Non-Markovian Globally Consistent Multi-Object Tracking

Supplementary Material

Andrii Maksai[1], Xinchao Wang[2], François Fleuret[3], and Pascal Fua[1]

[1]Computer Vision Laboratory, EPFL, Lausanne, Switzerland, {firstname.lastname}@epfl.ch
[2]Beckman Institute, UIUC, Illinois, USA {firstname}@illinois.edu
[3]IDIAP Reseach Institute, Martigny, Switzerland, {firstname.lastname}@idiap.ch

In Section 1, we provide the full definitions of the scoring functions n and m, described in Section 3.3 of the paper. In Section 2, we provide additional details of the optimizations used to improve the output of other method and to learn the patterns, described in Sections 4.1 and 4.2 of the paper. In Section 3, we provide full results of all methods on all datasets with various metrics, extending on the results from Section 6.4 of the paper. We also provide textual description of datasets and baselines. We describe component evaluation in Section 4, and, in Section 5, we provide evaluation of the computational requirements of our approach, in addition to he results given in Section 6.4 of the paper.

1. Full definitions of n and m functions

These functions are used to score the edges of a trajectory to compute how likely is it that a particular trajectory follows a particular pattern. As stated in Section 3.3 of the paper:

$$C(T,P,A) = \frac{\sum_{t\in T} M(t, p_{A(t_1)})}{\sum_{t\in T} N(t, p_{A(t_1)})}, \tag{1}$$

$$N(t,p) = n(I,t_1,p) + n(t_{|t|},O,p) + \sum_{1\le j\le |t|-1} n(t_j,t_{j+1},p), \tag{2}$$

$$M(t,p) = m(I,t_1,p) + m(t_{|t|},O,p) + \sum_{1\le j\le |t|-1} m(t_j,t_{j+1},p), \tag{3}$$

where T is a set of edges of all trajectories, A is the assignment between a trajectory and a pattern, and P is a set of patterns. As shown in (2) and (3), to score a trajectory we score all its edges plus the edges from I, the node denoting the beginnings of the trajectories, and the ones to O, the node denoting the ends of trajectories. As mentioned in the paper, we want $N(t,p)$ to reflect the full length of the trajectory and the pattern, and $M(t,p)$ to reflect the total length of the aligned trajectory and the pattern. In what follows, we provide definitions of n and m in all cases.

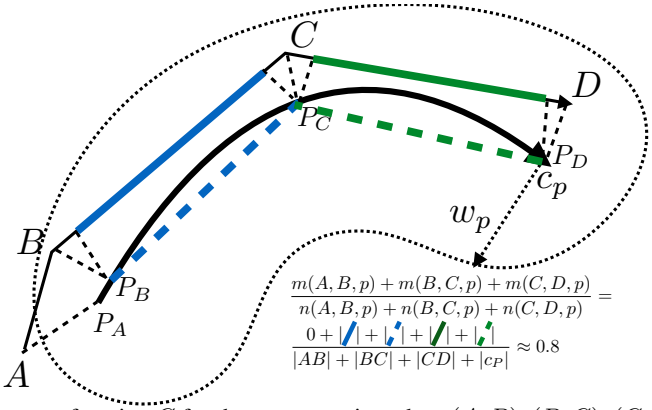


Figure 1. Example of computing the cost function C for three consecutive edges (A,B), (B,C), (C,D). Dotted line around the pattern centerline c_p shows the area within the distance w_p to the pattern. The denominator contains the total length of the edges plus the total length of the pattern, while the numerator contains the parts aligned with each other (in green and blue). The edge (A,B) is not counted as aligned, because A is further from the pattern than its width w_p.

In **Table 1**, we show how to compute n and m for edges that link two detections and follow some pattern. For n we take the pattern length to be positive or negative depending on whether the projection of the edge to the pattern is positive or negative. For m, we penalize edges far from the pattern and edges going in the direction opposite to the pattern, in two different ways, which gives rise to the three cases shown in the table. In **Table 2**, we show how to compute n when one of the nodes is I or O, denoting the start or the end of a trajectory. A special case arises when a node is in the first or the last frame of an input batch, and a trajectory going through it does not need to follow the pattern completely. This results in a total of two cases we show in the table. In **Table 3**, we show the two cases when we assign the transition to no pattern $\emptyset$, one case when we assign a normal edge joining two detections, and the other when we assign edge from I or to O, indicating the beginning or the edge of the trajectory.

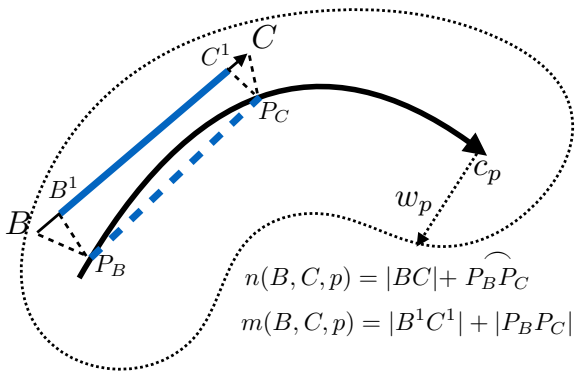
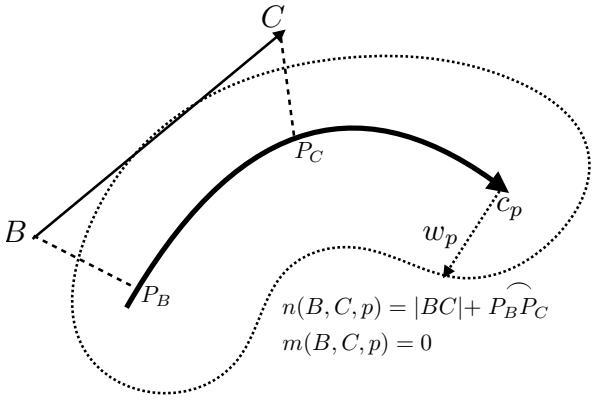
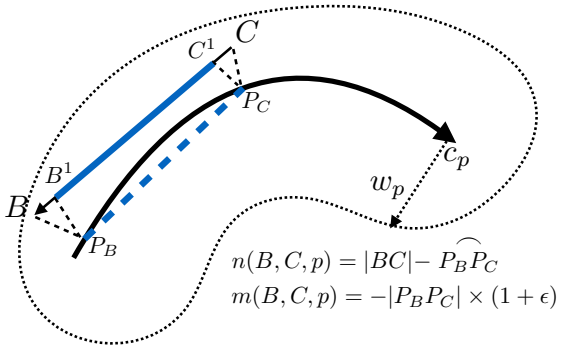
Case	Explanation	Figure
Normal edge aligned with the pattern: B and C are within distance w_p to the pattern centerline, P_B is earlier on the curve c_p that P_C.	For the edge (B, C), we find the nearest neighbor of the two endpoints on the pattern, namely P_B and P_C. Formally, we have $P_B = \arg\min_{x \in c_p} \|\|B - x\|\|$. Then we project P_B and P_C orthogonally back onto (B, C). This guarantees that $m(B, C, p) \leq n(B, C, p)$ with equality when (B, C) and (P_B, P_C) are two parallel segments of equal length, and also penalizes deviations from the pattern in direction.	
Normal edge aligned with the pattern: B and C are further away than w_p from the pattern centerline, P_B is earlier on the curve c_p that P_C.	$n(B, C, p)$ is calculated in the same way as done in the previous case. To penalize deviations from the pattern in distance, we take $m(B, C, p) = 0$	
Normal edge not aligned with the pattern: P_B is later on the curve c_p that P_C.	To keep our rule about N being the sum of lengths of pattern and trajectory, we need to subtract the length of arc from P_B to P_C, as it is pointing in the direction opposite to the pattern. To penalize this behavior, we take $m(B, C, p)$ to be $-\|P_B P_C\|$, multiplied by $1 + \epsilon$. In practice, we use $\epsilon = 1$.	

Table 1. Table describing full definitions of n and m in normal cases, when edges between two detections align with a pattern. They all follow naturally from the rule about N being the sum of length of trajectory and the pattern, and M being the sum of aligned lengths.

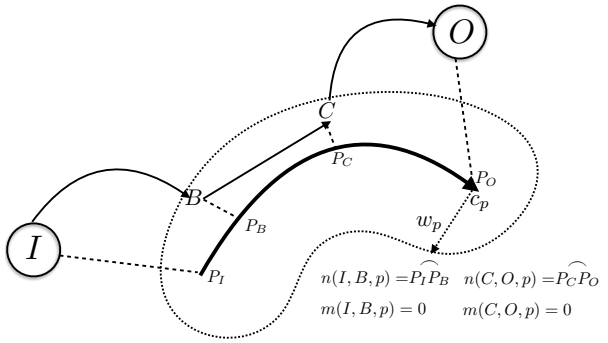
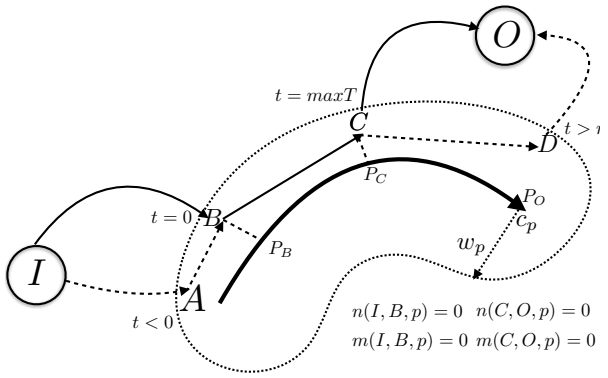
Case	Explanation	Figure
Edge from the source to a normal node / from a normal node to the sink	To keep our rule about N being the sum of lengths of pattern and trajectory, we need to add the length from the beginning of the pattern to the point closest to the node on the centerline / from the point closest to the node on the centerline to the end of the pattern. Since we didn't observe any parts of trajectory aligned with these parts, we take $m = 0$.	
Edge from the source to a normal node in the first frame of the batch / from a normal node in the last frame of the batch to the sink	We assume that our trajectories follow the path completely. However, this might be not true, which we observe from the middle, that is, the ones that begin in the first frame of the batch or end in the last frame. In that case we don't need to add the part of the pattern before / after the current point closest to the node, which is why we take $n = m = 0$.	

Table 2. Table describing full definitions of n and m in corner cases when one of the edges go through I or O, indicating the beginning or the end of a trajectory. They all follow naturally from the rule about N being the sum of length of trajectory and the pattern, and M being the sum of aligned lengths.

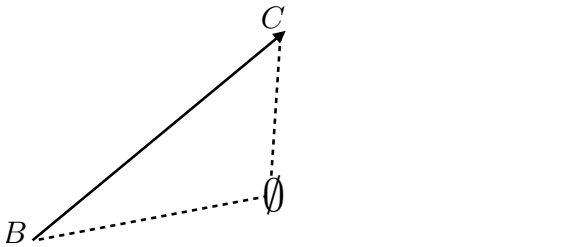
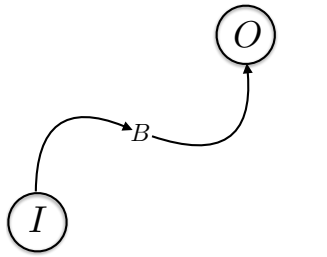
Case	Explanation	Figure
Normal edge aligned to no pattern	To keep our rule about N being the sum of lengths, we take n to be just the length of the trajectory, since we assume the length of empty pattern to be zero. We penalize such assignment by a fixed constant $\epsilon_\emptyset$, taking m to be n multiplied by such constant. In practice, we keep $\epsilon_\emptyset = 0.3$ when training from ground truth, or $\epsilon_\emptyset = -3$ otherwise.	 $n(B, C, p) = \|BC\|$ $m(B, C, p) = \|BC\| \times (1 + \epsilon_\emptyset)$
Edge from the source / to the sink, aligned to no pattern	To keep our rule about N, we take both $n = m = 0$.	 $n(I, B, p) = n(B, O, p) = 0$ $m(I, B, p) = m(B, O, p) = 0$

Table 3. Table describing full definitions of n and m in corner cases when there is no pattern. They all follow naturally from the rule about N being the sum of length of trajectory and the pattern, and M being the sum of aligned lengths.

2. Details of the optimization scheme

Here we provide details on our optimization schemes that improve the tracking output of other method and learn patterns, outlined in Sections 4.1 and 4.2 of the paper, respectively.

2.1. Tracking

As noted in the paper, we introduce the binary variables o_{ij}^p, denoting the number of people transitioning between the detections i and j, following pattern p. We put the following constraints on them:

$$\begin{aligned} \forall i \in \mathcal{D} \cup O \sum_{(i,j)\in\mathcal{E}, p\in P^*} o_{ij}^p &= 1 \,, \\ \forall j \in \mathcal{D}, p \in P^* \sum_{(i,j)\in\mathcal{E}} o_{ij}^p &= \sum_{(j,k)\in\mathcal{E}} o_{jk}^p \,. \end{aligned} \tag{4}$$

Then, during binary search, we fix a particular value of α, and check whether the problem constrained by (4) and the following has a feasible point:

$$\sum_{(i,j)\in T, p\in P^*} (m(i,j,p) - \alpha n(i,j,p)) o_{ij}^p \geq 0 \tag{5}$$

If a feasible point exists, we pick a value of α to be the lower bound of the best α, for which the problem is feasible, otherwise we pick it as an upper bound. We start with the upper bound of 1 and lower bound of 0, and pick α as an average between the upper and the lower bound (dichotomy). We repeat this process 10 times, allowing us to find the correct value of α with the margin of 2^{-10}.

2.2. Patterns

As noted in the paper, we introduce the binary variables a_{tp} denoting that a ground truth trajectory t follows the pattern p, and binary variables b_p denoting whether at least one trajectory follows the pattern p.

$$\begin{aligned} a_{tp} &\in \{0,1\} \,, \forall t \in T^*, p \in \mathcal{P} \,, \\ b_p &\in \{0,1\} \,, \forall p \in \mathcal{P} \,, \\ \sum_{p\in\mathcal{P}} a_{tp} &= 1 \,, \forall t \in T^* \,, \\ a_{tp} &\leq b_p \,, \forall t \in T^*, p \in \mathcal{P} \,. \end{aligned} \tag{6}$$

We then do the same binary search as described above to find the highest α, for which there exists a feasible point to a set of constraints (6) and the following:

$$\begin{aligned} &\sum_{t\in T^*} \sum_{p\in\mathcal{P}} (m(t,p) - \alpha n(t,p)) a_{tp} \geq 0 \,, \\ &\sum_{p\in\mathcal{P}} b_p \leq \alpha_p \,, \\ &\sum_{p\in\mathcal{P}} b_p M(p) \leq \alpha_c \,. \end{aligned} \tag{7}$$

We do five iterations of binary search, and we obtain the right value of α with precision of 2^{-5}. To create a set of all possible patterns $\mathcal{P}$ we combine the set of all possible trajectories in the current batch (taking only those that start after the beginning of the batch and end before the end of the batch to make sure they represent *full* patterns of movement) with a set of possible lengths. For all datasets except **Station**, our set of possible lengths is $\{0.5, 1, 3, 5, 7, 9, 11, 13, 15, 17\}$, while for the **Station** dataset we use $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ of the tracking area, since we don't know the exact sizes of the tracking area, but only estimated homography between the ground and image plane.

3. Full results

Here we provide the full results of all the methods on all the datasets, after the textual description of datasets and baselines. Tables 4, 5 are the full versions of Table 2 of the paper. Table 6 reports details on **Duke** dataset in details, as reported on the MOTChallenge website. In Tables 4, 5, we compare the original output of the method with the improvements brought by our approach in both supervised and unsupervised manner, denoted "-i" and "-o", respectively. In Table 7, we compare the methods when using the ground truth set of detections as input. As in the paper, we report the results for the matching distances of 3m (0.1 of the tracking area for the **Station** and **Rene** datasets), and for $\mathbf{IDF}_1$ metric we also show results for 1m to indicate that the ranking of the methods does not change, but the improvement brought by our methods is less visible due to reconstruction errors when we estimate the 3D position of the person from the bounding box. This fact is especially highlighted by the Table 7, where difference in the metric computed for distances of 3m. and 1m. is especially large.

Specifically, We report the $\mathbf{IDF}_1$, identity level precision and recall IDPR and IDRC defined in [13], as well as **MOTA**, precision and recall PR and RC, and the number of mostly tracked MT, partially tracked PT and mostly lost trajectories ML defined in [2].

Our evaluation of **Duke** dataset is available on MOTChallenge website under the name **PT_BIPCC**, and comparison on **MOT16** under the name PT.

Readers may note that often we observe an increase in the number of mostly lost trajectories and drop in recall. One of our optimization parameters controls whether or not to remove trajectories assigned to no pattern during post-processing (see Sec. 3.1 and 6.3). Removals reduce the number of false positives, but may discard tracks for some partially tracked people, which don't follow any pattern. This increases the ML and lowers the recall, as observed by the reviewer. This happens in large part because both contrast and visibility are low, resulting in poor detection quality, for example on ETH dataset.

3.1. Datasets

Duke. A dataset [13] with 8 cameras recording movements of people on various places of Duke university campus at 60fps, containing more than an hour of recordings.

Town. A sequence from the 2DMOT2015 benchmark [9]: a lively street where people walk in different directions.

ETH and **Hotel**. Sequences from the BIWI Walking Pedestrians dataset [12] that were originally used to model social behavior. In these datasets, using image and appearance information for tracking is difficult, due to recordings with an almost vertical viewing angle and low visibility in the **ETH**.

Station. A one hour-long recording of Grand Central station in New York with several thousands of annotated pedestrian tracks [23]. It was originally used for trajectory prediction in crowds.

MOT16. Sequences from the MOT Challenge 2016 [10]. We used MOT16-01 to evaluate the supervised approach because it features training and testing data recorded at the same place. By contrast, MOT16-08 does not and we used it to evaluate the unsupervised approach. Unfortunately, the other sequences are unsuitable for our current implementation because they involve either a moving camera, meaning there is no fixed scene to learn the patterns from, or only very few trajectories traversing the scene for training purposes.

Rene. A five-minute long sequence of traffic at a street junction [6]. Since only 30 seconds of it are annotated, we ran only the unsupervised approach on the whole sequence and used the annotated frames for evaluation purposes.

3.2. Baselines

MDP [20] formulates MOT as learning Markov Decision Process (MDP) policy and relies on reinforcement learning to do so. At the time of writing, this was the highest-ranking approach on the 2DMOT2015 [9] benchmark with a publicly available implementation.

SORT [3] is a real-time Kalman filter-based MOT approach. At the time of writing, this was the second highest-ranking approach on 2DMOT2015 benchmark with a publicly available implementation.

RNN [11] uses recurrent neural networks to predict the motion of people and perform MOT in real time. It does not require any appearance information, but only the bounding boxes coordinates. In our experiments, it outperformed all other methods that do not use appearance information.

KSP [1] is a simple approach to MOT that formulates the MOT problem as finding K Shortest Paths in spatio-temporal graph, without using appearance information.

2DMOT2015 Top Scoring Methods [4, 14, 18, 7, 21, 8, 19, 22] to which we will refer by the name that appears in the official scoreboard [9].

Method	Dataset	**IDF**$_1$	IDPR	IDRC	**MOTA**	PR	RC	MT	PT	ML
EAMTT	**Town**	0.72 (0.59)	0.76	0.68	0.73	0.92	0.82	158	68	20
EAMTT-i	**Town**	0.80 (0.63)	0.84	0.76	0.73	0.91	0.82	165	59	22
EAMTT-o	**Town**	0.82 (0.65)	0.83	0.80	0.74	0.89	0.86	182	44	20
JointMC	**Town**	0.75 (0.63)	0.90	0.65	0.64	0.95	0.68	128	54	64
JointMC-i	**Town**	0.77 (0.64)	0.91	0.66	0.64	0.95	0.68	129	52	65
JointMC-o	**Town**	0.76 (0.62)	0.88	0.67	0.65	0.93	0.71	138	50	58
MHT_DAM	**Town**	0.56 (0.45)	0.82	0.42	0.40	0.90	0.46	55	98	93
MHT_DAM-i	**Town**	0.56 (0.45)	0.83	0.42	0.40	0.90	0.46	59	90	97
MHT_DAM-o	**Town**	0.57 (0.45)	0.81	0.44	0.42	0.89	0.48	63	94	89
NOMT	**Town**	0.71 (0.62)	0.83	0.63	0.65	0.94	0.71	122	76	48
NOMT-i	**Town**	0.76 (0.65)	0.87	0.68	0.66	0.93	0.72	135	61	50
NOMT-o	**Town**	0.75 (0.63)	0.83	0.68	0.66	0.91	0.75	144	59	43
SCEA	**Town**	0.56 (0.43)	0.83	0.42	0.40	0.90	0.46	56	95	95
SCEA-i	**Town**	0.58 (0.45)	0.87	0.44	0.44	0.95	0.47	62	89	95
SCEA-o	**Town**	0.58 (0.43)	0.80	0.45	0.43	0.89	0.50	65	94	87
TDAM	**Town**	0.60 (0.48)	0.71	0.52	0.39	0.78	0.56	70	112	64
TDAM-i	**Town**	0.60 (0.48)	0.73	0.51	0.41	0.80	0.56	69	110	67
TDAM-o	**Town**	0.59 (0.45)	0.67	0.54	0.37	0.74	0.60	82	108	56
TSML_CDE	**Town**	0.68 (0.58)	0.75	0.63	0.72	0.95	0.79	143	79	24
TSML_CDE-i	**Town**	0.76 (0.62)	0.84	0.70	0.73	0.95	0.79	150	68	28
TSML_CDE-o	**Town**	0.78 (0.62)	0.82	0.74	0.74	0.92	0.83	161	68	17
CNNTCM	**Town**	0.58 (0.46)	0.79	0.46	0.45	0.90	0.53	63	110	73
CNNTCM-i	**Town**	0.61 (0.46)	0.80	0.49	0.48	0.90	0.55	73	96	77
CNNTCM-o	**Town**	0.62 (0.46)	0.77	0.52	0.48	0.87	0.59	85	95	66
KSP	**Town**	0.41 (0.26)	0.47	0.36	0.64	0.93	0.73	107	105	34
KSP-i	**Town**	0.69 (0.42)	0.78	0.61	0.65	0.93	0.73	118	91	37
KSP-o	**Town**	0.69 (0.42)	0.76	0.63	0.64	0.91	0.75	122	88	36
MDP	**Town**	0.59 (0.45)	0.65	0.55	0.50	0.81	0.68	103	97	46
MDP-i	**Town**	0.66 (0.49)	0.72	0.61	0.54	0.83	0.71	116	82	48
MDP-o	**Town**	0.63 (0.45)	0.66	0.61	0.50	0.79	0.73	113	94	39
RNN	**Town**	0.48 (0.30)	0.52	0.45	0.60	0.88	0.77	122	103	21
RNN-i	**Town**	0.59 (0.36)	0.65	0.55	0.61	0.90	0.76	125	98	23
RNN-o	**Town**	0.53 (0.34)	0.57	0.50	0.59	0.89	0.77	125	99	22
SORT	**Town**	0.62 (0.46)	0.81	0.50	0.57	0.98	0.61	49	152	45
SORT-i	**Town**	0.72 (0.47)	0.85	0.62	0.64	0.95	0.69	96	109	41
SORT-o	**Town**	0.65 (0.46)	0.83	0.60	0.60	0.90	0.65	174	58	14

Table 4. Full results for all methods on the **Town** dataset, when using our detections as input and using the results of state-of-the-art trackers as input. Number in brackets in **IDF**$_1$ column indicates result for the distance of 1 m.

DM [16, 17] decomposes the tracking graph into subgraphs and relies on strong matching models. It won the ECCV 2016 Multiple Object Tracking challenge.

BIPCC [13] solves binary integer problem of optimally grouping observations into clusters of detections of similar appearances, and delivers results with moderate recall, but very high precision with few identity switches.

Method	Dataset	**IDF**$_1$	IDPR	IDRC	**MOTA**	PR	RC	MT	PT	ML
KSP	**ETH**	0.45 (0.15)	0.45	0.45	0.47	0.72	0.71	182	148	22
KSP-i	**ETH**	0.62 (0.18)	0.71	0.54	0.48	0.75	0.57	134	144	74
KSP-o	**ETH**	0.57 (0.18)	0.59	0.67	0.49	0.67	0.76	217	121	14
MDP	**ETH**	0.55 (0.20)	0.63	0.48	0.40	0.79	0.60	113	194	45
MDP-i	**ETH**	0.58 (0.21)	0.76	0.46	0.41	0.83	0.50	105	143	104
MDP-o	**ETH**	0.58 (0.21)	0.64	0.62	0.41	0.72	0.69	157	146	49
RNN	**ETH**	0.51 (0.21)	0.54	0.49	0.48	0.80	0.73	170	162	20
RNN-i	**ETH**	0.54 (0.21)	0.76	0.39	0.48	0.85	0.44	68	184	100
RNN-o	**ETH**	0.54 (0.21)	0.40	0.47	0.47	0.64	0.76	205	127	20
SORT	**ETH**	0.67 (0.29)	0.82	0.57	0.50	0.87	0.61	130	175	47
SORT-i	**ETH**	0.66 (0.26)	0.84	0.55	0.49	0.86	0.56	136	129	87
SORT-o	**ETH**	0.67 (0.29)	0.79	0.68	0.49	0.80	0.70	167	148	37
KSP	**Hotel**	0.44 (0.14)	0.33	0.65	0.32	0.48	0.94	270	40	6
KSP-i	**Hotel**	0.53 (0.17)	0.38	0.75	0.33	0.47	0.94	273	35	8
KSP-o	**Hotel**	0.53 (0.17)	0.38	0.77	0.30	0.46	0.94	276	32	8
MDP	**Hotel**	0.40 (0.12)	0.34	0.46	0.33	0.47	0.64	133	92	91
MDP-i	**Hotel**	0.50 (0.13)	0.43	0.37	0.38	0.60	0.52	83	110	123
MDP-o	**Hotel**	0.37 (0.10)	0.28	0.47	0.30	0.40	0.67	143	105	68
RNN	**Hotel**	0.40 (0.14)	0.30	0.58	0.39	0.46	0.90	252	45	19
RNN-i	**Hotel**	0.40 (0.14)	0.30	0.59	0.39	0.46	0.90	258	38	20
RNN-o	**Hotel**	0.39 (0.13)	0.29	0.56	0.38	0.46	0.90	256	41	19
SORT	**Hotel**	0.54 (0.20)	0.45	0.68	0.37	0.55	0.82	207	87	22
SORT-i	**Hotel**	0.60 (0.20)	0.46	0.78	0.47	0.52	0.90	240	60	16
SORT-o	**Hotel**	0.58 (0.20)	0.46	0.78	0.35	0.53	0.88	238	64	14
KSP	**Station**	0.32	0.27	0.40	0.23	0.61	0.90	10166	1985	211
KSP-i	**Station**	0.42	0.35	0.52	0.19	0.60	0.91	10296	1879	187
KSP-o	**Station**	0.40	0.32	0.53	2.27	0.55	0.92	10597	1576	189
MDP	**Station**	0.48	0.39	0.63	0.51	0.56	0.90	9362	2293	437
MDP-i	**Station**	0.47	0.36	0.65	0.52	0.51	0.92	10047	1771	544
MDP-o	**Station**	0.47	0.37	0.66	0.50	0.52	0.92	10010	1930	422
RNN	**Station**	0.30	0.24	0.37	0.40	0.58	0.90	9826	2333	203
RNN-i	**Station**	0.30	0.24	0.38	0.41	0.59	0.90	9900	2260	202
RNN-o	**Station**	0.30	0.25	0.39	0.40	0.57	0.90	9898	2265	199
SORT	**Station**	0.50	0.50	0.50	0.32	0.71	0.72	5557	6181	624
SORT-i	**Station**	0.50	0.47	0.54	0.31	0.69	0.78	6996	4882	484
SORT-o	**Station**	0.52	0.48	0.57	0.31	0.67	0.79	7154	4703	505
KSP	**Rene**	0.48	0.48	0.49	0.31	0.89	0.38	11	3	13
KSP-i	**Rene**	0.55	0.69	0.49	0.36	0.65	0.47	15	8	4
MDP	**Rene**	0.66	0.55	0.63	0.52	0.58	0.51	13	7	7
MDP-i	**Rene**	0.69	0.57	0.67	0.54	0.58	0.53	17	6	4
RNN	**Rene**	0.54	0.54	0.52	0.40	0.77	0.43	12	3	12
RNN-i	**Rene**	0.55	0.54	0.53	0.40	0.77	0.43	14	7	6
SORT	**Rene**	0.59	0.63	0.27	0.48	0.24	0.61	4	7	16
SORT-i	**Rene**	0.66	0.77	0.31	0.49	0.24	0.77	14	9	4

Table 5. Full results for all methods on all the datasets except **Town** and **Duke**, when using our detections as input and using the results of state-of-the-art trackers as input. Number in brackets in **IDF**$_1$ column indicates result for the distance of 1 m.

Method	Sequence	**IDF$_1$**	IDPR	IDRC	**MOTA**	MOTP	MT	ML	IDs	Frags
BIPCC	Easy-1	57.3	91.2	41.8	43.0	79.0	24	46	39	75
BIPCC-i	Easy-1	57.8	91.9	42.2	42.9	79.0	24	46	41	75
BIPCC	Easy-2	68.2	69.3	67.1	44.8	78.2	133	38	60	184
BIPCC-i	Easy-2	69.2	70.4	68.0	44.7	78.2	133	39	52	172
BIPCC	Easy-3	60.3	78.9	48.8	57.8	77.5	52	22	16	36
BIPCC-i	Easy-3	59.8	78.2	48.4	57.8	77.5	52	22	19	36
BIPCC	Easy-4	73.5	88.7	62.8	63.2	80.2	36	18	7	20
BIPCC-i	Easy-4	76.0	91.7	64.9	63.2	80.2	36	18	9	20
BIPCC	Easy-5	73.2	83.0	65.4	72.8	80.4	107	17	54	139
BIPCC-i	Easy-5	73.3	83.0	65.6	72.6	80.4	107	17	46	132
BIPCC	Easy-6	77.2	87.5	69.1	73.4	80.2	142	27	55	127
BIPCC-i	Easy-6	80.9	91.7	72.4	73.4	80.2	142	27	58	127
BIPCC	Easy-7	80.5	93.6	70.6	71.4	74.7	69	13	23	86
BIPCC-i	Easy-7	80.5	93.6	70.6	71.4	74.7	69	13	23	86
BIPCC	Easy-8	72.4	92.2	59.6	60.7	76.7	102	53	46	134
BIPCC-i	Easy-8	72.7	92.2	60.0	60.9	76.6	103	52	42	135
BIPCC	Hard-1	52.7	92.5	36.8	37.8	78.1	6	34	55	103
BIPCC-i	Hard-1	52.5	91.9	36.7	37.4	78.1	6	35	61	106
BIPCC	Hard-2	60.6	65.7	56.1	47.3	76.5	68	12	194	298
BIPCC-i	Hard-2	61.0	66.0	56.7	46.6	76.5	66	12	194	291
BIPCC	Hard-3	62.7	96.1	46.5	46.7	77.9	24	4	6	12
BIPCC-i	Hard-3	62.7	96.1	46.5	46.7	77.9	24	4	6	12
BIPCC	Hard-4	84.3	86.0	82.7	85.3	81.5	21	0	1	9
BIPCC-i	Hard-4	92.3	93.6	91.0	85.5	81.4	21	0	2	9
BIPCC	Hard-5	81.9	90.1	75.1	78.3	80.7	57	2	13	37
BIPCC-i	Hard-5	81.9	90.1	75.1	78.3	80.7	57	2	13	37
BIPCC	Hard-6	64.1	81.7	52.7	59.4	76.7	85	23	225	369
BIPCC-i	Hard-6	64.7	82.4	53.3	59.4	76.7	85	23	230	369
BIPCC	Hard-7	59.6	81.2	47.1	50.8	73.3	43	23	148	218
BIPCC-i	Hard-7	59.8	81.4	47.2	50.6	73.3	42	23	145	203
BIPCC	Hard-8	82.4	94.9	72.8	73.0	75.9	34	5	10	27
BIPCC-i	Hard-8	82.4	94.9	72.8	73.0	75.9	34	5	10	27

Table 6. Full results of comparison on **Duke** dataset, compared to results of **BIPCC**.

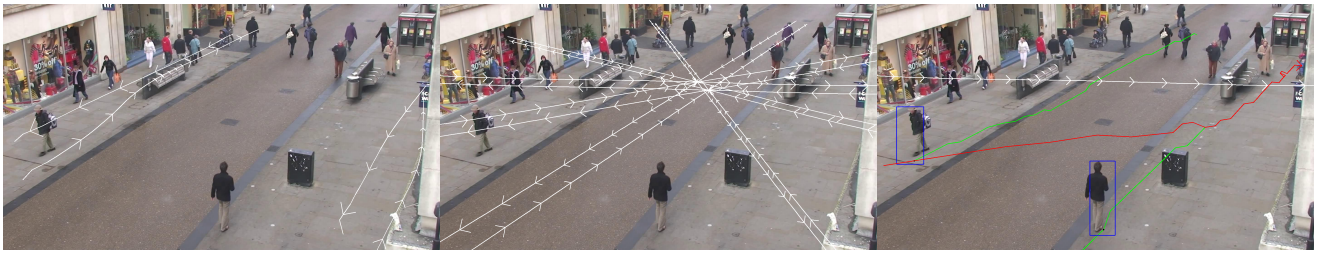
Method	Dataset	**IDF$_1$**	IDPR	IDRC	**MOTA**	PR	RC	MT	PT	ML
KSP	**Town**	0.56 (0.47)	0.55	0.57	0.87	0.93	0.97	226	8	12
MDP	**Town**	0.87 (0.84)	0.92	0.82	0.87	0.99	0.89	184	38	24
RNN	**Town**	0.65 (0.57)	0.65	0.65	0.85	0.95	0.95	222	19	5
SORT	**Town**	0.88 (0.85)	0.93	0.84	0.90	1.00	0.90	203	34	9
OUR	**Town**	**0.97 (0.92)**	0.97	0.97	**0.98**	1.00	1.00	245	1	0
KSP	**ETH**	0.59 (0.12)	0.58	0.60	0.70	0.87	0.89	287	56	9
MDP	**ETH**	0.89 (0.18)	0.91	0.87	0.85	0.95	0.91	300	42	10
RNN	**ETH**	0.65 (0.16)	0.64	0.65	0.73	0.89	0.90	289	62	1
SORT	**ETH**	**0.93 (0.20)**	0.98	0.88	0.85	0.97	0.87	307	31	14
OUR	**ETH**	0.92 (0.19)	0.92	0.92	**0.94**	0.98	0.98	347	5	0
KSP	**Hotel**	0.60 (0.21)	0.61	0.58	0.74	0.90	0.86	217	69	30
MDP	**Hotel**	0.85 (0.33)	0.87	0.83	0.84	0.95	0.90	249	37	30
RNN	**Hotel**	0.70 (0.28)	0.69	0.71	0.78	0.91	0.94	284	29	3
SORT	**Hotel**	0.88 (0.36)	0.97	0.81	0.82	0.99	0.83	191	107	18
OUR	**Hotel**	**0.94 (0.38)**	0.94	0.94	**0.97**	1.00	1.00	314	1	1
KSP	**Station**	0.45	0.44	0.45	**0.80**	0.93	0.95	10957	832	573
MDP	**Station**	0.75	0.70	0.80	0.68	0.81	0.93	464	67	51
RNN	**Station**	0.40	0.39	0.40	0.68	0.90	0.94	10870	1244	248
SORT	**Station**	**0.72**	0.85	0.63	0.70	1.00	0.74	4968	6481	913
OUR	**Station**	0.70	0.62	0.62	0.77	0.99	0.99	579	3	0

Table 7. Full results for all combinations of methods and datasets, when using our set of ground truth detections. Number in brackets in **IDF$_1$** column indicates result for the distance of 1 m.

4. Component Evaluation

Method	Learned patterns (OUR)				Straight line patterns				Markovian smoothness term			
Approach	**Town**	**ETH**	**Hotel**	**Station**	**Town**	**ETH**	**Hotel**	**Station**	**Town**	**ETH**	**Hotel**	**Station**
KSP	0.28	0.17	0.11	0.10	0.19	0.12	0.07	0.05	0.13	0.07	0.04	0.03
MDP	0.07	0.03	0.10	-0.01	0.06	0.02	0.02	-0.02	0.06	0.02	0.02	-0.02
RNN	0.11	0.03	0.00	0.00	0.10	0.02	0.00	0.00	0.05	0.01	-0.01	-0.01
SORT	0.10	0.00	0.06	0.00	0.06	0.00	0.03	0.00	0.04	0.00	-0.01	-0.03

Table 8. $\mathbf{IDF}_1$ improvement for each method and dataset for our method with learned patterns (**left**), for our method with patterns replaced by a pencil of lines, which still forces trajectories to start and end at the borders of the tracking area (**middle**), and for a method where transition cost is based on the local smoothness term, second order difference between coordinates of 3 consecutive detections in a trajectory (**right**).



(a) (b) (c)

Figure 2. **(a)** Examples of learned patterns on **Town** dataset. Note that there are only two prevalent directions in which people move. Patterns shown in white. **(b)** Pencil of lines representing straight line patterns traversing the scene in all directions. Patterns shown in white. **(c)** Example of an error made when we are using straight line patterns. Two real trajectories (in green) are incorrectly merged via false detections, producing a trajectory (in red) that closely follows one of the patterns (in white). However, in reality this pattern does not exist, but we used it because we didn't have a learning component. Note, that produced trajectory still starts at the boundary of the image and traverses the scene completely. Even without the learning procedure, our patterns force this. If we replace this non-Markovian constraint by a local smoothness term, errors are numerous, with many trajectories split in the middle.

Component evaluation To evaluate the importance of learning generic patterns such as ours as opposed to simpler ones, or an even simpler smoothness constraint, we introduce two more baselines. In the first, we take patterns to be straight lines crossing the scene in every possible direction. This is still non-Markovian as it forces trajectories to cross the scene completely, starting at one border and going to another. In the second, we find a set of trajectories through our tracking graph that minimizes the second order difference between triplets of consecutive locations, forcing trajectories to be locally smooth. This is Markovian in nature and does not require trajectories to cross the scene. In the first case, average improvement for all methods drops to (0.11, 0.02, 0.03, 0.02) from (0.16, 0.05, 0.04, and 0.04) as reported in Table 2 of the paper. In the second case, we observe a steeper drop to (0.07, 0.02, 0.01, 0.00). The difference in results is largest on **Station** dataset where non-linear non-Markovian patterns prevent trajectories from being terminated in stationary crowds, and on the **Hotel** dataset where it is difficult to differentiate between trajectories that traverse the scene and that end in the middle of the scene, entering the hotel. The detailed breakdown is given in Table 8.

Note that while using straight line patterns frees us from the learning step, it does not deliver much of a benefit in terms of optimization speed. As shown in the example of Figure 2, there are only four learned patterns, but if we define straight line patterns traversing the scene, their number is not known beforehand. This results in a trade-off, where picking too few patterns gives bad tracking results, while having too many of them slows the optimization scheme because of the number of possible patterns trajectories can follow.

Evaluation on Ground Truth Detections. For all baselines that accept a list of detections as input, and for which the code is available, we reran the same experiment using the ground truth detections instead of those computed by the POM algorithm [5] as before. This is a way to evaluate the performance of the linking procedure independently of that of the detections, and can be viewed as an evaluation of this component of our system. It reflects the theoretical maximum that can

be reached by all the approaches we compare, including our own. From Table 9 we observe that our approach performs very well in such setting.

Approach: Dataset:	**MDP**	**RNN**	**SORT**	**KSP**	**OUR**
Town	0.87	0.65	0.88	0.55	**0.93**
ETH	0.89	0.65	**0.93**	0.59	0.92
Hotel	0.85	0.70	0.88	0.60	**0.94**
Station	0.68	0.40	**0.72**	0.45	0.70

Approach: Dataset:	**MDP**	**RNN**	**SORT**	**KSP**	**OUR**
Town	0.87	0.85	0.90	0.87	**0.98**
ETH	0.85	0.73	0.85	0.70	**0.94**
Hotel	0.84	0.78	0.82	0.74	**0.97**
Station	0.75	0.68	0.70	**0.80**	0.77

Table 9. **IDF**$_1$ (left) and **MOTA** (right) evaluation results using ground detections. Best score for each dataset in bold.

Additional qualitative evaluation was performed on the **DataFromSky** data [15] is a one-minute long sequence of traffic recorded from a drone on an intersection. The data is provided with annotated ground truth trajectories, total of 34 vehicles. Due to the shake of the drone because of the wind, reliable detections from background subtraction are not available. However, we performed two experiments with this data. First, we learned the patterns to see if such scarce data is enough to reliably extract the motion patterns. Results are shown in Figure 3. Second, given ground truth detections we did a comparison similar to the one described in the previous paragraph. Our approach, given the learned patterns, obtained 92% **IDF**$_1$ and 94% **MOTA** scores, again showing that patterns provide valuable information for data association.

Figure 3. Frame from the **DataFromSky** dataset, with learned patterns in white.

5. Run Time Evaluation

Dataset	**Town**	**ETH**	**Hotel**	**Station**	**Station**
Frames	150	227	268	75	75
Trajectories	85	67	47	100	193
Patterns	7	5	4	26	26
Detections	2487	894	1019	1960	3724
Variables	70k	17k	18k	191k	450k
Time, s	26	4	4	160	>3600

Table 10. Optimization problem size and run time of our approach for processing a typical one min batch from each dataset.

Here we present the evaluation of running time of our approach depending on the parameters of the optimization. As mentioned in the Section 6.4 of the paper and shown in Fig. 4, the optimization time depends mostly on the number of possible transitions between people, which is controlled by D_1. The time for learning the patterns grows approximately quadratically. The number of variables in our optimization problem grows linearly with the length of the batch and number of patterns, and superlinearly with the number of people per frame (as the number of possible connections between people).

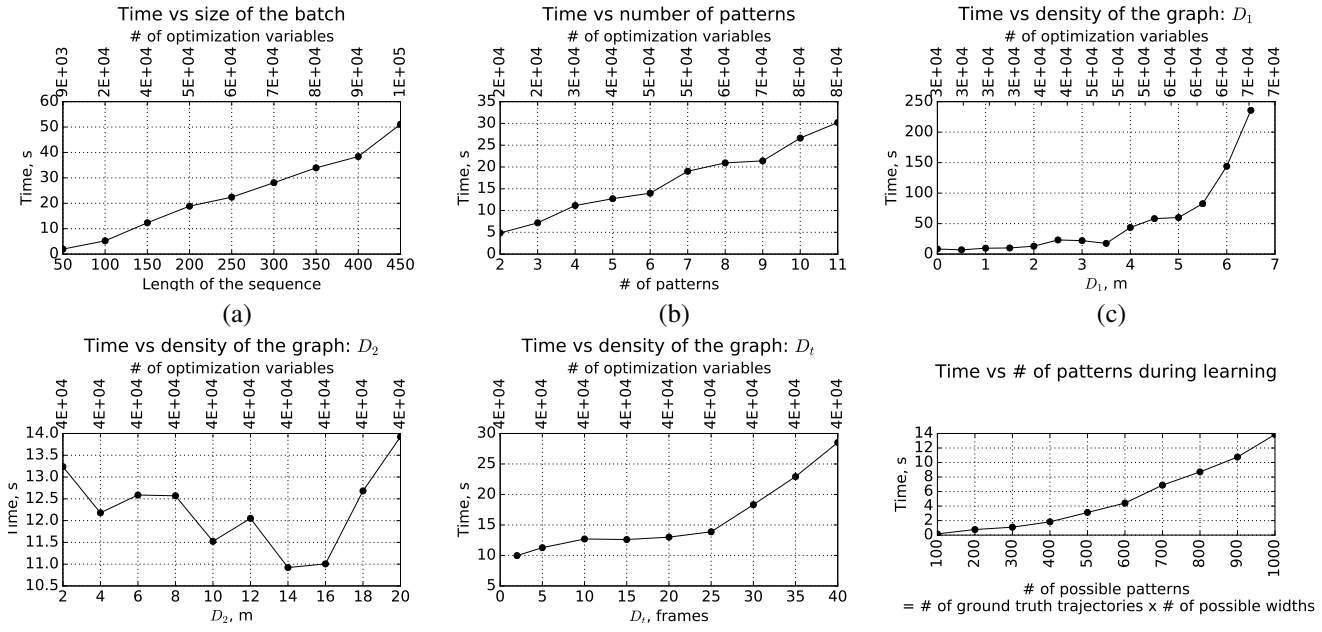


Figure 4. The running time and the number of variables of the optimization for tracking are approximately:

- linear with respect to the number of frames in the batch **(a)**,
- linear with respect to the number of patterns **(b)**,
- superlinear with respect to the maximum distance at which we join the detections in the neighbouring frames D_1, as it directly affects the density of the tracking graph **(c)**,
- almost independent from the maximum distance in space D_2 and it time D_t at which we join the endings and beginning of the input trajectories D_2, as it has almost no effect on the density of the tracking graph **(d), (e)**;

The running time and the number of variables of the optimization for learning patterns grows quadratically with the number of input trajectories, as each of them is both a trajectory that needs to be assigned to a pattern, and a possible centerline of a pattern **(f)**.

As shown by Tab. 10, for not too crowded datasets without large number of patterns our approach is able to process a minute of input frames under a minute. Pattern fitting scales quadratically with the number of given ground-truth trajectories and runs in less than 10 minutes for all datasets except **Station**. All results above were computed on datasets **ETH**, **Hotel**, **Town**, **Station**, since they shared same experimental protocol. As mentioned in Section 6.3 of the paper, for **Duke** dataset we ran evaluation for the whole length of sequence in the batch mode. For the sake of completeness, we also measured the running time. We processed around 300k * 8 frames in a total of 7853s, ranging from 654s to 1820s per sequence. This was achieved thanks to two reasons. Firstly, since the input tracks are already good, optimal value of hyperparameter D_1 was found to be 0, which enables our approach to merge or split trajectories, but not to intertwine them by splitting and then merging differently (See Sec. 3.2 of the paper for details). This further reduced the density of the graph. To speed up the approach, we added edges between trajectories not every frame, but every 0.5s, since identity switches are unlikely to happen more often.

References

[1] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua. Multiple Object Tracking Using K-Shortest Paths Optimization. 33(11):1806–1819, 2011.

[2] K. Bernardin and R. Stiefelhagen. Evaluating Multiple Object Tracking Performance: the Clear Mot Metrics. *EURASIP Journal on Image and Video Processing*, 2008, 2008.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple Online and Realtime Tracking. 2016.

[4] W. Choi. Near-Online Multi-Target Tracking with Aggregated Local Flow Descriptor. 2015.

[5] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-Camera People Tracking with a Probabilistic Occupancy Map. 30(2):267–282, February 2008.

[6] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier. Urban Tracker: Multiple Object Tracking in Urban Mixed Traffic. 2014.

[7] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele. A Multi-Cut Formulation for Joint Segmentation and Tracking of Multiple Objects. *arXiv preprint arXiv:1607.06317*, 2016.

[8] C. Kim, F. Li, A. Ciptadi, and J. Rehg. Multiple Hypothesis Tracking Revisited. 2015.

[9] L. Leal-taixe, A. Milan, I. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a Benchmark for Multi-Target Tracking. 2015.

[10] A. Milan, L. Leal-taixe, I. Reid, S. Roth, and K. Schindler. Mot16: A Benchmark for Multi-Object Tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[11] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler. Online Multi-Target Tracking using Recurrent Neural Networks. 2017.

[12] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking. 2009.

[13] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. *arXiv preprint arXiv:1609.01775*, 2016.

[14] R. Sanchez-matilla, F. Poiesi, and A. Cavallaro. Online Multi-Target Tracking with Strong and Weak Detections. 2016.

[15] R. systems s.r.o. Data from Sky User Guide, March 2017.

[16] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Subgraph Decomposition for Multi-Target Tracking. pages 5033–5041, 2015.

[17] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Multi-Person Tracking by Multicut and Deep Matching. 2016.

[18] B. Wang, G. Wang, K. L. Chan, and L. Wang. Tracklet Association by Online Target-Specific Metric Learning and Coherent Dynamics Estimation. 2016.

[19] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, K. L. Chan, and G. Wang. Joint Learning of Convolutional Neural Networks and Temporally Constrained Metrics for Tracklet Association. 2016.

[20] Y. Xiang, A. Alahi, and S. Savarese. Learning to Track: Online Multi-Object Tracking by Decision Making. 2015.

[21] M. Yang and Y. Jia. Temporal Dynamic Appearance Modeling for Online Multi-Person Tracking. 2016.

[22] J. H. Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon. Online Multi-Object Tracking via Structural Constraint Event Aggregation. 2016.

[23] B. Zhou, X. Wang, and X. Tang. Understanding Collective Crowd Behaviors: Learning a Mixture Model of Dynamic Pedestrian-Agents. 2012.