Learning by tracking: Siamese CNN for robust target association

Laura Leal-Taixé
TU München
Munich, Germany

Cristian Canton-Ferrer
Microsoft
Redmond (WA), USA

Konrad Schindler
ETH Zurich
Zurich, Switzerland

Abstract

This paper introduces a novel approach to the task of data association within the context of pedestrian tracking, by introducing a two-stage learning scheme to match pairs of detections. First, a Siamese convolutional neural network (CNN) is trained to learn descriptors encoding local spatio-temporal structures between the two input image patches, aggregating pixel values and optical flow information. Second, a set of contextual features derived from the position and size of the compared input patches are combined with the CNN output by means of a gradient boosting classifier to generate the final matching probability. This learning approach is validated by using a linear programming based multi-person tracker showing that even a simple and efficient tracker may outperform much more complex models when fed with our learned matching probabilities. Results on publicly available sequences show that our method meets state-of-the-art standards in multiple people tracking.

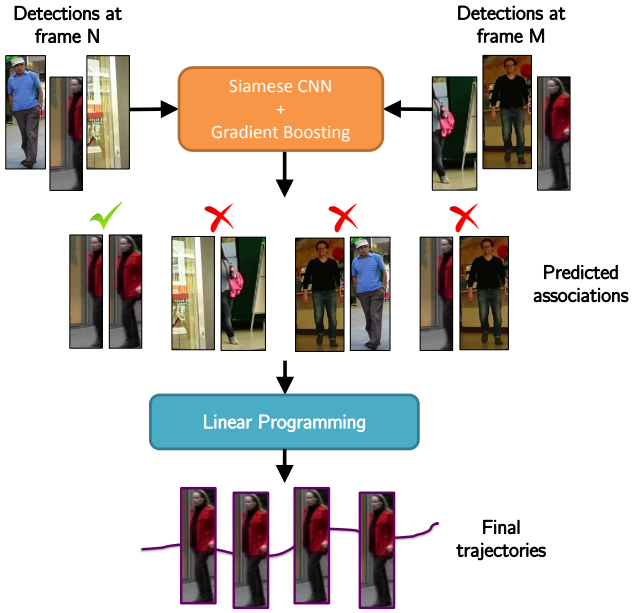


Figure 1: Multiple object tracking with learned detection associations.

1. Introduction

One of the big challenges of computer vision is scene understanding from video. Humans are often the center of attention of a scene, which leads to the fundamental problem of detecting and tracking them in a video. To track multiple people, *tracking-by-detection* has emerged as the preferred method. That approach simplifies the problem by dividing it into two steps. First, find probable pedestrian locations independently in each frame. Second, link corresponding detections across time to form trajectories.

The linking step, called *data association* is a difficult task on its own, due to missing and spurious detections, occlusions, and targets interactions in crowded environments. To address these issues, research in this area has produced more and more complex models: global optimization methods based on network flow [4, 64], minimum cliques [61] or discrete-continuous CRF inference [1]; models of pedestrian interaction with social motion models [35, 44]; integration of additional motion cues such as dense point trajectories [9, 23]; and person re-identification techniques to improve appearance models [30, 32]. Even though the models became progressively more sophisticated, the underlying descriptors, which are used to decide whether two detections belong to the same trajectory, remained quite simple and struggle in challenging scenarios (*e.g.*, crowds, frequent occlusions, strong illumination effects).

Recently, larger amounts of annotated data have become available and, with the help of these data, convolutional neural networks (CNNs) that learn feature representations as part of their training have outperformed heuristic, hand-engineered features in several vision problems [31]. Here, we adapt the CNN philosophy to multi-person tracking. In order to circumvent manual feature design for data association, we propose to learn the decision whether two detections belong to the same trajectory. Our learning framework has two stages: first, a CNN in Siamese twin architecture is trained to assess the similarity of two equally sized

image regions; second, contextual features that capture the relative geometry and position of the two patches of interest are combined with the CNN output to produce a final prediction, in our case using gradient boosting (GB). Given the learned, pairwise data association score we construct a graph that links all available detections across frames, and solve the standard Linear Programming (LP) formulation of multi-target tracking. We show that this simple and efficient linear tracker – in some sense the "canonical baseline" of modern multi-target tracking – outperforms much more complex models when fed with our learned edge costs.

1.1. Contributions

This paper presents three major contributions to the pedestrian tracking task:

- Within the context of tracking, we introduce a novel learning perspective to the data association problem.
- We propose to use a CNN in a Siamese configuration to estimate the likelihood that two pedestrian detections belong to the same tracked entity. In the presented CNN architecture, pixel values and optical flow are combined as a multi-modal input.
- We show that formulating data association with a linear optimization model outperform complex models when fed with accurate edge costs.

1.2. Related work

Multi-person tracking. Multi-person tracking is the input for a number of computer vision applications, such as surveillance, activity recognition or autonomous driving. Despite the vast literature on the topic [39], it still remains a challenging problem, especially in crowded environments where occlusions and false detections are common. Most modern methods use the tracking-by-detection paradigm, which divides the task into two steps: detecting pedestrians in the scene [17, 20, 25], and linking those detections over time to create trajectories. A common formalism is to represent the problem as a graph, where each detection is a node, and edges indicate a possible link. The data association can then be formulated as maximum flow [4] or, equivalently, minimum cost problem [28, 35, 45, 64], both efficiently solved to (near-)global optimality with LP, with a superior performance compared to frame-by-frame [29] or track-by-track [3] methods. Alternative formulations typically lead to more involved optimization problems, including minimum cliques [61] or general-purpose solvers like MCMC [59]. There are also models that represent trajectories in continuous space and use gradient-based optimization, sometimes alternating with discrete inference for data association [1].

A recent trend is to design ever more complex models, which include further vision routines in the hope that they benefit the tracker, including reconstruction for multi-camera sequences [36, 54], activity recognition [11] and segmentation [40]. In general, the added complexity seems to exhibit diminishing returns, at significantly higher computational cost.

Other works have focused on designing more robust features to discriminate pedestrians. Color-based appearance models are common [30], but not always reliable, since people can wear very similar clothes, and color statistics are often contaminated by the background pixels and illumination changes. Kuo *et al.* [32], borrow ideas from person re-identification and adapt them to "re-identify" targets during tracking. In [57], a CRF model is learned to better distinguish pedestrians with similar appearance. A different line of attack is to develop sophisticated motion models in order to better predict a tracked person's location, most notably models that include interactions between nearby people [1, 10, 35, 44, 47, 56]. A problem of such models is that they hand-craft a term for each external influence (like collision avoidance, or walking in groups). This limits their applicability, because it is difficult to anticipate all possible interaction scenarios. The problem can be to some degree alleviated by learning the motion model from data [33], although this, too, only works if all relevant motion and interaction patterns are present in the training data. Moreover, the motion model does not seem to be an important bottleneck in present tracking frameworks. By and large, more powerful dynamic models seem to help only in a comparatively small number of situations, while again adding complexity.

Measuring similarity with CNNs. Convolutional architectures have become the method of choice for end-to-end learning of image representations. In relation to our problem, they have also been remarkably successful in assessing the similarity of image patches for different tasks such as optical flow estimation [21], face verification [49], and depth estimation from multiple viewpoints [22, 60, 62].

In the context of tracking, CNNs have been used to model appearance and scale variations of the target [18]. Recently, several authors employ them to track via online learning, by continuously fine-tuning a pre-trained CNN model [8, 37, 51].

2. Learning to associate detections

Our tracking framework is based on the paradigm of tracking-by-detection, i.e. firstly, we run a detector through the sequences, and secondly, we link the detections to form trajectories. We propose to address the data association problem by learning a model to predict whether two detections belong to the same trajectory or not. We use two sets of features derived from the pedestrian detections to be compared. First, *local spatio-temporal features* learnt using

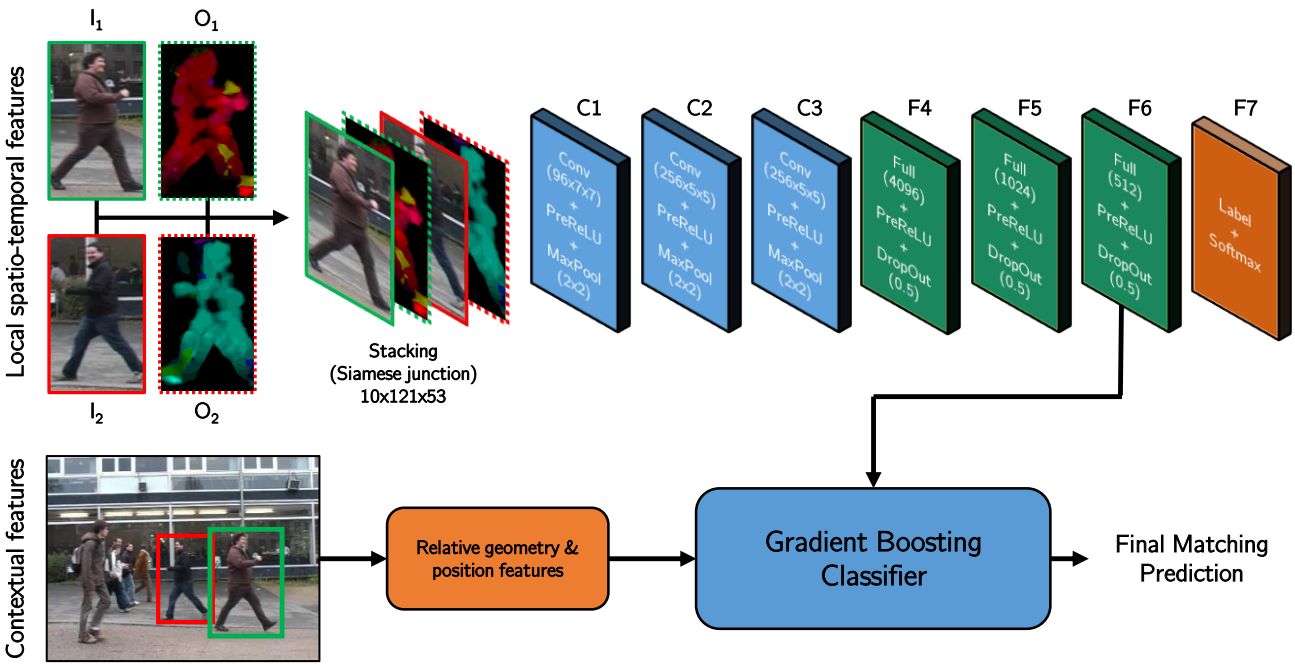


Figure 2: Proposed two-stage learning architecture for pedestrian detection matching.

a CNN and, second, *contextual features* encoding the relative geometry and position variations of the two detections. Finally, both sets of features are combined using a GB classifier [24] to produce the final prediction (see Fig.2). Decoupling local and global features processing and ensembling them in a later stage allows understanding the contribution of each factor plus adding robustness to the prediction [15, 50].

2.1. CNN for patch similarity

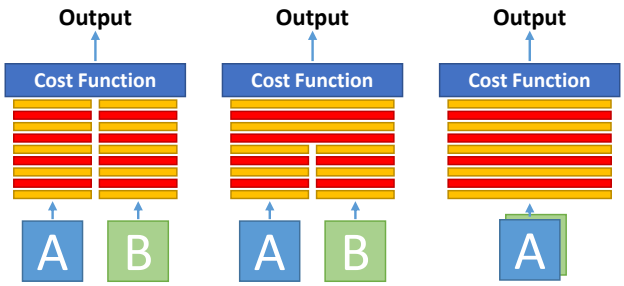
A common denominator when comparing two image patches using CNNs are Siamese architectures where two inputs are processed simultaneously by several layers with shared weights (convolutional and/or fully connected) that eventually merge at some point in the network. Siamese CNN topologies can be grouped under three main categories, depending on the point where the information from each input patch is combined (see Fig.3):

- **Cost function.** Input patches are processed by two parallel branches featuring the same network structure and weights. Finally, the top layers of each branch are fed to a cost function [12, 49] that aims at learning a manifold where different classes are easily separable.
- **In-network.** In this case, the top layers of the parallel branches processing the two different inputs are concatenated and some more layers are added on top of that [21, 62]. Finally, the standard softmax log-loss function is employed.
- **Joint data input.** The two input patches are stacked together forming a unified input to the CNN [21]. Again, the softmax log-loss function is used here.

While the two first approaches have yield good results in classification applications, the best performance for tasks involving comparison of detailed structures is obtained with the joint data input strategy. As pointed out by [60] and further corroborated by [21], jointly using information from both patches from the first layer tends to deliver a better performance. In order to verify this hypothesis within the scope of the tracking problem, we trained a Siamese network using the contrastive loss function [13]:

$$E = \frac{1}{2N} \sum_{n=1}^{N} (y)\, d + (1-y) \max(\tau - d, 0)\,,$$

where $d = ||a_n - b_n||_2^2$, being a_n and b_n the $L2$ normalized responses of the top fully connected layer of the parallel branches processing each input image, and $\tau = 0.2$ is the separation margin and y the label value encoded as 0 or 1. The topology of the CNN network has been the same all through the paper and shown in Fig.2. Our early experiments, showed a relative 8% AUC increase of the joint data input case over the best performing model from the other two topologies, given a fixed number of parameters.



(a) Cost function (b) In-network (c) Input stacking

Figure 3: Siamese CNN topologies

Architecture. The proposed CNN architecture takes as input four sources of information: the pixel values in the normalized LUV color format for each patch to be compared, $\mathbf{I}_1$ and $\mathbf{I}_2$, and the corresponding x and y components of their associated optical flow [19], $\mathbf{O}_1$ and $\mathbf{O}_2$. These four images are resized to a fixed size of 121x53 and stacked depth-wise to form a multi-modal 10-channel data blob $\mathbf{D}$ to be fed to the CNN. In order to improve robustness against varying light conditions, for each luma channel L of both $\mathbf{I}_1$ and $\mathbf{I}_2$ we perform a histogram equalization and a plane fitting, as introduced in [63].

The input data is processed first by three convolutional layers, $\mathbf{C}_{1,2,3}$, each of them followed by a PreReLU nonlinearity [27] and a max-pooling layer that renders the net more robust to miss alignments within the components of $\mathbf{D}$. Afterwards, four fully connected layers, $\mathbf{F}_{4,5,6,7}$, aim at capturing correlations between features in distant parts of the image as well as cross-modal dependencies, i.e. pixel-to-motion interactions between $\mathbf{I}_{1,2}$ and $\mathbf{O}_{1,2}$. The output of the last fully-connected layer is fed to a binary softmax which produces a distribution over the class labels (match/no match). The output of layer $\mathbf{F}_6$ in the network will be used as our raw patch matching representation feature vector to be fed to the second learning stage.

Training data generation. Pedestrian detections proposed using [17] are generated for each frame and associations between detections are provided across frames during the training phase. On one hand, positive examples, i.e. pairs of detections corresponding to target m, $(\mathbf{I}_t^m, \mathbf{I}_{t-k}^m)$, $1 \leq k < N$, are directly generated from the ground truth data, with a maximum rewind time of $N = 15$. On the other hand, negative examples are generated by either pairing two true detections with belonging to different people, a true detection with a false positive or two false positive detections; in order to increase the variety of data presented to the CNN, we enlarged the set of false positives by randomly selecting patches from the image of a given aspect ratio that do not overlap with true positive detections. By generating these random false positives, the CNN does not overfit to the specific type of false positives generated by the employed pedestrian detector thus increasing its capacity of generalization.

Learning. We trained the proposed CNN as a binary classification task, employing the standard back-propagation on feed-forward nets by stochastic gradient descent with momentum. The mini-batch size was set to 128, with an equal learning rate for all layers set to 0.01, sequentially decreased every 1.5 epochs by a factor 10, finally reaching 10^{-4}. Layer weight were initialized following [27] and we trained our CNN on a Titan GPU X for 50 epochs. The Lasagne/Theano framework was employed to run our experiments.

Data augmentation. Even if the available training data is fairly large, pairs of pedestrian detections tend not to have a large range of appearances stemming from the fact that the number of distinct people in the training corpus is limited. Adding variety to the input data during the training phase is a widely employed strategy to reduce overfitting and improve generalization of CNNs [15, 16, 31]. In our particular case, we have randomly added geometric distortions (rotation, translation, skewing, scaling and vertical flipping) as well as image distortions (Gaussian blur, noise and gamma). These transformations are applied independently for each of the two input patches but only allowing small relative geometric transformations between them (with the exception of vertical flipping that is applied to both images, when chosen). Since all these transformation are performed directly on GPU memory, the augmentation complexity cost is negligible.

2.2. Evidence aggregation with gradient boosting

The softmax output of the presented Siamese CNN might be used directly for pedestrian detection association but the accuracy would be low since we are not taking into account *where* and *when* these detections originated in the image. Therefore, the need for a set of contextual features and a higher order classifier to aggregate all this information.

Given two pedestrian detections at different time instants, $\mathbf{I}_{t_1}$ and $\mathbf{I}_{t_2}$, encoded by its position $\mathbf{x} = (x, y)$ and dimensions $\mathbf{s} = (w, h)$, we define our contextual features as: the relative size change, $(\mathbf{s}_1 - \mathbf{s}_2)/(\mathbf{s}_1 + \mathbf{s}_2)$, the position change, $(\mathbf{x}_1 - \mathbf{x}_2)$, and the relative velocity between them, $(\mathbf{x}_1 - \mathbf{x}_2)/(t_2 - t_1)$.

Combining the local and contextual sets of features is carried out using gradient boosting (GB) [24]. To avoid overfitting on the GB, CNN predictions for each of the train sequences are generated in a leave-one-out fashion following the stacked generalization concept introduced in [53]. Finally, the GB classifier is trained by concatenating the CNN and contextual features. In our case, we trained the GB classifier using 400 trees using the distributed implementation presented in [7].

3. Tracking with Linear Programming

In this section, we present the tracking framework where we incorporate the score defined in the previous section in order to solve the data association problem.

Let $\mathcal{D} = \{\mathbf{d}_i^t\}$ be a set of object detections with $\mathbf{d}_i^t = (x, y, t)$, where (x, y) is the 2D image position and t defines the time stamp. A trajectory is defined as a list of ordered object detections $T_k = \{\mathbf{d}_{k_1}^{t_1}, \mathbf{d}_{k_2}^{t_2}, \cdots, \mathbf{d}_{k_N}^{t_N}\}$, and the goal of multiple object tracking is to find the set of trajectories $\mathcal{T}* = \{T_k\}$ that best explains the detections $\mathcal{D}$. This can be expressed as a Maximum A-Posteriori (MAP) problem and directly mapped to a Linear Programming formulation, as detailed in [35, 64].

The data association problem is therefore defined by a linear program with objective function:

$$\mathcal{T}* = \underset{\mathcal{T}}{\mathbf{argmin}} \sum_i C_{\text{in}}(i) f_{\text{in}}(i) + \sum_i C_{\text{out}}(i) f_{\text{out}}(i) + \sum_i C_{\text{det}}(i) f(i) + \sum_{i,j} C_{\text{t}}(i,j) f(i,j) \quad (1)$$

subject to edge capacity constraints, flow conservation at the nodes and exclusion constraints.

The costs C_{in} and C_{out} define how probable it is for a trajectory to start or end. The detection cost $C_{\text{det}}(i)$ is linked to the score that detection i was given by the detector. Intuitively, if the score s_i is very high, the cost of the edge should be very negative, so that flow will likely pass through this edge, including the detection i in a trajectory. We normalize the costs $s_i = [0, 1]$ for a sequence, and define the detection cost as:

$$C_{\text{det}}(i) = \begin{cases} \frac{-s_i}{V_{\text{det}}} + 1 & \text{if } s_i < V_{\text{det}} \\ \frac{-s_i+1}{1-V_{\text{link}}} - 1 & \text{if } s_i \geq V_{\text{det}} \end{cases} \quad (2)$$

If we set, for example, $V_{\text{det}} = 0.5$, the top half confident detections will correspond to edges with negative cost, and will most likely be used in some trajectory. By varying this threshold, we can adapt to different types of detectors that have different rates of false positives.

The cost of a link edge depends only on the probability that the two detections i and j belong to the same trajectory, as estimated by our classifier:

$$C_{\text{t}}(i,j) = \begin{cases} \frac{-s_{i,j}^{\text{RF}}}{V_{\text{link}}} + 1 & \text{if } s_{i,j}^{\text{RF}} < V_{\text{link}} \\ \frac{-s_{i,j}^{\text{RF}}+1}{1-V_{\text{link}}} - 1 & \text{if } s_{i,j}^{\text{RF}} \geq V_{\text{link}} \end{cases} \quad (3)$$

Note in Eq. (1), that if all costs are positive, the trivial solution will be zero flow. A trajectory is only created if its total cost is negative. We define detection costs to be negative if we are confident that the detection is a pedestrian, while transition costs are negative if our classifier is very confident that two detections belong to the same trajectory. We control with V_{det} and V_{link} the percentage of negative edges that we want in the graph. The in/out costs, on the other hand, are positive and they are used so that the tracker does not indiscriminately create many trajectories. Therefore, a trajectory will only be created if there is a set of confident detections and confident links whose negative costs outweigh the in/out costs. $C_{\text{in}} = C_{\text{out}}$, V_{det} and V_{link} are learned from training data as discussed in the next section.

The Linear Program in Eq. (1) can be efficiently solved using Simplex [35] or k-shortest paths [4]. Note, that we could use any other optimization framework, such as maximum cliques [61], or Kalman filter [44] for real-time applications.

4. Experimental results

This section presents the results validating the efficiency of the proposed learning approach to match pairs of pedestrian detections as well as its performance when creating trajectories by means of the aforementioned linear programming tracker. In order to provide comparable results with the rest of the state-of-the-art methods, we employed the large MOTChallenge [34] dataset, a common reference when addressing multi-object tracking problems. It consists of 11 sequences for training, almost 40,000 bounding boxes, and 11 sequences for testing, over 60,000 boxes, comprising sequences with moving and static cameras, dense scenes, different viewpoints, etc.

4.1. Detection matching

We first evaluate the performance of the proposed learning approach when predicting the probability of two detections belonging to the same trajectory by means of the ROC curve computed on the training data of MOT15 [34], as shown in Fig.4. Two result groups are depicted: first, when only using the CNN classifier (best AUC: 0.718) and, second, when using the two stage CNN+GB classifier (best AUC: 0.954); the later yielding to a relative 41% increase in classification performance. Oversampling the image (1,2,4 and 8 fixed locations) and averaging their predictions proved to deliver a significant improvement, specially for the CNN part of the end-to-end system. However, the impact of oversampling in the CNN+GB classifier is less relevant hence it may be avoided to reduce the overall computation load.

An analysis of the ROC curve on the MOT15 training data allowed us to find the operation point, i.e. probability threshold V_{link} within the linear programming tracking, that would maximize its accuracy. In our case, we set $V_{\text{link}} = 0.35$, after cross-validation.

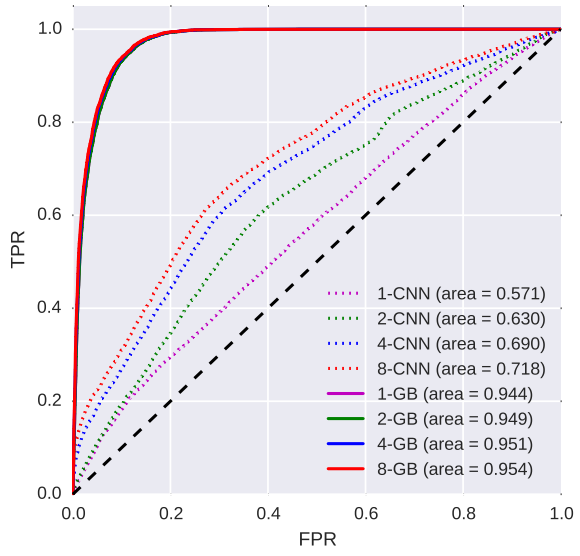


Figure 4: Performance accuracy for the Siamese CNN and the full two-stage learning approach (CNN+GB), when using an oversampling of 8,4,2 and 1 per pair at the input.

4.2. Multiple people tracking

Evaluation metrics. To evaluate multiple-object tracking performance, we used CLEAR MOT metrics [5], tracking accuracy (TA) and precision (TP). TA incorporates the three major error types (missing recall, false alarms and identity switches (IDsw)) while TP is a measure for the localization error, where 100% again reflects a perfect alignment of the output tracks and the ground truth. There are also two measures taken from [38] which reflect the temporal coverage of true trajectories by the tracker: mostly tracked (MT, $> 80\%$ overlap) and mostly lost (ML, $< 20\%$). We use only publicly available detections and evaluation scripts provided in the benchmark [34].

Determining optimal parameters. As discussed before, the LP parameter $V_{\text{link}} = 0.35$ is given by the operation point of the ROC curve. The other LP parameters, $C_{\text{in}} = C_{\text{out}}$, V_{det} are determined by parameter sweep with cross-validation on the training MOT15 data in order to obtain the maximum tracking accuracy.

Baselines. We compare to two tracking methods based on Linear Programming. The first is using only 2D distance information as feature (LP2D), the second [33] is learning to predict the motion of a pedestrian using image features (MotiCon). This comparison is specially interesting, since the optimization structure for all methods is based on Linear Programming, and the only factor that changes is the way the edge costs are computed. In this way, we can see the real contribution of our proposed learn-based costs. As it can be seen in Table 1, the results indicate that our learned data association costs are more accurate, and that this better low-level evidence is the key factor driving the performance improvement.

Dataset	Method	TA	TP	MT	ML	IDsw
TUD-Crossing	MotiCon	58.2	70.8	23.1	15.4	403
	LP2D	49.5	74.1	15.4	15.4	48
	Proposed	73.7	73.0	69.2	15.4	197
PETS09-S2L2	MotiCon	46.6	67.6	9.5	14.3	238
	LP2D	40.7	70.2	9.5	16.7	319
	Proposed	34.5	69.7	7.1	19.0	282
ETH-Jelmoli	MotiCon	43.5	72.9	20.0	28.9	37
	LP2D	40.7	73.5	15.6	26.7	41
	Proposed	42.3	72.8	24.4	31.1	30
ETH-Linthescher	MotiCon	18.3	77.7	1.5	74.1	72
	LP2D	16.9	76.4	2.0	73.6	77
	Proposed	16.7	74.2	4.6	78.7	9
ETH-Crossing	MotiCon	22.8	72.9	3.8	65.4	8
	LP2D	21.4	76.3	3.8	65.4	10
	Proposed	27.5	74.1	3.8	65.4	4
AVG-TownCentre	MotiCon	11.9	70.3	0.9	69.9	74
	LP2D	15.5	68.5	8.4	33.2	260
	Proposed	19.3	69.0	4.4	44.7	142
ADL-Rundle-1	MotiCon	1.0	70.3	18.8	12.5	136
	LP2D	2.9	72.2	15.6	21.9	252
	Proposed	26.5	71.6	28.1	28.1	33
ADL-Rundle-3	MotiCon	18.1	71.8	4.5	20.5	217
	LP2D	13.7	72.8	2.3	25.0	400
	Proposed	39.7	72.9	11.4	34.1	33
KITTI-16	MotiCon	38.8	70.1	0.0	11.8	36
	LP2D	35.5	72.0	0.0	11.8	47
	Proposed	36.9	72.6	0.0	17.6	24
KITTI-19	MotiCon	33.8	69.9	6.5	21.0	100
	LP2D	20.1	65.2	8.1	21.0	97
	Proposed	26.7	66.2	6.5	29.0	70
Venice-1	MotiCon	18.2	72.9	0.0	29.4	74
	LP2D	11.0	72.4	0.0	35.3	98
	Proposed	22.3	73.0	0.0	41.2	4

Table 1: Detailed result on the 11 sequences of MOTChallenge test, compared to two other methods that use also Linear Programming.

Finally we show the results on the test set of MOTChallenge in Table 2, where we compare to numerous state-of-the-art trackers. Our method is among the top performing trackers, and contains less false positives than any other method. Note, that we do not use any type of post-processing. Again, it clearly outperforms methods based on Linear Programming (LP2D and MotiCon), thanks to the proposed edge costs.

Method	TA	TP	MT	ML	IDsw	FP
NOMT [9]	**33.7**	**71.9**	12.2	44.0	442	7762
MHT-DAM [30]	32.4	71.8	**16.0**	43.8	435	9064
MDP [55]	30.3	71.3	13.0	**38.4**	680	9717
SiameseCNN (proposed)	29.0	71.2	8.5	48.4	639	**5160**
LP-SSVM [52]	25.2	71.7	5.8	53.0	849	8369
ELP [43]	25.0	71.2	7.5	43.8	1396	7345
JPDA-m [46]	23.8	68.2	5.0	58.1	**365**	6373
MotiCon [33]	23.1	70.9	4.7	52.0	1018	10404
SegTrack [40]	22.5	71.7	5.8	63.9	697	7890
LP2D (baseline)	19.8	71.2	6.7	41.2	1649	11580
DCO-X [41]	19.6	71.4	5.1	54.9	521	10652
CEM [42]	19.3	70.7	8.5	46.5	813	14180
RMOT [58]	18.6	69.6	5.3	53.3	684	12473
SMOT [14]	18.2	71.2	2.8	54.8	1148	8780
ALExTRAC [6]	17.0	71.2	3.9	52.4	1859	9233
TBD [26]	15.9	70.9	6.4	47.9	1939	14943
TC-ODAL [2]	15.1	70.5	3.2	55.8	637	12970
DP-NMS [45]	14.5	70.8	6.0	40.8	4537	13171
LDCT [48]	4.7	71.7	11.4	32.5	12348	14066

Table 2: Results on the MOTChallenge test set.

5. Conclusions

In this paper we have presented a two-stage learning based approach to associate detections within the context of pedestrian tracking. In a first pass, we create a multi-dimensional input blob stacking image and optical flow information from to the two patches to be compared; these data representation allows the following Siamese convolutional neural network to learn the relevant spatio-temporal features that allow distinguishing whether these two pedestrian detections belong to the same tracked entity. These local features are merged with some contextual features by means of a gradient boosting classifier yielding to a unified prediction.

In order to highlight the efficiency of the proposed detection association technique, we use a modified linear programming based tracker [64] to link the proposed correspondences and form trajectories. The complete system is evaluated over the standard MOTChallenge dataset [34], featuring enough data to ensure a satisfactory training of the CNN and a thorough and fair evaluation. When comparing the proposed results with the state-of-the-art, we observe that a simple linear programming tracker fed with accurate information reaches comparable performance than other more complex approaches.

Future research within this field involve applying the proposed approach to more generic target tracking, leveraging already trained models and extending the second stage classifier to deal with more complex contextual features, e.g. social forces [35]. Evaluation of the proposed architecture over on datasets is currently under investigation.

References

[1] A. Andriyenko and K. Schindler. Discrete-continuous optimization for multi-target tracking. *CVPR*, 2011. 1, 2

[2] S. Bae and K. Yoon. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. *CVPR*, 2014. 7

[3] J. Berclaz, F. Fleuret, and P. Fua. Robust people tracking with global trajectory optimization. *CVPR*, 2006. 2

[4] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI*, 2011. 1, 2, 5

[5] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *Image and Video Processing*, 2008(1):1–10, May 2008. 6

[6] A. Bewley, L. Ott, F. Ramos, and B. Upcroft. Alextrac: Affinity learning by exploring temporal reinforcement within association chains. *ICRA*, 2016. 7

[7] T. Chen and T. He. xgboost: extreme gradient boosting. GitHub, 2015. 4

[8] Y. Chen, X. Yang, B. Zhong, S. Pan, D. Chen, and H. Zhang. CNNTracker: Online discriminative object tracking via deep convolutional neural network. *Applied Soft Computing*, 2015. 2

[9] W. Choi. Near-online multi-target tracking with aggregated local flow descriptor. *ICCV*, 2015. 1, 7

[10] W. Choi and S. Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. *ECCV*, 2010. 2

[11] W. Choi and S. Savarese. A unified framework for multi-target tracking and collective activity recognition. *ECCV*, 2012. 2

[12] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. *CVPR*, 2005. 3

[13] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. *CVPR*, 2005. 3

[14] C. Dicle, O. Camps, and M. Sznaier. The way they move: Tracking targets with similar appearance. *ICCV*, 2013. 7

[15] S. Dieleman, A. van den Oord, I. Korshunova, J. Burms, J. Degrave, L. Pigou, and P. Buteneers. Classifying plankton with deep neural networks. Blog entry, 2015. 3, 4

[16] S. Dieleman, K. Willett, and J. Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(10):1441–1459, 2015. 4

[17] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *PAMI*, 2014. 2, 4

[18] J. Fan, W. Xu, Y. Wu, and Y. Gong. Human tracking using convolutional neural networks. *IEEE Transactions on Neural Networks*, 21(10):1610–1623, 2010. 2

[19] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, volume 2749 of *Lecture Notes in Computer Science*, pages 363–370. 2003. 4

[20] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010. 2

[21] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. *ICCV*, 2015. 2, 3

[22] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. DeepStereo: Learning to predict new views from the world's imagery. *arXiv:1506.06825*, 2015. 2

[23] K. Fragkiadaki, W. Zhang, G. Zhng, and J. Shi. Two-granularity tracking: mediating trajectory and detections graphs for tracking under occlusions. *ECCV*, 2012. 1

[24] J. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002. 3, 4

[25] J. Gall, A. Yao, N. Razavi, L. van Gool, and V. Lempitsky. Hough forests for object detection, tracking and action recognition. *TPAMI*, 2011. 2

[26] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3d traffic scene understanding from movable platforms. *TPAMI*, 2014. 7

[27] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *ICCV*, 2015. 4

[28] H. Jiang, S. Fels, and J. Little. A linear programming approach for multiple object tracking. *CVPR*, 2007. 2

[29] Z. Khan, T. Balch, and F. Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *TPAMI*, 2005. 2

[30] C. Kim, F. Li, A. Ciptadi, and J. Rehg. Multiple hypothesis tracking revisited: Blending in modern appearance model. *ICCV*, 2015. 1, 2, 7

[31] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. *ANIPS*, 2012. 1, 4

[32] C.-H. Kuo and R. Nevatia. How does person identity recognition help multi-person tracking? *CVPR*, 2011. 1, 2

[33] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese. Lerning an image-based motion context for multiple people tracking. *CVPR*, 2014. 2, 6, 7

[34] 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. 5, 6, 7

[35] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. *ICCV. 1st Workshop on Modeling, Simulation and Visual Analysis of Large Crowds*, 2011. 1, 2, 5, 7

[36] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn. Branch-and-price global optimization for multi-view multi-object tracking. *CVPR*, 2012. 2

[37] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *ArXiv e-prints*, 2015. 2

[38] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybrid-boosted multi-target tracker for crowded scene. *CVPR*, 2009. 6

[39] W. Luo, X. Zhao, and T.-K. Kim. Multiple object tracking: A review. *arXiv:1409.7618 [cs]*, Sept. 2014. 2

[40] A. Milan, L. Leal-Taixé, K. Schindler, and I. Reid. Joint tracking and segmentation of multiple targets. In *CVPR*, 2015. 2, 7

[41] A. Milan, K. Schindler, and S. Roth. Multi-target tracking by discrete-continuous energy minimization. *TPAMI*, 2016. 7

[42] A. Milan and S. R. R. Schindler. Continuous energy minimization for multitarget tracking. *TPAMI*, 2014. 7

[43] P. M. N. McLaughlin, J. Martinez Del Rincon. Enhancing linear programming with motion modeling for multi-target tracking. *WACV*. 7

[44] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: modeling social behavior for multi-target tracking. *ICCV*, 2009. 1, 2, 5

[45] H. Pirsiavash, D. Ramanan, and C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. *CVPR*, 2011. 2, 7

[46] H. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, and I. R. A. Dick. Joint probabilistic data association revisited. *ICCV*, 2015. 7

[47] P. Scovanner and M. Tappen. Learning pedestrian dynamics from the real world. *ICCV*, 2009. 2

[48] F. Solera, S. Calderara, and R. Cucchiara. Learning to divide and conquer for online multi-target tracking. *ICCV*, 2015. 7

[49] Y. Taigman, Y. Ming, M. Ranzato, and L. Wolf. DeepFace: Closing the gap to human-level performance in face verification. *CVPR*, 2014. 2, 3

[50] H. Wang, A. Cruz-Roa, A. Basavanhally, H. Gilmore, N. Shih, M. Feldman, J. Tomaszewski, F. Gonzalez, and A. Madabhushi. Cascaded ensemble of convolutional neural networks and handcrafted features for mitosis detection, 2014. 3

[51] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *ArXiv e-prints*, 2015. 2

[52] S. Wang and C. Fowlkes. Learning optimal parameters for multi-target tracking. *BMVC*, 2015. 7

[53] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992. 4

[54] Z. Wu, T. Kunz, and M. Betke. Efficient track linking methods for track graphs using network-flow and set-cover techniques. *CVPR*, 2011. 2

[55] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. *ICCV*, 2015. 7

[56] K. Yamaguchi, A. Berg, L. Ortiz, and T. Berg. Who are you with and where are you going? *CVPR*, 2011. 2

[57] B. Yang and R. Nevatia. An online learned crf model for multi-target tracking. *CVPR*, 2012. 2

[58] J. Yoon, H. Yang, J. Lim, and K. Yoon. Bayesian multi-object tracking using motion context from multiple objects. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015. 7

[59] Q. Yu, G. Medioni, and I. Cohen. Multiple target tracking using spatio-temporal Markov chain Monte Carlo data association. *CVPR*, 2007. 2

[60] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. *CVPR*, 2015. 2, 3

[61] A. Zamir, A. Dehghan, and M. Shah. Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. *ECCV*, 2012. 1, 2, 5

[62] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. *CVPR*, 2015. 2, 3

[63] C. Zhang and Z. Zhang. Improving multiview face detection with multi-task deep convolutional neural networks. *IEEE Winter Conference on Applications of Computer Vision*, 2014. 4

[64] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. *CVPR*, 2008. 1, 2, 5, 7