input $\mathbf{x}_t$ is changed in Bilinear LSTM, while standard LSTM memory updates shown in Eq. (5.2) are used since the memory vector $\mathbf{h}_{t-1}$ is retained as in Eq. (5.3).

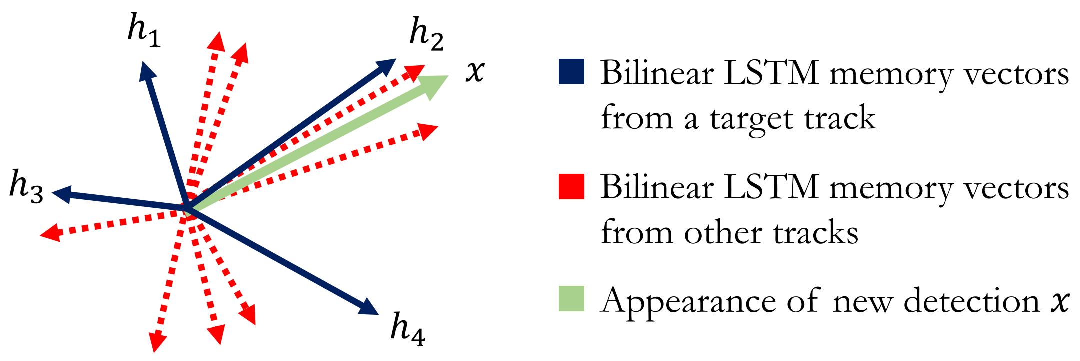
Bilinear LSTM bears some resemblance to the popular transformer model [107] in natural language processing in that both utilize a multiplicative relationship between the sequence and a new token, but they have some important differences. In a transformer model, the inner product is taken between the key of a new token with *all* previous tokens in the sequence. This makes them very memory-inefficient and unsuitable for online operations that span hundreds of frames in multi-object tracking. In Bilinear LSTM, the memory size is fixed, and the memory is updated automatically using the LSTM updating rule. This has the effect of automatically grouping similar appearances into the same row of the memory $\mathbf{H}_{t-1}^{\text{reshaped}}$ in Eq. (5.3), so that the memory size does not grow linearly with respect to the sequence length. Hence, we believe that Bilinear LSTM is a suitable choice for online tracking in which multiple appearances for each track need to be stored.

### 5.2.2 Application to Multi-object Tracking

When Bilinear LSTM is used in multi-object tracking, each track $T_l(t-1)$ will have its own LSTM memory $\mathbf{h}_{t-1}^l$ which is stored during the tracking process. All new detections at frame $t$ go through a CNN to generate their corresponding $\mathbf{x}_t$, which are then used to compare with $\mathbf{h}_{t-1}^l$ for all the tracks. When each detection has been scored with each track, the detections will be assigned to the existing tracks by either greedy assignment, or multiple hypothesis tracking (MHT) assignment. Finally, the features from the assigned bounding box are used as $\mathbf{x}_t^l$ to update the track memory with Eq. (5.2). The updated memory $\mathbf{h}_{t-1}^l$ will be stored and then the same process will be repeated in the next frame. All tracks share the same LSTM network as their appearances will be dynamically updated in the memory, hence there is no re-training needed for any new object.

### 5.2.3 Multi-Track Pooling Module

A limitation of the work in Chapter 4 and the previous work [25] is that only past appearances of the same track were considered, as only matched detections were inputted as $\mathbf{x}_t^l$ to update the LSTM memory at time $t$. However, during tracking, different targets can have similar appearances. For example, in pedestrian tracking, there could always be many people wearing similar white shirt and black pants, or black suits and black pants. These people need to be distinguished via more detailed features, such as shoes, objects in hand,



$h_1$
$h_2$
$x$
$h_3$
$h_4$
Bilinear LSTM memory vectors from a target track
Bilinear LSTM memory vectors from other tracks
Appearance of new detection $x$

Figure 5.2. Bilinear LSTM and the proposed improvements by multi-track pooling module. Bilinear LSTM stores multiple memory vectors for each track (in blue) so that a new detection $x_t$ can be matched with multiple templates via inner products. In this work, we improve on it by concatenating the memory with multiple stored memory vectors from other simultaneously tracked targets (in red) to serve as negative examples, so that the template matching process can take into account more subtle differences between the positive track and negative tracks (best viewed in color).

etc. Thus, simple appearance matching may not be discriminative enough.

We propose to extend the track scoring process to consider **all** the tracked objects, other than only the current object of concern. Instead of simply growing the memory to store more templates, which can hit a memory bottleneck, we jointly consider all the objects that have been tracked. Tracked objects are usually the ones most easily confused with the target (if a detection does not come from a track, then its appearance is likely significantly different from any tracked objects (e.g. pedestrians)). Hence, taking the appearance information about these objects into consideration could greatly improve the discriminative power of the tracker for each target (see Fig. 5.2).

In order to consider other tracked objects as well, we propose to modify Eq. (5.1) to:

$$f(d_t, T_{1:M}(t-1); \theta) = p(d_t \in T_l(t) | T_{1:M}(t-1)) \tag{5.4}$$

where $T_{1:M}(t-1) = \{T_1(t-1), T_2(t-1), ..., T_M(t-1)\}$ represents the existing tracks in the previous frame. In this chapter, we denote $M$ as the number of tracks in the previous frame $t-1$ and denote $N$ as the number of detections in the current frame $t$. [25] also trained recurrent neural networks to output this kind of conditional probability by using track interaction cues, but their interaction cues were the most recent locations of other tracks, not appearances of other tracks.

In Bilinear LSTM, each track in the previous frame is associated with a unique LSTM memory $\mathbf{h}^{l}_{t-1}$. When new detections $\mathbf{x}_t$ arrive in the current frame, we compute $\mathbf{m}_t$ in Eq. (5.3) between each of the $M$ existing tracks and each of the $N$ new detections. We denote $\mathbf{m}_t^{+}$ as the matching between $\mathbf{h}^{l}_{t-1}$ computed using the target track (i.e. the current object of concern) and $\mathbf{x}_t$ (as computed by Eq. (5.3)). For the other $M-1$ tracks in the scene, we denote $\mathbf{M}_t^{-}$ as a matrix in which each row represents $\mathbf{m}_t$ computed by a non-target object track (i.e. other objects in the scene) and the new detection ($\mathbf{x}_t$):

$$\mathbf{M}_t^{-} = [\mathbf{m}_{t,1}^{-}, \mathbf{m}_{t,2}^{-}, ..., \mathbf{m}_{t,M-1}^{-}]^{\top} \tag{5.5}$$

where $\mathbf{m}_{t,i}^{-}$ represents the matching between the $i$-th non-target object track and the new detection.

Here the main difficulty is that we have one positive track and an indefinite number of negative tracks. We propose to compress $M-1$ $\mathbf{m}_t$s using max pooling in order to obtain a fixed size memory representation $\mathbf{m}_t^{-}$ regardless of the number of the tracks in the previous frame:

$$\mathbf{m}_t^{-}(j) = \max_i \mathbf{M}_t^{-}(i,j) \tag{5.6}$$

where $\mathbf{m}_t^{-}(j)$ represents the $j$-th element in $\mathbf{m}_t^{-}$, and $\mathbf{M}_t^{-}(i,j)$ represents an element located in the $i$-th row and $j$-th column of $\mathbf{M}_t^{-}$. Thus, the $j$-th element in $\mathbf{m}_t^{-}$ is computed by taking the maximum of the $j$-th column of $\mathbf{M}_t^{-}$.

Since the Bilinear LSTM memory stores correlation responses between multiple templates and the input, applying max pooling to $\mathbf{M}_t^{-}$ allows us to detect high correlation

Table 5.1. Proposed Network Architecture for Track Proposal Classifier. The right two columns represent the multi-track pooling module where $(M-1)$ other tracks are processed.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Soft-max | 2 | | | | | | |
| FC | 2 | | | | | | |
| Concatenation ($\mathbf{m}_t^{\text{all}}$) | 16 | | | | | | |
| Matrix-vector Multiplication-relu ($\mathbf{m}_t^+$) | 8 | | | Max-pooling ($\mathbf{m}_t^-$) | 8 | | |
| | | | | Matrix-vector Multiplication-relu ($\mathbf{M}_t^-$) | $(M-1)\times 8$ | | |
| Reshape | $8\times 256$ | Reshape | $256\times 1$ | Reshape | $(M-1)\times 8\times 256$ | Reshape | $256\times 1$ |
| LSTM | 2048 | | | LSTM | $(M-1)\times 2048$ | | |
| FC-relu | 256 | FC-relu | 256 | FC-relu | $(M-1)\times 256$ | FC-relu | 256 |
| ResNet-50 | 2048 | ResNet50 | 2048 | ResNet-50 | $(M-1)\times 2048$ | ResNet50 | 2048 |
| $\mathbf{x}_{t-1}^+$ | $128\times 64\times 3$ | $\mathbf{x}_t$ | $128\times 64\times 3$ | $\mathbf{x}_{t-1,i}^-$ | $(M-1)\times 128\times 64\times 3$ | $\mathbf{x}_t$ | $128\times 64\times 3$ |

responses generated by non-target object tracks for the detection which is currently considered. Thus, values in $\mathbf{m}_t^-$ can show whether any of the non-target object tracks has a similar appearance to that of the current detection. We obtain the final memory representation for a track classifier by concatenating $\mathbf{m}_t^+$ and $\mathbf{m}_t^-$ as:

$$\mathbf{m}_t^{\text{all}} = \left[\left(\mathbf{m}_t^+\right); \left(\mathbf{m}_t^-\right)\right] \tag{5.7}$$

The network runs a fully-connected layer after $\mathbf{m}_t^{\text{all}}$ followed by softmax, that outputs the binary decision that determines whether the new detection belongs to the target track ($\mathbf{m}_t^+$).

Table 5.1 shows the proposed network architecture. In practice we compute the Bilinear LSTM memories for all the tracks and then construct $M_t^-$ by simply stacking the precomputed memory vectors into a matrix, which can be done efficiently. We adopt ResNet 50 [87] as our convolutional neural network and use Bilinear LSTM proposed in Chapter 4 as our recurrent network. In addition to the proposed appearance model, we adopt the motion model presented in Chapter 4 as the learned motion model. It is a binary classifier based on a traditional LSTM that receives the bounding box coordinates as input. We combined the appearance model and motion model to form a joint model that utilizes both appearance and motion cues. We used this joint model to generate the results in Table 5.11, 5.12, and 5.13. More details about the joint model's network architecture can be found in Sec. 5.5.

## 5.3 Training

In this section, we describe the training strategy for the proposed neural network architecture by sampling mini-batches so that tracks within the mini-batch are correlated, in order for the multi-track pooling module to train effectively. We explain the method that generates our training data consisting of both *actual* multi-object tracking episodes and *random* tracking episodes from public multi-object tracking datasets [90, 80] and then present our choice of loss function.

### 5.3.1 Actual Tracking Episodes as Training Data

We generate actual tracking episodes by processing ground truth track labels sequentially and generating track proposals every frame in an online manner. Specifically, when we have $M$ tracks in the previous frame and $N$ detections in the current frame, we generate $MN$ track-detection pairs in the current frame by considering all possible pairs between the existing tracks and the new detections. Each proposal is associated with a binary label where positive label indicates a match and negative represents that the track and detection belong to different objects. We use these $MN$ track proposals as a mini-batch for each training iteration and repeat this process until we process all the frames in video. Then we move on to the next training video. This process is repeated over all training videos during training.

**Truncated backpropagation through time.** As we process training videos sequentially, tracks become too long to fit into GPU memory. One way to avoid this is to split tracks into shorter ones, which is similar to the training data used in Chapter 4 and [25] for recurrent models. However, this is sub-optimal because a recurrent model would only be able to learn temporal structures with an extent limited by the maximum sequence length that the model sees during training.

In order to address this issue, we adopt truncated backpropagation through time (BPTT)

[106]. In truncated BPTT, the maximum number of time steps where the loss is backpropagated is limited by a fixed-size time window. However, entire sequences are processed by a recurrent model through multiple training iterations by a moving time window. Since long sequences are processed through multiple training iterations, the recurrent memories that store the sequence information need to be shared across the training iterations. This allows the model to access the temporal context beyond the temporal extent which is determined by the size of the time window. By training a recurrent model with the original long tracks, the model has a better chance to learn how to exploit long-term information when solving the track proposal classification problem.

### 5.3.2 Random Tracking Episodes as Training Data

We also use short random track proposals, which is similar to the training data used in Chapter 4 and [25], as our additional training data. We generate short random track proposals as follows. At each training iteration, we first pick a video where we generate tracks and randomly select the start frame and end frame for obtaining track proposals. We take all the tracks that exist in the end frame. Denote $N_{\text{max}}$ as the maximum number of tracks that we use to construct random track proposals. If the number of the tracks in the end frame is greater than $N_{\text{max}}$, we randomly pick $N_{\text{max}}$ tracks among them. In order for each mini-batch to have tracks of different lengths, we randomly clip the selected tracks such that each track in the batch start in different frames.

We take $N$ detections from the selected tracks in the end frame and take $M$ tracks from the previous frame to form track proposals ($M < N$ when new tracks are born in the selected end frame. Otherwise, $M = N$). We generate $MN$ track proposals by considering all possible matchings between these two. Thus, $M$ track proposals are associated with a positive label (same object), and $MN - M$ proposals are associated with a negative label.

### 5.3.3 Loss Function

For each mini-batch, there are $MN$ track proposals, each of which incurs a loss value. Thus, our cross-entropy loss is written as:

$$L(t) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \alpha_{ij}(t) L_{ij}(t) \tag{5.8}$$

where $t$ represents a frame number in video, and $\alpha_{ij}$ is a weighting factor. The cross-entropy loss term for each training example $L_{ij}(t)$ is defined as:

$$L_{ij}(t) = \begin{cases} - & \log p(d_{jt} \in T_i(t) | T_{1:M}(t-1)), \text{if} y_{ij}(t) = 1 \\ - & \log(1 - p(d_{jt} \in T_i(t) | T_{1:M}(t-1))), \text{otherwise} \end{cases} \tag{5.9}$$

where $d_{jt}$ represents a $j$th detection in $t$, and $y_{ij}(t)$ is a ground truth binary label representing whether $d_{jt}$ belongs to a track $T_i(t)$ or not. For the weighting factor $\alpha_{ij}(t)$, we adopted the weighting factor of Focal loss [108] to address the class imbalance of our training data (i.e. there are much more negative examples than positive examples). The weighting factor is then written as:

$$\alpha_{ij}(t) = \begin{cases} \beta_+ (1 - p(d_{jt} \in T_i(t) | T_{1:M}(t-1)))^2, & \text{if } y_{ij}(t) = 1 \\ \beta_- (p(d_{jt} \in T_i(t) | T_{1:M}(t-1)))^2, & \text{otherwise} \end{cases} \tag{5.10}$$

where $\beta_+$ and $\beta_-$ are class-specific constant weights which are found empirically as suggested in [108]. For positive labels, we used $\beta_+ = 4$, and, for negative labels (i.e. $d_{jt} \notin T_i(t)$), we used $\beta_- = 1$.

## 5.4 Tracking Algorithm

In this work, we chose to use the greedy association because it is the fastest association algorithm, and our training setting simulates such a greedy association algorithm (i.e. the loss value was calculated for each track independently and then averaged across all tracks in a mini-batch). The greedy data association algorithm runs in $O(MN)$ time. A more complicated scheme would require additional computational complexity (e.g. the Hungarian algorithm runs in $O(MN^2)$) and would require more careful parameter tuning for the tracker.

### 5.4.1 Greedy Data Association

Our greedy data-association works as follows. It initializes new tracks with the detections in the first frame. The tracker generates and stores the LSTM memory $\mathbf{h}_{t-1}$ for each of the tracks. When new detections $\mathbf{x}_t$ arrive in the next frame, we compute the association likelihood for every possible track-detection pair using the track proposal classifier. We set the threshold for the association likelihood to 0.5. The data association problem is then solved in a greedy manner starting from the highest matching likelihood. The tracks are updated with the newly assigned detections, and new tracks are born from the detections which are not associated with any of the existing tracks. The tracker updates the LSTM memory from $\mathbf{h}_{t-1}$ to $\mathbf{h}_t$ for every track according to the data association result. This process is repeated until all the video frames are processed.

We adopt a simple track termination strategy which terminates tracks if they have more missing detections than actual detections or the number of consecutive missing detections becomes larger than a threshold $N_{\text{miss}}$. The terminated tracks will not be used in the data association process anymore and will not be used for computing non-target object memories either.

### 5.4.2 Track Extension and Bounding Box Correction

In online tracking, we attempt to extend a track by generating additional detections. When there is no detection in a predicted target location in the current frame, we generate a new detection bounding box using the location predicted by a Kalman filter and then use our track proposal classifier to decide whether or not the newly generated detection belongs to the current target. Since we rely on motion cues when generating these additional detections, the bounding box coordinates of the newly generated detections might not be accurate. Thus, we train a bounding box coordinate corrector which predicts the correct bounding box coordinates based on the CNN features, which is similar to the bounding box regression module presented in [109]. More details are included in Sec. 5.5.

## 5.5 Additional Architecture Details

In this section, we present the network architectures used in all of our experiments.

### 5.5.1 Track Proposal Classifier

Table 5.4 shows the network architecture for our joint appearance and motion model with the proposed multi-track pooling module. We used this network to test the proposed approach on the MOT Challenge Benchmarks in Table 5.11, 5.12, and 5.13. Table 5.5 shows the appearance baseline model used in Table 5.7, 5.8, 5.9, and 5.10.

### 5.5.2 Track Proposal Classifier Input

For the appearance models, an object image (i.e. object detector output) is used as input. We first resize the object image to $64 \times 128$ (width, height) and then subtract the ImageNet mean from the image. For the motion model, a location (top left corner) and scale (width, height) of a detection bounding box is used as input. The motion model input is thus represented by a 4-dimensional vector $(x^{\text{topleft}}, y^{\text{topleft}}, w, h)$. We normalize

Table 5.2. Split 1

| Training Set |
|---|
| MOT17 - {02, 04, 05, 09, 10, 11, 13}<br>MOT15 - {PETS09-S2L1, ETH - (Sunnyday, Bahnhof), TUD - (Campus, Stadtmitte), KITTI-(13, 17)},<br>ETH - (Jelmoli, Seq01), KITTI - (16, 19),<br>PETS09-S2L2, TUD-Crossing, AVG-TownCentre |
| Validation Set |
| MOT 19 Challenge - {01, 02, 03, 05} |
| Test Set |
| MOT17 - {01, 03, 06, 07, 08, 12, 14} |

Table 5.3. Split 2

| Training Set |
|---|
| MOT15 - {PETS09-S2L1, ETH - (Sunnyday, Bahnhof), TUD - (Campus, Stadtmitte), KITTI-(13, 17)},<br>ETH - (Jelmoli, Seq01), KITTI - (16, 19),<br>PETS09-S2L2, TUD-Crossing, AVG-TownCentre |
| Validation Set |
| MOT17 - {02, 04, 05, 09, 10, 11, 13} |
| Test Set |
| MOT17 - {01, 03, 06, 07, 08, 12, 14} |

the input vector using the image size so the final form of input for the motion model is $(\frac{x^{\text{topleft}}}{\text{image width}}, \frac{y^{\text{topleft}}}{\text{image height}}, \frac{w}{\text{image width}}, \frac{h}{\text{image height}})$.

## 5.5.3 Bounding Box Coordinate Corrector

Following the bounding box regressor presented in [109], we regress four scalars that corrects the location and scale of the original bounding box from a cropped image. We also have an additional prediction head that classifies false positive detections using the same input image. The network architecture that we used for the box corrector is shown in Table 5.6. We utilized the same CNN as the one used in a track proposal classifier. Specifically, once the track classifier was trained, we froze all the CNN weights and then trained the additional linear layers in the two prediction heads from scratch. We used raw DPM detections provided by the MOT16 Challenge organizers as training data to train this module.

Table 5.4. The appearance + motion model used to generate the results in Table 5.11, 5.12, and 5.13. The number of non-target object tracks used in the multi-track pooling module is represented by $M-1$.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Soft-max | 2 | | | | | | | | |
| FC | 2 | | | | | | | | |
| FC-relu | 24 | | | | | | | | |
| Concatenation | 24 | | | | | | | | |
| 2× FC-relu | 16 | | | | | | | 2× FC-relu | 8 |
| Concatenation ($\mathbf{m}_t^{\text{all}}$) | 16 | | | | | | | | |
| Matrix-vector Multiplication-relu ($\mathbf{m}_t^+$) | 8 | | | Max-pooling ($\mathbf{m}_t^-$) | 8 | | | | |
| | | | | Matrix-vector Multiplication-relu ($\mathbf{M}_t^-$) | $(M-1)\times 8$ | | | | |
| Reshape | $8\times 256$ | Reshape | $256\times 1$ | Reshape | $(M-1)\times 8\times 256$ | Reshape | $256\times 1$ | | |
| LSTM | 2048 | | | LSTM | $(M-1)\times 2048$ | | | FC-relu | 8 |
| FC-relu | 256 | FC-relu | 256 | FC-relu | $(M-1)\times 256$ | FC-relu | 256 | LSTM | 64 |
| ResNet-50 | 2048 | ResNet50 | 2048 | ResNet-50 | $(M-1)\times 2048$ | ResNet50 | 2048 | FC-relu | 64 |
| $\mathbf{x}_{t-1}^+$ | $128\times 64\times 3$ | $\mathbf{x}_t$ | $128\times 64\times 3$ | $\mathbf{x}_{t-1,i}^-$ | $(M-1)\times 128\times 64\times 3$ | $\mathbf{x}_t$ | $128\times 64\times 3$ | $\mathbf{x}_t^{\text{location, scale}}$ | 4 |

Table 5.5. The baseline appearance model that we compared with our proposed appearance model.

| | | | |
|---|---|---|---|
| Soft-max | 2 | | |
| FC | 2 | | |
| Matrix-vector Multiplication-relu ($\mathbf{m}_t$) | 8 | | |
| Reshape | $8\times 256$ | Reshape | $256\times 1$ |
| LSTM | 2048 | | |
| FC-relu | 256 | FC-relu | 256 |
| ResNet-50 | 2048 | ResNet50 | 2048 |
| $\mathbf{x}_{t-1}$ | $128\times 64\times 3$ | $\mathbf{x}_t$ | $128\times 64\times 3$ |

Table 5.6. Bounding Box Coordinate Corrector. ResNet-50 is shared with the appearance model.

| | | | |
|---|---|---|---|
| FC | 4 | Soft-max | 2 |
| 2× FC-relu | 512 | 2× FC-relu | 512 |
| Reshape | 16384 | | |
| ResNet-50 (Block4) | $4\times 2\times 2048$ | | |
| $\mathbf{x}_{t-1}^+$ | $128\times 64\times 3$ | | |

## 5.6 Additional Training Details

In this section, we describe details about the training settings that we used for our experiments.

### 5.6.1 Dataset

Table 5.2 and 5.3 shows the training, validation, and test sets that we used in this chapter.

### 5.6.2 Training Setting - Appearance Model

We used the SGD optimizer with the initial learning rate of 0.005 for Bilinear LSTM and the initial learning rate of 0.0005 for ResNet 50 (pre-trained on ImageNet). We trained the model with the initial learning rate for the first 4 epochs ($\sim$ 120k iterations), and then reduced the learning rate with the decay factor of 0.1 for the next 4 epochs and reduced it one more time for the last 4 epochs ($\sim$ 360k iterations in total).

For the actual tracking episodes, we used truncated BPTT with a temporal window size of 10. We used all the ground truth tracks in the current frame as our training data so the mini-batch size in this case was equal to the number of ground truth tracks in the current frame.

For the random tracking episodes, we used 40 frames as the maximum frame gap in the randomly selected start and end frame. Thus, each mini-batch could contain a track whose length is up to 40. Due to the limited GPU memory, the number of tracks for random tracking episodes was limited by $N_{\text{max}}$. We used $N_{\text{max}} = 8$ in our experiments.

### 5.6.3 Training Setting - Motion Model

We used the Adam optimizer [96] with the initial learning rate of 0.001. We trained the motion model with the initial learning rate for the first 4 epochs ($\sim$ 120k iterations), and then reduced the learning rate with a decay factor of 0.1 for the next 2 epochs and reduced it one more time for the last 6 epochs ($\sim$ 360k iterations in total).

### 5.6.4 Training Setting - Appearance and Motion Model

We first trained the appearance and motion models separately as described above before jointly training the model presented in Table 5.4. Thus, new layers (i.e. top five rows in Table 5.4) were trained from scratch, and the rest of the layers (except for ResNet 50 in which all the weights were frozen) was fine-tuned from the pre-trained models. We used the SGD optimizer with the initial learning rate of 0.005 for the new layers and 0.0005 for the pre-trained layers. We trained the model with the initial learning rate for the first 2 epochs ($\sim$ 60k iterations), and then reduced the learning rate with the decay factor of 0.1 for the next 2 epochs ($\sim$ 120k iterations in total).

When we trained the joint model, we realized that the appearance features can be stronger than the motion features in the beginning of the training. As a result, our resulting model heavily relied on the appearance features, often ignoring the motion features. In order to make our model balance between these two types of features, we placed a dropout layer on the appearance features right before the concatenation layer (i.e. top $6^{\text{th}}$ row on the appearance side in Table 5.4). In the beginning of the training, we randomly dropped appearance features and then gradually decreased the drop rate as the training proceeded. This trick prevented the joint model from relying too much on the appearance cues in the early training stage. In our experiments, we used 0.9 as the drop rate for the first $\sim$19k iterations, 0.6 for the next $\sim$10k iterations, 0.3 for the next $\sim$9k iterations, and 0.0 for the rest of the training.

### 5.6.5 Online Hard Example Mining

We found that online hard example mining can improve the model performance. We trained all the model with all the training examples for the first two epochs ($\sim$60k iterations). We trained the models for the remaining epochs with top $k$ hard examples (i.e. $k$ examples that incurred high loss values) in the mini-batch. We used $k = 30$ in our experiments. Note that the actual tracking episodes that we generated as our training data enabled effective online hard example mining since each mini-batch contained all possible matchings between the tracks and the new detections in the selected frame.

### 5.6.6 Missing Detection Augmentation

We randomly drop the bounding boxes in the tracks for missing detection augmentation. For each track in each mini-batch, we randomly choose the missing detection rate from a probability between 0.1 and 0.9. After selecting the missing detection rate, we randomly drop the bounding boxes in the selected track according to the selected rate.

### 5.6.7 Noisy Track Augmentation

In addition to using the ground truth tracks as the training data, we also generate tracks from noisy object detections. Given ground truth tracks and noisy detections, one can assign correct track IDs to noisy detections by finding an assignment that maximizes the Intersection over Union score (IoU) between ground truth tracks and object detections. The MOT Challenge Benchmark provides public object detections from three detectors (DPM [88], FRCNN [79], SDP [89]). Thus, we generate three additional sets of noisy tracks constructed from these public detections. Note that the track-detection assignments are optimal although the resulting tracks are noisier than the original ground truth tracks. Localization and missing detection errors caused by the object detector are embedded naturally in such tracks, which can potentially help the track classifier to generalize better in testing when in noisy detections are used as input to the tracker.

## 5.7 Experiments

We tested the proposed method on the MOT 16 and MOT 17 (Multiple Object Tracking) Challenge Benchmark [80]. We train on the MOT 17 training sequences, as well as additional public training sequences including 7 MOT 15 training sequences [90] which were not included in MOT 17 training/testing sequences to our training data. In contrast to several recent work [45, 25], we did not utilize the Person Re-identification datasets [94, 93] to pretrain our CNN.

For our ablation study, we used two validation sets. The first one is the MOT 19 Challenge training sequences [110] which have 2,390 annotated object tracks. In contrast to the MOT 17 sequences, this new challenge dataset provides heavily crowded scenes captured in new environments, which makes it good for validating the proposed appearance model.

Note that this dataset was only used for our ablation study and thus was not used to train our model in Table 5.11, 5.12, and 5.13. The second validation set is the MOT 17 Challenge training sequences which have 512 annotated object tracks. When the second validation set is used, we used the MOT 15 training sequences as our training data. See Sec. for the video sequence names used in the training, validation, and test sets.

We used the standard MOT metrics such as IDF1 [95], IDS, MOTA [67], Mostly Tracked (MT), Mostly Lost (ML), and Fragmentation (Frag) [80] for performance comparison in our experiments.

### 5.7.1 Ablation Study

In the ablation study, we show the effectiveness of the proposed multi-track pooling module by comparing its performance to the original Bilinear LSTM. Firstly, we tested their performance without using any motion cues during tracking. Secondly, we used motion cues during tracking by adopting a simple motion gating strategy that allows detections that are close to the current track to be considered as a possible matching.

**Data Association.** We ran the greedy data association algorithm described in the previous section with the following hyperparameter setting: 0.5 as the association threshold and $N_{\text{miss}} = 60$. In the ablation study, we did not interpolate missing detections in the final tracks using our track extension module in order to make the MOTA scores close across different methods. This allowed us to compare other tracking metrics in a fairer setting.

**Comparison with Bilinear LSTM.** We examined the effect of the proposed multi-track pooling module on the tracking performance. For this ablation study to be fair, both Bilinear LSTM and our method were trained on the same training set with the same training setup described in Sec. . Table 5.7 and 5.8 show the tracking results when no motion cues were used during tracking. In this case, the appearance model needed to do the heavy lifting. Table 5.9 and 5.10 show the tracking results when a simple motion gating strategy was applied. Bilinear LSTM with the multi-track pooling module consistently outperformed

Table 5.7. Performance comparison on MOT 19 train sequences (val1) when motion gating is not used.

| Method | MOTA | IDF1 | IDS | MT | ML | Frag |
|---|---|---|---|---|---|---|
| B-LSTM | 44.8 | 31.3 | 15,367 | 12.6 | 27.0 | 38,182 |
| Ours | 44.9 | **35.0** | **11,940** | 12.7 | 27.5 | **37,017** |

Table 5.8. Performance comparison on MOT 17 train sequences (val2) when motion gating is not used.

| Method | MOTA | IDF1 | IDS | MT | ML | Frag |
|---|---|---|---|---|---|---|
| B-LSTM | 49.1 | 52.5 | 1,112 | 21.1 | 31.9 | 1,066 |
| Ours | 49.4 | **53.9** | **809** | 21.1 | 31.5 | 1,070 |

Table 5.9. Performance comparison on MOT 19 train sequences (val1) when the simple motion gating strategy is used.

| Method | MOTA | IDF1 | IDS | MT | ML | Frag |
|---|---|---|---|---|---|---|
| B-LSTM | 45.1 | 39.6 | 9,137 | 12.6 | 27.6 | 36,379 |
| Ours | 45.0 | **40.5** | **7,873** | 12.6 | 28.1 | **35,169** |

Table 5.10. Performance comparison on MOT 17 train sequences (val2) when the simple motion gating strategy is used.

| Method | MOTA | IDF1 | IDS | MT | ML | Frag |
|---|---|---|---|---|---|---|
| B-LSTM | 49.3 | 56.7 | 847 | 21.1 | 32.1 | 1,038 |
| Ours | 49.6 | 56.8 | **616** | 21.1 | 32.1 | 1,040 |

the original Bilinear LSTM on IDF1, IDS and Fragmentations, showing its effectiveness in multi-object tracking on both of our validation sets.

### 5.7.2 MOT Challenges

We evaluated both the online and near-online versions of our tracker for the MOT 17/16 Benchmarks. In the online version, we utilized the track extension module described in the previous section to recover missing detections (except for the tracker in Table 5.11 in which we turned off both the extension module and the bounding box corrector). In the near-online version, we performed local track smoothing instead for recovering missing detections. For the second case, we denote the method as near-online in Table 5.12 and

Table 5.11. MOT 17 Challenge (with trackers that utilized public detections + Tracktor [113]).
Note that we used **bold** for the best number and blue color for the second-best number.

| Method | Type | IDF1 | MOTA | IDS | MT | ML | Frag | FP | FN | Hz |
|---|---|---|---|---|---|---|---|---|---|---|
| GSM-Tracktor [112] | online | 57.8 | **56.4** | 1,485 | **22.2** | **34.5** | **2,763** | 14,379 | **230,174** | 8.7 |
| Tracktor++v2 [113] | online | 55.1 | 56.3 | 1,987 | 21.1 | 35.3 | 3,763 | 8,866 | 235,449 | 1.5 |
| TrctrD17 [111] | online | 53.8 | 53.7 | 1,947 | 19.4 | 36.6 | 4,792 | 11,731 | 247,447 | 4.9 |
| Tracktor++ [113] | online | 52.3 | 53.5 | 2,072 | 19.5 | 36.6 | 4,611 | 12,201 | 248,047 | 1.5 |
| **Ours** | online | **60.4** | 55.9 | **1,188** | 20.5 | 36.7 | 4,187 | **8,653** | 238,853 | **24.8** |

5.13 since local track smoothing requires lookahead frames.

Recent approaches [111, 112] utilized Tracktor [113] to first refine public detections, which resulted in higher scores due to more accurate detections. In order to compare with these recent approaches, we also used public detections processed by Tracktor as input to our tracker and presented the comparison in Table 5.11 separately. It can be seen that our approach significantly improves the IDF1 score and identity switches over other online tracktor-based approaches, besides being at least 3 times faster than the nearest competitor. Note that this does not include the processing time spent by Tracktor.

In Table 5.12 and 5.13, we did not utilize Tracktor and compared our results with other online and near-online trackers which did not utilize Tracktor. Again our greedy tracker is the fastest among the top performing trackers and our performance is comparable with the best trackers. Considering its simplicity and speed, we believe our method demonstrates strong state-of-the-art performance on the MOT Challenge.

In near-online trackers, we significantly improve over our baseline MHT-bLSTM on both IDF1 (by $7.5\%$) and MOTA (by $12.8\%$), obtaining the best performance in near-online tracking. We also have the smallest amount of identity switches and fragmentations and the most objects that are mostly tracked in MOT 17. Note that these are obtained with a greedy data association algorithm and only local smoothing is added on top of the online tracker performance, hence the speed is even faster than the online version since local smoothing removed more false positives.

Table 5.12. MOT 17 Challenge (Published online and near-online methods using public detections).

| Method | Type | IDF1 | MOTA | IDS | MT | ML | Frag | FP | FN | Hz |
|---|---|---|---|---|---|---|---|---|---|---|
| STRN-MOT17 [50] | online | **56.0** | 50.9 | 2,397 | 18.9 | 33.8 | 9,363 | 25,295 | 249,365 | 13.8 |
| DMAN [51] | online | 55.7 | 48.2 | **2,194** | **19.3** | 38.3 | 5,378 | 26,218 | 263,608 | 0.3 |
| MOTDT17 [98] | online | 52.7 | 50.9 | 2,474 | 17.5 | 35.7 | 5,317 | 24,069 | 250,768 | 18.3 |
| AM-ADM17 [114] | online | 52.1 | 48.1 | 2,214 | 13.4 | 39.7 | **5,027** | 25,061 | 265,495 | 5.7 |
| HAM-SADF17 [115] | online | 51.1 | 48.3 | **1,871** | 17.1 | 41.7 | **3,020** | **20,967** | 269,038 | 5.0 |
| PHD-GSDL17 [99] | online | 49.6 | 48.0 | 3,998 | 17.1 | 35.6 | 8,886 | 23,199 | 265,954 | 6.7 |
| FAMNet [116] | online | 48.7 | **52.0** | 3,072 | 19.1 | **33.4** | 5,318 | **14,138** | 253,616 | 0.0 |
| **Ours** | online | 54.9 | 51.5 | 2,563 | **20.5** | 35.5 | 7,745 | 29,623 | **241,618** | **20.1** |
| MHT-bLSTM [77] | near-online | 51.9 | 47.5 | 2,069 | 18.2 | 41.7 | 3,124 | 25,981 | 268,042 | 1.9 |
| EDMT17 [101] | near-online | 51.3 | 50.0 | 2,264 | 21.6 | 36.3 | 3,260 | 32,279 | 247,297 | 0.6 |
| MHT-DAM [56] | near-online | 47.2 | 50.7 | 2,314 | 20.8 | 36.9 | 2,865 | **22,875** | 252,889 | 0.9 |
| **Ours** | near-online | **55.8** | **53.6** | **1,845** | **23.4** | **34.5** | **2,299** | 23,669 | **236,226** | **22.7** |

Table 5.13. MOT 16 Challenge (Published online and near-online methods using public detections).

| Method | Type | IDF1 | MOTA | IDS | MT | ML | Frag | FP | FN | Hz |
|---|---|---|---|---|---|---|---|---|---|---|
| STRN-MOT16 [50] | online | 53.9 | **48.5** | 747 | 17.0 | **34.9** | 2,919 | 9,038 | 84,178 | 13.5 |
| DMAN [51] | online | **54.8** | 46.1 | 532 | **17.4** | 42.7 | 1,616 | 7,909 | 89,874 | 0.3 |
| MOTDT [98] | online | 50.9 | 47.6 | 792 | 15.2 | 38.3 | 1,858 | 9,253 | 85,431 | 20.6 |
| STAM16 [102] | online | 50.0 | 46.0 | **473** | 14.6 | 43.6 | 1,422 | 6,895 | 91,117 | 0.2 |
| RAR16pub [26] | online | 48.8 | 45.9 | 648 | 13.2 | 41.9 | 1,992 | 6,871 | 91,173 | 0.9 |
| KCF16 [117] | online | 47.2 | 48.8 | 648 | 15.8 | 38.1 | **1,116** | 5,875 | 86,567 | 0.1 |
| AMIR [25] | online | 46.3 | 47.2 | 774 | 14.0 | 41.6 | 1,675 | **2,681** | 92,856 | 1.0 |
| **Ours** | online | 53.5 | 48.3 | 733 | 17.0 | 38.7 | 2,349 | 9,799 | **83,712** | **21.0** |
| NOMT [82] | near-online | **53.3** | 46.4 | **359** | **18.3** | 41.4 | **504** | 9,753 | 87,565 | 2.6 |
| EDMT [101] | near-online | 47.9 | 45.3 | 639 | 17.0 | **39.9** | 946 | 11,122 | 87,890 | 1.8 |
| MHT-bLSTM [77] | near-online | 47.8 | 42.1 | 753 | 14.9 | 44.4 | 1,156 | 11,637 | 93,172 | 1.8 |
| MHT-DAM [56] | near-online | 46.1 | 45.8 | 590 | 16.2 | 43.2 | 781 | **6,412** | 91,758 | 0.8 |
| **Ours** | near-online | 52.5 | **49.9** | 579 | **19.7** | **38.6** | 674 | 7,111 | **83,676** | **23.8** |

## 5.8 Conclusion

In this chapter, we introduce a novel multi-track pooling module that enables joint updating of appearance models using all tracks, thereby improving matching reliability when targets are similar in appearance. We propose a novel training strategy for track pooling that utilizes within-batch dependencies among tracks and supports training over long sequences. The resulting tracker is based on a Bilinear LSTM architecture and performs greedy data association. With this simple approach, it achieves real-time tracking performance with an accuracy equivalent to state-of-the-art trackers that are significantly slower.

# CHAPTER 6
# CONCLUSION AND FUTURE WORK

Data association is one of the key problems in the tracking-by-detection paradigm. Designing an effective track scorer or classifier that can accurately rank track proposals in data association is very important because the quality of the tracking output is largely determined by the data association result in the tracking-by-detection approaches. Traditionally, designing such a track classifier involved developing hand-crafted features and tuning model hyperparameters that control contributions of different types of the features manually. Recent developments in deep learning have enabled the track classifier to be learned in an end-to-end fashion, mostly removing the manual feature engineering and making the resulting track classifier more effective. The improvements in track classifiers have also made the entire tracking-by-detection pipeline much faster and simpler by removing the need for running complicated association algorithms in most cases which often come with a large number of parameters to be tuned.

## 6.1 Limitations

The proposed approaches in this dissertation have the following limitations. First, it is assumed that there exists only one detection bounding box for each object in a frame. Thus, if there were multiple object detection bounding boxes for a single object in a frame, the proposed methods would associate only one of them with the track from the previous frame and initiate new tracks using the other detections that are not associated with the existing tracks, even though these bounding boxes still represent the same object. The additional tracks generated by these duplicate bounding boxes often end up being short, which can be removed via post-processing. However, if the detector outputs multiple bounding boxes for a single object consistently over time, the proposed tracker is likely to generate multiple tracks for a single object, which will be counted as false positive tracks during evaluation. This issue was addressed via post-processing in the proposed approaches in this dissertation, but there are existing multi-object tracking frameworks that can handle this issue effectively [118, 45, 119]. Second, in object tracking, tracking models are often fine-tuned with online training examples which are collected on the fly [18, 54, 27] in order to make the appearance models generalize better on test sequences. The work presented in Chapter 5 also utilizes online track examples to improve the appearance model's discriminative power on test sequences, but the appearance model learns how to generate its memory by jointly considering online track examples during offline training. While our work has an advantage in terms of speed (i.e. no additional training time during testing), the works that utilize online learning for deep neural networks [18, 54, 27] may have an advantage in terms of learning more discriminative appearance features via fine-tuning pre-trained CNNs.

## 6.2 Future Work

### 6.2.1 Unifying Object Detection and Tracking

Most top-performing tracking-by-detection approaches [52, 50, 27] do not utilize an end-to-end trainable architecture that takes raw video as input and generates tracks directly from the input video. Instead, the object detector is run first, and then the tracker takes the output from the detector as input and generates the object tracks by grouping the input detections into tracks. This makes the performance of most MOT methods heavily dependent on the object detector performance. Also, this makes the process of generating object tracks in video more complicated than necessary by requiring the users to run two different models separately which are developed independent from each other. Unifying object detector and tracker makes the entire multi-object tracking pipeline much more streamlined and faster. Also, it can potentially lead to higher detection recall rates as well. Specifically, once the object is detected in the first frame, the tracking prediction head can be utilized to detect the same object in subsequent frames by tracking it. Two recent works [120, 113] have pioneered in this direction by having a shared backbone network that extracts features from input images for both detection and tracking. The features from the shared backbone network are then sent to separate prediction heads that generate detection and tracking outputs respectively. In these prior works, simple tracking models that encode the most recent locations or appearances of the tracks were used. Thus, unifying an object detection model with more complicated tracking models that effectively reason long-term motion and appearance cues can be the next step. It would also be interesting to examine the interactions between the detection model and the tracking model when they are jointly trained in order to see if the joint training improves the performance on both tasks, or vice-versa.

### 6.2.2 Multi-object Tracking (MOT) Loss Functions

Computing important MOT evaluation metrics such as MOTA [67] and IDF1 [95] involve non-differentiable operations, so one cannot train a tracking model with a loss function that maximizes the model's performance for these MOT metrics. Thus, tracking models are trained with proxy losses such as a binary classification loss, regression loss, and triplet loss which are not directly related to the MOT evaluation metrics. Thus, designing a new loss function by approximating important MOT evaluation metrics using differentiable operations can address the mismatch between the loss function and evaluation metrics in multi-object tracking. Two recent works [111, 49] proposed new loss functions that are differentiable proxies for MOTA, MOTP [111], and IDF1 [49] and showed that minimizing these new loss functions leads to better performance on the target metrics. However, [49] also showed that optimizing for one metric can decrease the models' performance on other MOT metrics. Thus, designing new MOT loss functions that can balance well across multiple MOT evaluation metrics can be an interesting, future research direction.

### 6.2.3 Sequence Model for MOT

In multi-object tracking, LSTM-based architectures have been a popular choice for processing object tracks for track scoring and classification [49, 27, 25, 46]. Due to its recurrent memory update, LSTM provides an efficient memory architecture for online and near-online trackers (i.e. the model processes only one detection in each frame). Recently, the Transformer [107] has become the most popular sequence model in the field of natural language processing, replacing the previous popular models such as LSTM and GRU in many language tasks. One downside of the Transformer in the context of MOT is that its memory size grows linearly with respect to the length of the input sequence. This makes the original Transformer formulation unsuitable for fast, online tracking in which the model needs to process object tracks that span hundreds of frames. Thus, exploring possible modifications in the Transformer in order to make its memory size more manageable while maintaining its representational power for its application in MOT can be an interesting, future research direction. This will make a Transformer-based model more suitable for online MOT applications.

# REFERENCES

[1] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *IJCAI*, 1981, pp. 674–679.

[2] M. Naphade, Z. Tang, M.-C. Chang, D. C. Anastasiu, A. Sharma, R. Chellappa, S. Wang, P. Chakraborty, T. Huang, J.-N. Hwang, and S. Lyu, “The 2019 ai city challenge,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[3] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, “Learning features by watching objects move,” in *CVPR*, 2017.

[4] X. Wang and A. Gupta, “Unsupervised learning of visual representations using videos,” in *ICCV*, 2015.

[5] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, “3d traffic scene understanding from movable platforms,” *PAMI*, 2014.

[6] W. Choi and S. Savarese, “A unified framework for multi-target tracking and collective activity recognition,” in *ECCV*, 2012.

[7] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *CVPR*, 2012.

[8] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, “Globally-optimal greedy algorithms for tracking a variable number of objects,” in *CVPR*, 2011.

[9] J. Berclaz, E. Turetken, F. Fleuret, and P. Fua, “Multiple object tracking using K-shortest paths optimization,” *PAMI*, 2011.

[10] L. Zhang and R. Nevatia, “Global data association for multi-object tracking using network flows,” in *CVPR*, 2008.

[11] A. Butt and R. Collins, “Multi-target tracking by Lagrangian relaxation to min-cost network flow,” in *CVPR*, 2013.

[12] D. Reid, “An algorithm for tracking multiple targets,” *IEEE Transactions on Automatic Control*, 1979.

[13] I. J. Cox and S. L. Hingorani, “An efficient implementation of Reid’s multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking,” *PAMI*, 1996.

[14] M. Han, W. Xu, H. Tao, and Y. Gong, “An algorithm for multiple object trajectory tracking,” in *CVPR*, 2004.

[15] D. J. Papageorgiou and M. R. Salpukas, “The maximum weight independent set problem for data association in multiple hypothesis tracking,” *Optimization and Cooperative Control Strategies*, 2009.

[16] Z. Khan, T. Balch, and F. Dellaert, “MCMC-based particle filtering for tracking a variable number of interacting targets,” *PAMI*, 2005.

[17] S. Oh, S. Russell, and S. Sastry, “Markov Chain Monte Carlo data association for multi-target tracking,” *IEEE Transactions on Automatic Control*, 2009.

[18] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in *CVPR*, 2016.

[19] D. Held, S. Thrun, and S. Savarese, “Learning to track at 100 fps with deep regression networks,” in *European Conference Computer Vision (ECCV)*, 2016.

[20] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2015.

[21] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, “End-to-end representation learning for correlation filter based tracking,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] W. B. Thompson and T.-C. Pong, “Detecting moving objects,” *International Journal of Computer Vision*, vol. 4, no. 1, pp. 39–57, 1990.

[23] E. L.-M. Pia Bideau, “It’s moving! a probabilistic model for causal motion segmentation in moving camera videos,” in *European Conference on Computer Vision (ECCV)*, 2016.

[24] P. Bideau, A. RoyChowdhury, R. R. Menon, and E. Learned-Miller, “The best of both worlds: Combining cnns and geometric constraints for hierarchichal motion segmentation,” in *CVPR*, 2018.

[25] S. S. A. Sadeghian A. Alahi, “Tracking the untrackable: Learning to track multiple cues with long-term dependencies,” in *ICCV*, 2017.

[26] K. Fang, Y. Xiang, X. Li, and S. Savarese, “Recurrent autoregressive networks for online multi-object tracking,” in *WACV*, 2018.

[27] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, “Online multi-object tracking with dual matching attention networks,” in *ECCV*, 2018.

[28] J. Liu, P. Carr, R. T. Collins, and Y. Liu, “Tracking sports players with context-conditioned motion models,” in *CVPR*, 2013.

[29] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua, “Multi-commodity network flow for tracking multiple people,” *PAMI*, 2014.

[30] C. Huang, Y. Li, and R. Nevatia, “Multiple target tracking by learning-based hierarchical association of detection responses,” *PAMI*, 2013.

[31] W. Brendel, M. Amer, and S. Todorovic, “Multiobject tracking as maximum weight independent set,” in *CVPR*, 2011.

[32] R. T. Collins, “Multitarget data association with higher-order motion models,” in *CVPR*, 2012.

[33] A. Andriyenko, K. Schindler, and S. Roth, “Discrete-continuous optimization for multi-target tracking,” in *CVPR*, 2012.

[34] A. Segal and I. Reid, “Latent data association: Bayesian model selection for multi-target tracking,” in *ICCV*, 2013.

[35] H. B. Shitrit, J. Berclaz, F. Fleuret, and P. Fua, “Tracking multiple people under global appearance constraints.,” in *ICCV*, 2011.

[36] A. Smeulder, D. Chu, R. Cucchiara, S. Calderara, A. Deghan, and M. Shah, “Visual tracking: An experimental survey,” *PAMI*, 2014.

[37] S.-H. Bae and K.-J. Yoon, “Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning,” in *CVPR*, 2014.

[38] B. Yang and R. Nevatia, “An online learned CRF model for multi-target tracking,” in *CVPR*, 2012.

[39] C.-H. Kuo, C. Huang, and R. Nevatia, “Multi-target tracking by on-line learned discriminative appearance models.,” in *CVPR*, 2010.

[40] C.-H. Kuo and R. Nevatia, “How does person identity recognition help multi-person tracking?” In *CVPR*, 2011.

[41] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool, “Online multi-person tracking-by-detection from a single, uncalibrated camera,” *PAMI*, 2011.

[42] X. Song, J. Cui, H. Zha, and H. Zhao, “Vision-based multiple interacting targets tracking via on-line supervised learning,” in *ECCV*, 2008.

[43] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler, “Learning by tracking: Siamese cnn for robust target association,” in *CVPR Workshops*, 2016.

[44] J. Son, M. Baek, M. Cho, and B. Han, “Multi-object tracking with quadruplet convolutional neural networks,” in *CVPR*, 2017.

[45] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking with lifted multicut and person re-identification," in *CVPR*, 2017.

[46] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *AAAI*, 2017.

[47] T. Yang and A. B. Chan, "Recurrent filter learning for visual tracking," *arXiv:1708.03874*, 2017.

[48] D. Gordon, A. Farhadi, and D. Fox, "Re3: Re al-time recurrent regression networks for visual tracking of generic objects," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 788–795, 2018.

[49] A. Maksai and P. Fua, "Eliminating exposure bias and metric mismatch in multiple object tracking," in *CVPR*, 2019.

[50] J. Xu, Y. Cao, Z. Zhang, and H. Hu, "Spatial-temporal relation networks for multi-object tracking," in *ICCV 2019*, 2019.

[51] N. L.M.K.W.Z.M.-H. Y. Ji Zhu Hua Yang, "Online multi-object tracking with dual matching attention networks," in *ECCV*, 2018.

[52] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *CVPR*, 2020.

[53] X. Weng, Y. Wang, Y. Man, and K. Kitani, "GNN3DMOT: Graph Neural Network for 3D Multi-Object Tracking with 2D-3D Multi-Feature Learning," *CVPR*, 2020.

[54] L. Ma, S. Tang, M. J. Black, and L. V. Gool, "Customized multi-person tracker," in *ACCV 2018*, 2018.

[55] S. Pellegrini, A. Ess, K. Schindler, and L. J. V. Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *ICCV*, 2009.

[56] C. Kim, F. Li, A. Ciptadi, and J. Rehg, "Multiple hypothesis tracking revisited," in *ICCV*, 2015.

[57] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House, 1999, ISBN: 9781580530064.

[58] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[59] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[60] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg, "Video segmentation by tracking many figure-ground segments," in *ICCV*, 2013.

[61] P. R. Ostergard, "A new algorithm for the maximum-weight clique problem," *Nordic Journal of Computing*, 2001.

[62] S. Busygin, "A new trust region technique for the maximum weight clique problem," *Discrete Appl. Math.*, 2006.

[63] A. R. Zamir, A. Dehghan, and M. Shah, "GMCP-tracker: Global multi-object tracking using generalized minimum clique graphs," in *ECCV*, 2012.

[64] A. Milan, K. Schindler, and S. Roth, "Detection-and trajectory-level exclusion in multiple object tracking," in *CVPR*, 2013.

[65] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a benchmark for multi-target tracking," *arXiv:1504.01942 [cs]*, 2015.

[66] J. Ferryman and A. Ellis, "PETS2010: Dataset and challenge," in *AVSS*, 2010.

[67] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *Image and Video Processing*, 2008.

[68] A. Milan, K. Schindler, and S. Roth, "Challenges of ground truth evaluation of multi-target tracking," in *CVPR Workshop*, 2013.

[69] S. Wang and F. C., "Learning optimal parameters for multi-target tracking," in *BMVC*, 2015.

[70] N. McLaughlin, J. Martinez Del Rincon, and P. Miller, "Enhancing linear programming with motion modeling for multi-target tracking," in *WACV*, 2015.

[71] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese, "Learning an image-based motion context for multiple people tracking," in *CVPR*, 2014.

[72] A. Milan, L. Leal-Taixé, I. Reid, and K. Schindler, "Joint tracking and segmentation of multiple targets," in *CVPR*, 2015.

[73] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multitarget tracking," *PAMI*, 2014.

[74] J. Yoon, H. Yang, J. Lim, and K. Yoon, "Bayesian multi-object tracking using motion context from multiple objects," in *WACV*, 2015.

[75] C. Dicle, O. Camps, and M. Sznaier, "The way they move: Tracking targets with similar appearance," in *ICCV*, 2013.

[76] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, “3D traffic scene understanding from movable platforms,” *PAMI*, 2014.

[77] C. Kim, F. Li, and J. Rehg, “Multi-object tracking with neural gating using bilinear lstm,” in *ECCV*, 2018.

[78] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *CVPR*, 2014.

[79] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *NIPS*, 2015.

[80] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, “MOT16: A benchmark for multi-object tracking,” *arXiv:1603.00831*, 2016.

[81] J. Hong Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon, “Online multi-object tracking via structural constraint event aggregation,” in *CVPR*, 2016, pp. 1392–1400.

[82] W. Choi, “Near-online multi-target tracking with aggregated local flow descriptor,” in *ICCV*, 2015.

[83] S. Tang, B. Andres, M. Andriluka, and B. Schiele, “Multi-person tracking by multicut and deep matching,” in *ECCV Workshops*, 2016.

[84] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997.

[85] S. Blackman, “Multiple hypothesis tracking for multiple target tracking,” *Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[86] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[87] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.

[88] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part based models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[89] F. Yang, W. Choi, and Y. Lin, “Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers,” in *CVPR*, 2016.

[90] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, “MOTChallenge 2015: Towards a benchmark for multi-target tracking,” *arXiv:1504.01942*, 2015.

[91] A. Ess, B. Leibe, K. Schindler, and L. van Gool, “A mobile vision system for robust multi-person tracking,” in *CVPR*, 2008.

[92] M. Andriluka, S. Roth, and B. Schiele, “People-tracking-by-detection and people-detection-by-tracking,” in *CVPR*, 2008.

[93] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, “Scalable person re-identification: A benchmark,” in *ICCV*, 2015.

[94] W. Li, R. Zhao, T. Xiao, and X. Wang, “Deepreid: Deep filter pairing neural network for person re-identification,” in *CVPR*, 2014.

[95] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance measures and a data set for multi-target, multi-camera tracking,” in *ECCV Workshop*, 2016.

[96] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.

[97] M. Keuper, S. Tang, Z. Yu, B. Andres, T. Brox, and B. Schiele, “A multi-cut formulation for joint segmentation and tracking of multiple objects,” *arXiv:1607.06317*, 2016.

[98] C. Long, A. Haizhou, Z. Zijie, and S. Chong, “Real-time multiple people tracking with deeply learned candidate selection and person re-identification,” in *ICME*, 2018.

[99] Z. Fu, P. Feng, F. Angelini, J. A. Chambers, and S. M. Naqvi, “Particle phd filter based multiple human tracking using online group-structured dictionary learning,” *IEEE Access*, vol. 6, pp. 14 764–14 778, 2018.

[100] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn, “Fusion of head and full-body detectors for multi-object tracking,” in *CVPR Workshops*, 2018.

[101] J. Chen, H. Sheng, Y. Zhang, and Z. Xiong, “Enhancing detection model for multiple hypothesis tracking,” in *CVPR Workshops*, 2017.

[102] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, “Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism,” in *ICCV*, 2017.

[103] E. Levinkov, J. Uhrig, S. Tang, M. Omran, E. Insafutdinov, A. Kirillov, C. Rother, T. Brox, B. Schiele, and B. Andres, “Joint graph decomposition & node labeling: Problem, algorithms, applications,” in *CVPR*, 2017.

[104] L. Fagot-Bouquet, R. Audigier, Y. Dhome, and F. Lerasle, “Improving multi-frame data association with sparse representations for robust near-online multi-object tracking,” in *ECCV*, 2016.

[105] S.-H. Bae and K.-J. Yoon, “Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 595–610, 2018.

[106] I. Sutskever, "Training recurrent neural networks," *PhD thesis*, 2013.

[107] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 5998–6008.

[108] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection (best student paper award)," in *ICCV*, 2017.

[109] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[110] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "CVPR19 tracking and detection challenge: How crowded can it get?" *arXiv:1906.04567 [cs]*, Jun. 2019, arXiv: 1906.04567.

[111] Y. Xu, A. Osep, Y. Ban, R. Horaud, L. Leal-Taixé, and X. Alameda-Pineda, "How to train your deep multi-object tracker," in *CVPR*, 2020, pp. 6787–6796.

[112] Q. Liu, Q. Chu, B. Liu, and N. Yu, "Gsm: Graph similarity model for multi-object tracking," in *IJCAI*, C. Bessiere, Ed., Main track, International Joint Conferences on Artificial Intelligence Organization, Jul. 2020, pp. 530–536.

[113] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, "Tracking without bells and whistles," in *ICCV 2019*, 2019.

[114] S. B. S. Lee M. Kim, "Learning discriminative appearance models for online multi-object tracking with appearance discriminability measures," *IEEE Access*, 2018.

[115] Y. chul Yoon, A. Boragule, Y. min Song, K. Yoon, and M. Jeon, "Online multi-object tracking with historical appearance matching and scene adaptive detection filtering," in *AVSS*, 2018.

[116] P. Chu and H. Ling, "Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking," in *ICCV 2019*, 2019.

[117] P. Chu, H. Fan, C. Tan, and H. Ling, "Online multi-object tracking with instance-aware tracker and dynamic model refreshment," in *WACV*, 2019.

[118] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Subgraph decomposition for multi-target tracking," in *CVPR*, 2015.

[119] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele, "Motion segmentation and multiple object tracking by correlation co-clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[120] X. Zhou, V. Koltun, and P. Krähenbühl, “Tracking objects as points,” *ECCV*, 2020.