TRACKING HUNDREDS OF PEOPLE IN DENSELY CROWDED SCENES WITH PARTICLE FILTERING SUPERVISING DEEP CONVOLUTIONAL NEURAL NETWORKS

Gianni Franchi[1] *, Emanuel Aldea*[1]

[1]Université Paris-Saclay
SATIE

Séverine Dubuisson[2]*, Isabelle Bloch*[3]

[2]Aix Marseille University, CNRS, LIS
[3]LTCI, Télécom Paris, Institut Polytechnique de Paris

ABSTRACT

Tracking an entire high-density crowd composed of more than five hundred individuals is a difficult task that has not yet been accomplished. In this article, we propose to track pedestrians using a model composed of a Particle Filter (PF) and three Deep Convolutional Neural Networks (DCNN). The first network is a detector that learns to localize the persons. The second one is a pretrained network that estimates the optical flow, and the last one corrects the flow. Our contribution resides in the way we train this last network by PF supervision, and in Markov Random Field linking the different tracks.

Index Terms— Computer Vision, Crowd tracking, Self supervised learning, Deep learning

1. INTRODUCTION

Crowd tracking is a crucial bottleneck for studying the movements within crowds raising risks for public safety, and it can help analysts understand crowd dynamics. In this article, we propose to track persons in a crowd using a Particle Filter (PF) [1]. Many related articles aim to track people in crowds [2, 3, 4, 5, 6], but often the considered pedestrian systems are not dense, or they are composed of relatively few people. In our work, we track a dense crowd composed of at least 500 people in each frame, which is a novelty and a significant challenge. Tracking in these conditions is much harder because people are occluded very frequently; moreover, the compactness and shared goal of the people in the field of view makes it unfeasible to rely on distinguishing people in groups.

In order to track crowds in extremely dense circumstances we propose to use the PF, constrained with a Markov Random Field model. Moreover, to predict the position of the people in the following frame, we make use of the optical flow [7, 8, 9] of the sequence. Since the flow estimation is often inaccurate due to frequent occlusions, we correct it using a Deep Convolutional Neural Network (DCNN). However, since the construction of a training database for this DCNN is intractable, we propose a training procedure controlled by the PF. This approach is original because it allows training a DCNN in the absence of a database. Since we do not have the ground truth flow value corresponding to the pedestrian displacement, we use the results of the PF during the initial transition (when it is the most reliable). More specifically, the DCNN is initially supervised by the PF, then subsequently the PF stops training once the estimated results are considered no longer reliable. As a result, the algorithm learns to track, based on its decisions.

The contribution of this work is threefold: (1) a novel approach coupling a PF to an innovative MRF; (2) a new framework with the PF monitoring three DCNNs and supervising the training of one DCNN; (3) state of the art results on a dataset composed of more than 500 targets being tracked simultaneously.

2. MULTI-OBJECT PARTICLE FILTERING SUPERVISING DCNN

In this paper, we propose to track pedestrians in a dense crowd that are annotated in the first frame (at time $t = 0$) (i.e. we ignore potential new appearing instances during the sequence). A video is considered as a set of T frames $\{I^t, t = 0...T-1\}$, where each I^t is a 2D image. The objects in the first frame are called seeds (here person heads, to be tracked). At instant t, the object $\mathbf{o}_i^t$ ($i = 1...N_t$, where N_t denotes the number of objects in frame t) is described by the 2D coordinates $\mathbf{x}_i^t$ of its center, its width w and height h (supposed to be constant here, hence an object is simply considered as a 2D vector). We note $\{\hat{\mathbf{o}}_i^t, i = 1...\hat{N}_t\}$ the set of detections provided by a DCNN (RetinaNet [10]), and $\{\mathbf{o}_i^{\Diamond t}, i = 1...N_t^{\Diamond}\}$ the output of the PF. Note that N_t, $\hat{N}_t$ and $N_t^{\Diamond}$ are not necessarily equal. The output of the PF consists of tracks, where the track i is $\mathbf{t}_i = \{\mathbf{o}_i^{\Diamond 0}, \ldots, \mathbf{o}_i^{\Diamond T-1}\}$.

2.1. Method

Our tracking task is novel and challenging computationally as our tracker is expected to manage from five hundred to one thousand pedestrians. Hence, to perform the tracking we use

This work was partly funded by ANR grant ANR-15-CE39-0005 and by QNRF grant NPRP-09-768-1-114.

three DCNNs monitored by the PF. The PF acts as a global estimation process that relies on the information provided by the three DCNNs as illustrated in Figure 1. The first DCNN, called **DCNN detection**, is the object detection DCNN proposed in [10]. It takes an image as input and outputs a set of detections. We use the RetinaNet algorithm and hyperparameters in [10] that perform well, with the Resnet 50 architecture [11]. The second DCNN, called **DCNN O. Flow**, is Deepflow [9], which takes as input two images and outputs the optical flow. This second DCNN was pretrained on the synthetic MPI-Sintel dataset [12]. We call F^{t+1} the optical flow provided as ouptut by this DCNN with input I^t and I^{t+1}. Let us introduce below the third and last network.

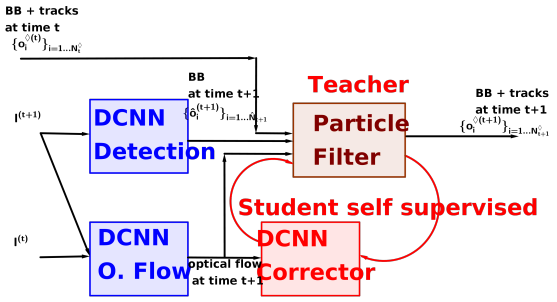

Fig. 1: The DCNNs used for the tracking.

DCNN Corrector: training supervised by the PF. Finding the exact displacement of a person in a crowd is a very complex task, due mainly to strong occlusions and visual homogeneity. In order to estimate pedestrian displacement, we rely on the **DCNN O. Flow** which provides the optical flow between images at t and $t+1$.

The prediction of the position derived from the optical flow would be the following: $\tilde{\mathbf{x}}_i^{t+1} = \mathbf{x}_i^{\Diamond t} + \mathbf{f}_i^{t+1}$, where $\mathbf{f}_i^{t+1}$ denotes the 2D displacement (flow) F^{t+1} centered on object i at t. Note that if $t = 0$ we consider that $\mathbf{x}_i^{\Diamond 0} = \mathbf{x}_i^0$. We use the notation $\tilde{\mathbf{x}}_i^{t+1}$ instead of $\mathbf{x}_i^{t+1}$ since it is a prediction of $\mathbf{x}_i^{t+1}$.

However, the occlusions among the people, the contradictory movements, the size of the objects in the field of view (FOV), and the prediction errors of **DCNN O. Flow** prevent this DCNN from estimating well enough a pedestrian position at time $t+1$ based on the position at time t. Hence we propose to use $\mathbf{f}^*$ (with $\mathbf{f}^{*0} = \mathbf{f}^0$) and, for $t > 0$, we consider the following two cases:

$$\mathbf{f}_i^{*t+1} = \begin{cases} \mu \mathbf{f}_i^{t+1} + (1-\mu)\mathbf{f}_i^t & \text{if } \mathbf{f}_i^{t+1} < \lambda \mathbf{f}_i^t \\ \text{sign}(\mathbf{f}_i^{t+1}) \min(|\mathbf{f}_i^{t+1}|, |\mathbf{f}_i^t|) & \text{otherwise} \end{cases}$$

where the inequality test, the absolute value, and the minimum are taken component wise. We have chosen $\lambda = 10$ and $\mu = 0.8$, leading to good results empirically (λ is linked to the diameter of a pedestrian head in the data-set, that is in the range of 15 pixels). The underlying idea is that if the flow changes too much between t and $t+1$, then the person being tracked might have been occluded and we should consider the flow field of another close people instead. For example, this might happen for a person moving slower than its immediate neighbors: we then consider the smallest flow computed in this neighborhood so that the prediction remains close to its previous position. Then, we predict the position of each object of frame t, by:

$$\tilde{\mathbf{x}}_i^{t+1} = \mathbf{x}_i^{\Diamond t} + \mathbf{f}_i^{*t+1} \quad (1)$$

However, this idea is not sufficient to handle all occlusion cases within crowds. A possible solution would be to consider a DCNN that takes as input a crop of the optical flow around the person i and outputs the movement of this person between times t and $t+1$. The difficulty is that this requires using a training set that would be prohibitively costly to annotate. Thus, we propose to overcome this by relying on the PF to create the dataset, and then by using this dataset to learn the parameters of the neural network. This network is then used to predict the movement of people. Since **DCNN Corrector** learns to optimize its parameters, we denote this step as unsupervised. More specifically, between times $t = 0$ and $t = T_{\text{data}}$, at each transition step, we apply the PF as described in the next paragraph, where $T_{\text{data}} < T$ represents the time where the training of **DCNN Corrector** stops and its testing starts. Thus, at time $t \in [0, T_{\text{data}}]$ for every person i we have two vectors $\mathbf{x}_i^{\Diamond t+1}$ and $\mathbf{x}_i^{\Diamond t}$ resulting from the PF at $t+1$ and t, respectively. We can then calculate: $\mathbf{f}_i^{\dagger t+1} = \mathbf{x}_i^{\Diamond t+1} - \mathbf{x}_i^{\Diamond t}$

Hence we can build a training dataset by collecting in the $2w \times 2h$-size region surrounding each object $\mathbf{x}_i^{\Diamond t}$ both the flow values from F^{t+1} and vector $\mathbf{f}_i^{\dagger t+1}$. Here $2w$ and $2h$ were set experimentally to 22 for the Makkah dataset (see Section 3 for the datasets used in our experiments). Regions extracted from F^{t+1} are given as input of **DCNN Corrector**, while $\mathbf{f}_i^{\dagger t+1}$ is the output. The **DCNN Corrector** that we use is composed of three convolution layers (of kernel size equal to 3×3), each of them being followed by a ReLU activation function and a max pooling layer. While the first convolution has 5 feature maps, the second and third ones have 10 and 20 feature maps, respectively. Then, the third max pooling layer is followed by drop out, the result is reshaped as a vector, and two fully connected layers are applied to it. The first fully connected layer (composed of 50 neurons) is followed by a ReLU and a drop out layer. All the drop out layers have a probability to keep the element equal to 0.7.

In order to train this small network for each time $t \in [0, T_{\text{data}}]$, we have collected data and trained the network with a fixed learning rate equal to 0.0001 for 5 epochs. When $t > T_{\text{data}}$ we do not train the **DCNN Corrector** anymore and just test it on the new crops to provide corrected flow to the PF estimates. Hence, $\mathbf{f}^{\Diamond t+1}$, the final displacement that we

want to estimate, is given by:

$$\mathbf{f}^{\Diamond t+1} = \begin{cases} \mathbf{f}^{*t+1} \text{ if } t \leq T_{\text{data}} \\ \mathbf{f}^{\dagger t+1} \text{ otherwise.} \end{cases} \quad (2)$$

2.2. PF for multiple interacting object tracking.

Let us first consider a single object, thus a single target to track. Let O_i^t be the random variable of the object i we are tracking at time t, $\mathbf{o}_i^{\Diamond t}$ the state at time t composed of the two coordinates of the corresponding bounding box, $\mathbf{h}_i^t$ the observation of object i in frame t corresponding to a histogram computed from the crop around the object, and H_i^t the random variable associated with it. We call T_i the random variable of track i ($i \in [1, N_0]$). Our goal is to find the best $\mathbf{t}_i = \{\mathbf{o}_i^{\Diamond 0}, \dots \mathbf{o}_i^{\Diamond t+1}\}$ by estimating $\mathcal{P}(T_i = \mathbf{t}_i \mid H_i^0 = \mathbf{h}_i^0, \dots H_i^t = \mathbf{h}_i^{t+1})$. For that, we use the filtering distribution $\mathcal{P}(O_i^{t+1} = \mathbf{o}_i^{\Diamond t+1} \mid H_i^0 = \mathbf{h}_i^0, \dots H_i^{t+1} = \mathbf{h}_i^{t+1})$ and the transition state distribution $\mathcal{P}(O_i^{t+1} = \mathbf{o}_i^{\Diamond t+1} \mid O_i^t = \mathbf{o}_i^{\Diamond t})$. Since the filtering distribution is hard to obtain, the PF framework approximates this distribution using weighted samples [1] $\{\mathbf{o}_{i,k}^{\Diamond t+1}, \tilde{w}_{i,k}^{\dagger t+1}\}_{k=0}^{NP}$ with $\tilde{w}_{i,k}^{\dagger t+1} \sim \mathcal{P}(O_i^{t+1} = \mathbf{o}_i^{\Diamond t+1} \mid H_i^0 = \mathbf{h}_i^0, \dots H_i^{t+1} = \mathbf{h}_i^{t+1})$, with NP the number of particles, and where each particle $\mathbf{o}_{i,k}^{\Diamond t+1}$ is a possible realization of the state. The particles are propagated using a proposal function $Q(O_i^{t+1} = \mathbf{o}_{i,k}^{\Diamond t+1} \mid O_i^0 = \mathbf{o}_{i,k}^{\Diamond 0}, \dots, O_i^t = \mathbf{o}_{i,k}^{\Diamond t}, H_i^0 = \mathbf{h}_i^0, \dots, H_i^{t+1} = \mathbf{h}_i^{t+1})$, and, by abuse of notation, the weights at time $t+1$ are given by: $\tilde{w}_{i,k}^{\dagger t+1} = \tilde{w}_{i,k}^{\dagger t} \times \frac{\mathcal{P}(H_i^{t+1} \mid O_i^{t+1}) \mathcal{P}(O_i^{t+1} \mid O_i^t)}{Q(O_i^{t+1} \mid O_i^0, \dots, O_i^t, H_i^0, \dots H_i^{t+1})}$.

Up to now, we have considered that all tracks are independent. We will now add spatial constraints into the PF framework to extend it to a multiple interacting object tracking. In contrast to [13, 14] we consider a random field that accounts for the fact that people can walk and interact together. Hence our Markov Field depends on the previous temporal configuration.

Let us consider that the tracks lie on a spatial graph $G^{t+1} = \left(\{O^t \cup O^{t+1}\}, E^{t+1}\right)$, with $\{O^t \cup O^{t+1}\}$ the set of objects in frames $t+1$ and t, and E^{t+1} the set of edges that link the objects that have spatial relations. More formally we consider, as in the traditional PF framework, that there is a dependence between O_i^{t+1} and O_i^t for all objects i, but we also consider that O_i^{t+1} depends on all objects O_j^{t+1} in the neighborhood of object i. In addition to this classic form of dependence, we add the possibility for object O_i^{t+1} to depend on O_j^t, the object j at time t that shares the same neighborhood condition. Hence, the dynamic state distribution is a Markov random field of the following form: $\mathcal{P}(O_0^{t+1}, \dots, O_{N_t^\Diamond}^{t+1} \mid O_0^t, \dots, O_{N_t^\Diamond}^t) \propto \prod_{i=1}^{N_t} \mathcal{P}(O_i^{t+1} \mid O_i^t) \prod_{(i,j) \in E^{t+1}} \exp(-\psi(O_i^{t+1}, O_j^{t+1}, O_i^t, O_j^t))$ with ψ the interaction potential of the field.

To build the Markov field we use a weight $\tilde{w}_{\text{spatial},i,k}^{\dagger t+1}$ that represents the spatial distance. Hence the weights of the particles are given by:

$$\tilde{w}_{i,k}^{\dagger t+1} = \frac{\mathcal{P}\left(H_i^{t+1} \mid O_i^{t+1}\right) \mathcal{P}\left(O_i^{t+1} \mid O_i^t\right)}{Q\left(O_i^{t+1} \mid O_i^0, \dots, O_i^t, H_i^0, \dots H_i^{t+1}\right)} \times \tilde{w}_{i,k}^{\dagger t} \tilde{w}_{\text{spatial },i,k}^{\dagger t+1}.$$

where $\tilde{w}_{\text{spatial },i,k}^{\dagger t+1} = \exp(\sum_{(i,j) \in E^{t+1}} -\lambda_{\text{spatial}}(D(\mathbf{o}_i^{\Diamond t}, \mathbf{o}_j^{\Diamond t})) - D(\mathbf{o}_{i,k}^{\Diamond t+1}, \mathbf{o}_j^{\Diamond t+1})))$, λ_{spatial} is a coefficient used to balance the spatial constraint, and $D(\mathbf{o}_i^{\Diamond t}, \mathbf{o}_j^{\Diamond t})$ is the Euclidean distance between the centers of objects i and j at time t. We set the threshold $\lambda_{spatial}$ to 100. Particle k for object i will have a high weight if this particle represents a coherent movement according to the crowd, that means that its distance to its neighbor on the next time must not change. Finally, the weights $\tilde{w}_{\text{spatial },i,k}^{\dagger t+1}$ are normalized such that their sum equals 1. We conclude this section with the full overview of our tracking framework below:

We calculate the position of the tracks in frame $t+1$ based on the tracks at time t, and on the corrected optical flow $\mathbf{f}^{\Diamond t+1}$ defined above.

1. Objects position prediction at $t+1$ using seeds as the estimations from frame t, and the corrected flow. For each object: $\tilde{\mathbf{x}}_i^{t+1} = \mathbf{x}_i^{\Diamond t} + \mathbf{f}^{\Diamond t+1}$ (Eq. 2).
2. From each position $\tilde{\mathbf{x}}_i^{t+1}$, $i = 1 \dots N_{t+1}^\Diamond$, propagate $NP = 150$ (for the Makkah data set) particles using a Gaussian law centered on this position and of bi-dimensional variance $\mathbf{f}^{\Diamond t+1}$. We can have $N_{t+1}^\Diamond \neq N_0$ since some objects can leave the FOV of the camera.
3. Extract a 25-bin histogram from a $2w \times 2h$ region surrounding all particles at $t+1$ and objects at t, yielding $\{\mathbf{h}_i^t, i \in [1, N_t^\Diamond]\}$ for objects at t and $\{\mathbf{h}_{i,k}^{t+1}, (i,k) \in [1, N_{t+1}^\Diamond] \times [1, NP]\}$ for particles at $t+1$.
4. Compute the weight of each particle k as $w_{i,k}^{t+1} = \exp(-\lambda_{\text{var}} \|\mathbf{h}_{i,k}^{t+1} - \mathbf{h}_i^t\|^2)$. Weights are then normalized to give $\tilde{w}_{i,k}^{t+1} = (w_{i,k}^{t+1} / \sum_{k=1}^{NP} w_{i,k}^{t+1}) \times \tilde{w}_{\text{spatial },i,k}^{\dagger t+1}$.

   The position vector of the object i at $t+1$ is given by: $\mathbf{x}_i^{*t+1} = \sum_{k=1}^{NP} \tilde{w}_{i,k}^{t+1} \mathbf{x}_{i,k}^{t+1}$, with $\mathbf{x}_{i,k}^{t+1}$ the position vector of the kth particle of object i. Then $S^{*t+1} = \{\mathbf{o}_i^{*t+1}, i \in [1, N_{t+1}^\Diamond]\}$ is the corresponding set of objects.
5. Correct positions in S^{*t+1} using the detections provided by the DCNN detector. For each object in S^{*t+1}, if there is an overlap with at least one object of $\hat{S}^{t+1} = \{\hat{\mathbf{o}}_i^{t+1}, i \in [1, \hat{N}_{t+1}]\}$, replace it by the best overlapping in $\hat{S}^{t+1}$. The resulting set is $S^{\dagger t+1} = \{\mathbf{o}_i^{\dagger t+1}, i \in [1, N_{t+1}^\Diamond]\}$.
6. For all the objects of $S^{\dagger t+1}$ that have been corrected during the previous step, apply the correction shifting that maps $\mathbf{x}_i^{*t+1}$ to $\mathbf{x}_i^{\dagger t+1}$, to all particles of object i. From now on, we denote by $\mathbf{x}_i^{\dagger t+1}$ the new position of the object i even for those whose position did not change (that were not corrected during the previous step).
7. For the particles that have been shifted, we extract the 25-bin histogram from their surrounding $w \times h$ size area and compute the weights: $w_{i,k}^{\dagger t+1} = \exp(-\lambda_{\text{var}} \|\mathbf{h}_{i,k}^{\dagger t+1} - \mathbf{h}_i^t\|^2)$. As in the previous step, $w_{i,k}^{\dagger t+1}$ stands for all the weights, even for particles that have not been shifted (and then whose weight was not changed).
8. Normalize the weights $w_{i,k}^{\dagger t+1}$ to get $\tilde{w}_{i,k}^{\dagger t+1}$, and estimate the position of the objects by:

$$\mathbf{x}_i^{\Diamond t+1} = \sum_{k=1}^{NP} \tilde{w}_{i,k}^{\dagger t+1} \mathbf{x}_{i,k}^{\dagger t+1} \quad (3)$$

   These last coordinates represent the estimation provided by the PF.
9. Resample the particles using multinomial resampling.

3. EXPERIMENTS AND RESULTS

To validate our tracking algorithm we performed two sets of experiments. In the first experiment we used sequence 6 of [15], which is to the best of our knowledge the most extensive annotated dense crowd sequence freely available, comprising 333 frames with an average of 440 people per frame. On this dataset, we achieve the same results as the NMC algorithm proposed in [15], with the tracking accuracy threshold set to 15 on 133 frames, in order to account for the imprecision in the annotation. The last 200 frames were used to train the **DCNN detection**. We achieve a tracking accuracy [15] of 98%, despite the fact that the data available for training is imprecise and incomplete, and despite the poor performance of the optical flow provided by **DCNN O. Flow**.

For the second experiment, we have used two datasets at different times of the Makkah pilgrimage. First, for **DCNN detection**, we use a training set composed of 9 images with a total of 8028 instances, and a validation set composed of 2 images with 1700 instances. There are no tracking annotations on these images, the only annotations present are bounding boxes around the head of each pedestrian. Secondly, we have a tracking dataset composed of 620 pedestrian trajectories generated across 50 frames, and based on seeds sampled mostly in the high-density area of the Makkah crowd during the Hajj period (Figure 2). Our annotations span only the area of the field of view in which the human annotation may still be performed without ambiguity. We have compared our results[1] with those of two algorithms. The first algorithm, denoted as `flow`, only uses optical flow to track. We have also compared our results with those of `NMC`, the algorithm described in [15]. This method presents state-of-the-art results in crowd tracking and has initial conditions close to ours. `PF` and `PFMRF` denote the algorithm described in Section 2.2, without and with the the proposed MRF, respectively. Additionally, the use of the **DCNN Corrector** is specified by the term `DCNN Cor` and the corresponding tuning parameters. In order to evaluate the quality of a tracking algorithm we use three measures, MOTA [16] and MOTP [16], widely used in tracking for medium dense multi object tracking datasets, and tracking accuracy (TA) [15], well adapted for crowd datasets. We set the IoU threshold for MOTA to 0.5, which is standard for object detection, and the threshold for the tracking accuracy to 15, as in [15].

Comparative results are provided in Table 1: we note that the `PFMRF` variants provide better results than `flow` and `NMC`, in terms of both object detection and tracking, for the three metrics. The value $T_{\text{data}} = 5$ is a good compromise since PF might provide less reliable data for a higher value. Training for more than 5 epochs for each time does not increase the tracking quality because of the relatively limited amount of data. Hence the results reported in Table 1 underline that the **DCNN Corrector** improves the performance of our algorithm and it is actually needed to track the crowd effectively.

[1] `hebergement.u-psud.fr/emi/MOHICANS/output.mp4`

	MOTP	MOTA	TA
flow	22%	16%	39%
NMC	54%	61%	78%
PF	59%	66%	73%
PFMRF	60%	70%	78%
PFMRF DCNN Cor 2 epochs $T_{\text{data}} = 5$	60%	70%	78%
PFMRF DCNN Cor 5 epochs $T_{\text{data}} = 5$	**64%**	**75%**	**84%**
PFMRF DCNN Cor 10 epochs $T_{\text{data}} = 5$	62%	72%	82%
PFMRF DCNN Cor 5 epochs $T_{\text{data}} = 10$	62%	73%	82%
PFMRF DCNN Cor 5 epochs $T_{\text{data}} = 50$	63%	73%	83 %

Table 1: Performance of the tracking algorithms on the Makkah dataset.

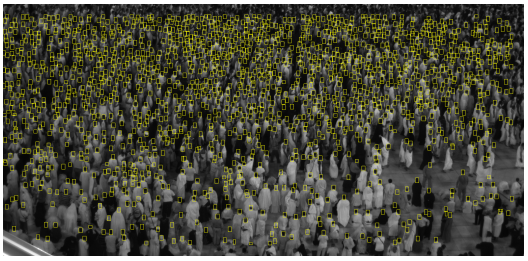

Fig. 2: Example of detection results of the RetinaNet, with ResNet-50 as architecture. Multiple factors such as the density, occlusion level and homogeneity of the scene impede significantly the tracking process.

4. CONCLUSION

In this paper, we propose to integrate unsupervised/self-supervised learning into a crowded scene tracking algorithm via pseudo ground truth estimations provided by a Particle Filter. This original contribution circumvents the requirement of a costly annotation for a deep architecture aimed at correcting a prior optical flow estimation in the context of crowd dynamics. Our results show that the Markov Random Field that we propose in order to model spatial constraints among the targets and the DCNN supervised by the PF are the key ingredients for tracking simultaneously a significant number of pedestrians in an homogeneous high-density crowd.

In future work, we intend to extend our method in order to propose a global consistency DCNN that will be able to learn the crowd dynamics in a self-supervised manner based on more complex spatial interactions among the pedestrians.

5. REFERENCES

[1] N. J. Gordon, D. J. Salmond, and A. FM Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE proceedings F (radar and signal processing)*. IET, 1993, vol. 140, pp. 107–113.

[2] P. Chu and H. Ling, "FAMNet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking," *arXiv preprint arXiv:1904.04989*, 2019.

[3] A. Dehghan, S. Modiri Assari, and M. Shah, "Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking," in *International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4091–4099.

[4] A. Dehghan, Y. Tian, P. Torr, and M. Shah, "Target identity-aware network flow for online multiple target tracking," in *International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1146–1154.

[5] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn, "Fusion of head and full-body detectors for multi-object tracking," in *International Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1428–1437.

[6] G. Wang, Y. Wang, H. Zhang, R. Gu, and J.-N. Hwang, "Exploit the connectivity: Multi-object tracking with TrackletNet," *arXiv preprint arXiv:1811.07258*, 2018.

[7] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. Van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," *arXiv preprint arXiv:1504.06852*, 2015.

[8] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *International Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2462–2470.

[9] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *International Conference on Computer Vision*, 2013, pp. 1385–1392.

[10] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *International Conference on Computer Vision*, 2017, pp. 2980–2988.

[11] K. He, X. Zhang, S. Ren, and J Sun, "Deep residual learning for image recognition," in *International Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[12] D. J Butler, J. Wulff, G. B Stanley, and M. J Black, "A naturalistic open source movie for optical flow evaluation," in *European Conference on Computer Vision*. Springer, 2012, pp. 611–625.

[13] O. Hirose, S. Kawaguchi, T. Tokunaga, Y. Toyoshima, T. Teramoto, S. Kuge, T. Ishihara, Y. Iino, and R. Yoshida, "SPF-CellTracker: Tracking multiple cells with strongly-correlated moves using a spatial particle filter," *ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 6, pp. 1822–1831, 2018.

[14] Z. Khan, T. Balch, and F. Dellaert, "An MCMC-based particle filter for tracking multiple interacting targets," in *European Conference on Computer Vision*. Springer, 2004, pp. 279–290.

[15] H. Idrees, N. Warner, and M. Shah, "Tracking in dense crowds using prominence and neighborhood motion concurrence," *Image and Vision Computing*, vol. 32, no. 1, pp. 14–26, 2014.

[16] A. Milan, K. Schindler, and S. Roth, "Challenges of ground truth evaluation of multi-target tracking," in *International Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 735–742.