This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

Tracking-by-Counting: Using Network Flows on Crowd Density Maps for Tracking Multiple Targets

Weihong Ren, Xinchao Wang, Jiandong Tian, Yandong Tang, and Antoni B. Chan

***Abstract*—State-of-the-art multi-object tracking (MOT) methods follow the tracking-by-detection paradigm, where object trajectories are obtained by associating per-frame outputs of object detectors. In crowded scenes, however, detectors often fail to obtain accurate detections due to heavy occlusions and high crowd density. In this paper, we propose a new MOT paradigm, tracking-by-counting, tailored for crowded scenes. Using crowd density maps, we jointly model detection, counting, and tracking of multiple targets as a network flow program, which simultaneously finds the global optimal detections and trajectories of multiple targets over the whole video. This is in contrast to prior MOT methods that either ignore the crowd density and thus are prone to errors in crowded scenes, or rely on a suboptimal two-step process using heuristic density-aware point-tracks for matching targets. Our approach yields promising results on public benchmarks of various domains including people tracking, cell tracking, and fish tracking.**

***Index Terms*—People tracking, crowd density map, multiple people tracking, flow tracking.**

I. Introduction

Multiple-object tracking (MOT) is crucial for many computer vision tasks such as video analytics. Despite many years of effort, MOT remains a very challenging task due to factors like occlusions between the targets. Recent MOT approaches have been focused on the *tracking-by-detection* paradigm, whose goal is to first detect the targets in each frame and then associate them into full trajectories. Such approaches have been successful in scenarios with low-density of targets. In crowded scenes, however, they often fail to extract the correct trajectories due to the detection failures caused by occlusions and the high densities of targets, even with state-of-the-art detectors trained on large-scale datasets [1–6].

We propose in this paper a novel MOT approach, explicitly designed for handling crowded scenes. We incorporate object counting, a reliable and informative clue in crowded scenarios, into our modeling, and solve the multiple-object *detection and tracking* simultaneously over the whole video sequence. Specifically, for each frame, we estimate an object density map, based on which a spatio-temporal sliding window is applied for estimating object counts. We then construct a spatiotemporal graph over the whole video sequence, where each node denotes a candidate detection at a pixel location, each edge denotes a possible association, and a sum over a set of nodes denotes a count in the corresponding sliding window. Using the constructed graph, we model the joint detection-tracking-counting problem as a network flow program. The network-flow constraints and the object-count constraint reinforce each other and together benefit the tracking. The high quality solutions of the network flow program, or the complete object trajectories, are obtained used off-the-shelf solvers.

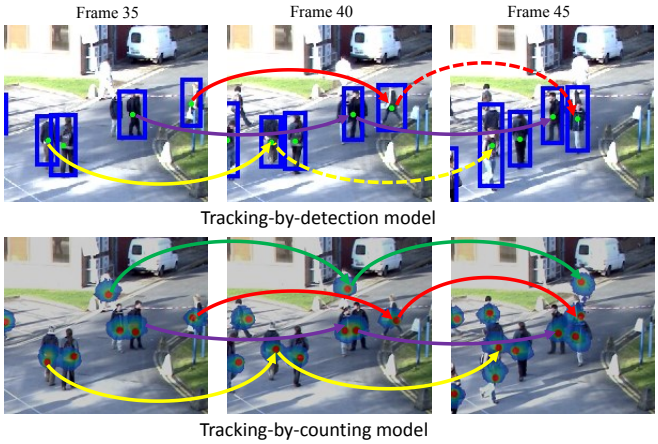


Fig. 1: Tracking results of tracking-by-detection and tracking-by-counting models on *PETS2009*. For clear visualization, we only show part of the trajectories in the scene. Solid lines denote the correct associations while the dotted lines denote the wrong ones. The object detections used by the tracking-by-detection model are obtained by Faster-RCNN, which fails to localize some occluded people. Our tracking-by-counting model, on the other hand, estimates object densities and imposes the count constraint into the joint detection and tracking framework, resulting in better tracking results especially for occluded people. The red dots on the lower row denote the recovered detections.

We show in Fig. 1 a qualitative comparison between an advanced tracking-by-detection approach DCEM [7] and our proposed tracking-by-counting. State-of-the-art detectors like Faster-RCNN in this case miss some partially-occluded pedestrians and thus cause tracking failures. The proposed tracking-by-counting model, on the other hand, produces the accurate detections for all the objects using the crowd density maps, and finds the corresponding associations among them. Also, the tracking-by-counting model is able to localize and track objects with similar appearance with the background, like the person in white at the top, which is missed by Faster-RCNN.

Although MOT approaches using density maps have been explored by prior methods, they either rely on heuristic local

Weihong Ren and Antoni B. Chan are with Department of Computer Science, City University of Hong, Kowloon, Hong Kong (e-mail: weihonren2-c@my.cityu.edu.hk, abchan@cityu.edu.hk).

Xinchao Wang is with Department of Computer Science, Stevens Institute of Technology, New Jersey, United States (e-mail: xinchao.w@gmail.com).

Jiandong Tian and Yandong Tang are with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, 110016, China; Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang, 110016 China (e-mail: tianjd@sia.cn, ytang@sia.cn)

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

point tracks or human-initialized target locations in the first frame. Specifically, the approach of [8] first detects the head locations of all the people by encouraging the detections to be consistent with the estimated density maps, and then counts the point tracks passing through a given paired heads in different frames. If this count is larger than a threshold, a match is declared. One drawback of [8] is thus the simple data association rule, which does not use an appearance or a motion model during tracking and could therefore fail when objects with similar appearance move close to each other. The approach of [9] utilizes appearance, motion and contextual cues but relies on initialization of object locations in the first frame. Furthermore, it is not able to detect new objects coming to the scene. In contrast to [8] and [9], our approach explicitly incorporates the object-count constraints obtained using a sliding window on density map videos, and network-flow constraints that impose temporal-consistent tracks, into a joint detection-tracking-counting model over the whole video sequence. We model this tracking-by-counting approach using a network flow program and obtain the high quality solution using standard commercial solvers.

Our contribution is therefore a tracking-by-counting approach that jointly solves multi-object detection and tracking simultaneously, by explicitly incorporating counting constraints from object density maps into the framework. This is achieved by our network flow programming formulation. The proposed model is the first attempt towards bridging the gap of counting, detection and multi-object tracking. We demonstrate the power of our approach on benchmarks of various domains including people tracking, fish tracking, and cell tracking.

The remainder of this paper is organized as follows. In Section II we review previous work on MOT and crowd counting. In Section III we introduce our tracking-by-counting, and in Section IV conduct experiments on benchmark dattasets. In Section V we extend our model to handle large-scale datasets through incorporating object detection results. Finally Section VI concludes the paper.

II. Related work

In this section, we briefly review related work including tracking-by-detection MOT approaches, deep-learning MOT approaches and object counting approaches. Comprehensive reviews on MOT can be found in [10].

A. Multi-object tracking using tracking-by-detection

Early multi-object tracking rely on filtering techniques, e.g., Kalman filtering [11, 12] and particle filtering [13–15]. Although these methods can be applied to real-time tracking task, they are usually prone to problems like drifts that are difficult to recover from. Recently, tracking-by-detection has become the standard paradigm for MOT. The main idea is to split the problem into two parts: object detection and data association. Over the past few years, object detection has seen great improvement thanks to deep learning techniques [2, 3, 16], but data association remains a challenge for multi-object tracking.

Most of the tracking-by-detection methods regard data association as a global optimization problem and focus on designing various optimization algorithm, such as continuous energy minimization [17–19], Conditional Random Fields (CRFs) [7, 20–23] and min-cost network flow [24–31]. [17] formulated multi-object tracking as minimization of a continuous energy function which can incorporate appearance information, physical information and trajectory prior together. To reduce Identity Switches (IDS) generated by the continuous model, [7] further extended [17] as a discrete-continuous model which introduces a pair-wise label cost imposing penalty if two labels co-exist that should not appear simultaneously. [32] presented a pioneering work using CRFs to segment and label sequence, and explored the possibility of CRFs for multi-object tracking. Based on CRFs, [33] designed a set of unary functions that model motion and appearance for discriminating all objects, as well as a set of pairwise functions that differentiate corresponding pairs of tracklets. Recent approaches have formulated multi-object tracking as a min-cost network flow optimization problem, where the optimal flow in a connected graph of all detections both selects the best matched candidates and encodes the tracks among them. The solutions to min-cost network flow can be optimized through different algorithms, e.g., shortest paths [25, 34], integer programming [28, 30, 35], linear programming [36, 37] and dynamic programming [38, 39]. In crowded scenes, however, tracking-by-detection approaches often fail to extract the correct trajectories due to the detector failures such as missed detections. To remedy occlusions or missed detections, several works also integrate additional detection information for tracking. For example, [40] uses both joint and body detections for data association, but this method also will fail in highly crowded scenes. Other early works like [41, 42] consider people tracking in structured crowd scenes that have clear and smoonth motion patterns. [43–45] are tailored for crowd scene tracking, but they mainly focus on finding missed detections by using motion models and handcrafted features and thus can only handle medium crowd density.

B. Multi-object tracking using deep learning

Deep learning techniques have been widely used in object detection [2, 3, 16] and visual object tracking [46–48]. Recently, several multi-object tracking algorithms also have been proposed based on convolutional neural networks (CNNs) [49–54] and recurrent neural networks (RNNs) [55, 56]. By combining image formation and optical flow as a multi-modal input, [49] proposed to use a siamese CNN to estimate the likelihood that two pedestrian detections belong to the same trajectory. Unlike [49], [50] directly used siamese CNN to encode appearance cue and motion cue to construct a generalized linear assignment model for tracklet association. [53] proposed a Quadruplet Convolutional Neural Networks (Quad-CNN) which has a multi-task loss to jointly learn association metric and bounding-box regression. The target association is then performed by a minimax label propagation using the association metric and refined bounding box from the Quad-CNN. Using the merits of single object tracker, [52] initialized each detection with a CNN-based tracker which has a spatial-temporal attention mechanism (STAM) to handle the drift caused by occlusion and interaction among targets.

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

Inspired by the Bayesian filtering idea, [55] presented an RNN that can incorporate all multi-target tracking tasks including prediction, data association, state update as well as initiation and termination of targets within a unified network structure, but maintaining a LSTM for each detection costs too much, especially for crowed scenes. [56] proposed a multi-object tracking method based on RNN, which encodes appearance, motion and interactions together to compute the similarity scores between the tracked targets and the newly detected objects. Also using LSTMs, [57] developed a sophisticated model to reduce Identity Switches (IDS), but it discards many occluded detections, which leads to low tracking accuracy. Recently, [58] used U-Net [59] to address the problem of recognizing bees and their orientations in a densely packed honeybee comb, but it is difficult to apply the method to natural scenes. [60] adopts object detector for data association without specific training on tracking task, and its performance depends heavily on the object detector. One drawback of this method is that it causes too many IDS.

C. Crowd counting, detection and tracking using density maps

Unlike object detection, which focuses on individual targets, crowd counting methods aim to predict the number of object in an image without explicitly detecting or tracking the object. One effective method is to estimate a crowd density map [61], where the sum over a region in the image corresponds to the number of object in that region. Previous methods [61–63] usually regard density map estimation as a regression problem, and mainly focus on feature estimation and loss function design to make the estimation robust to scene changes. Recent methods use CNNs [64–67] and LSTM [68, 69] for density map estimation, and have achieved good performance for a wide range of scenes.

Density maps can also be used for object detection. [70] first estimated the object density maps, and then used 2D integer programming to predict object detections based on the density maps. In [8], a "density-aware" detection and tracking model was proposed to combine individual person detections with crowd density maps. [8] solves an energy minimization problem where the candidate detections with high scores in the detection score map are preferred, and where the detections consistent with the crowd density map are encouraged. However, their tracking framework does not contain appearance models for each object, and only uses simple nearest-neighbors correspondence between frames, which could fail in crowd scenes when two objects are close to each other. Unlike the existing methods, our model jointly formulates object detection, counting and tracking on density maps as a network flow program, where the object appearance, motion and spatiality are taken into account and the global flexible solution can be found.

III. Tracking-by-counting model

Our tracking-by-counting model optimizes object detection and tracking simultaneously in one framework, shown in Fig. 2. It consists of two main components: the object count constraint for detection, and the network flow constraint for data association. The workflow of our method is as follows. The density maps are first estimated from the input video. Then, sliding windows are selected over the whole video to create count constraints for object detection on the density maps. Meanwhile, a graph is built by densely sampling pixels from the density maps for data association. Finally, our tracking-by-counting model is optimized using Integer Programming. In the remainder of this section, we first introduce object detection on crowd density maps, followed by the new proposed *tracking-by-counting* model.

A. Object detection on density maps using count constraints

Density maps have been used for object detection on 2D image space [8, 70]. Here, we extend object detections to 3D density maps over a video space. We first define a set of 2D sliding windows over the density map video $\boldsymbol{\mathcal{D}}(t)$ [70]. A 2D window is centered at a pixel in the frame, and its spatial size is set as the average target size. The 2D window moves at a fixed step vertically, horizontally, or temporally in the 3D video space. In our implementation, the stride size is (3,6), and windows with low density value (<0.005) are dropped to reduce complexity. Each sliding window at frame t is represented as a mask vector $\mathbf{w}_k^t \in \{0,1\}^N$, where N is the number of pixels in the entire video and k is the window index. The pixels within the ROI of the window are assigned value of 1, and 0 otherwise.

Using the density maps, the number of people in the sliding window $\mathbf{w}_k^t$ is estimated as

$$\hat{n}_k^t \approx \left(\mathbf{w}_k^t\right)^T \mathbf{d}, \tag{1}$$

where the vector $\mathbf{d} \in \mathbb{R}^N$ is the vectorization of all the density maps $\boldsymbol{\mathcal{D}}$ in a video. On the other hand, we encode candidate object detections in the entire video by a single N-vector $\mathbf{x} = [x_1, x_2, ..., x_N] \in \{0,1\}^N$, where x_i = 1 if pixel location i has an object centered there, and 0 otherwise. Thus, the object count in the same window $\mathbf{w}_k^t$ can also be represented as

$$n_k^t = \left(\mathbf{w}_k^t\right)^T \mathbf{x}. \tag{2}$$

The optimal object detections $\mathbf{x}$ in the video can be obtained by minimizing the counting difference between (1) and (2) over all the sliding windows, as in [70],

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{t=1}^{\mathrm{T}} \sum_{k=1}^{\mathrm{K}_t} \left| \left(\mathbf{w}_k^t\right)^T \mathbf{x} - \hat{n}_k^t \right|, \tag{3}$$

where T is the total frame number of a video, and K_t is the number of sliding windows in frame t. Note that (3) only yields object detections without any continuity constraints. In the next subsections, we show how (3) can be integrated with network flow to perform tracking.

1) Bounding box estimation: To estimate the bounding box for each detected object, we use the density map and perspective map for scale estimation, following [70]. The bounding box is found whose sum on the density map is close to 1, and also consistent with the perspective map of the scene. The perspective map is estimated by linearly interpolating the detection boxes (e.g., produced by the public detector in

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

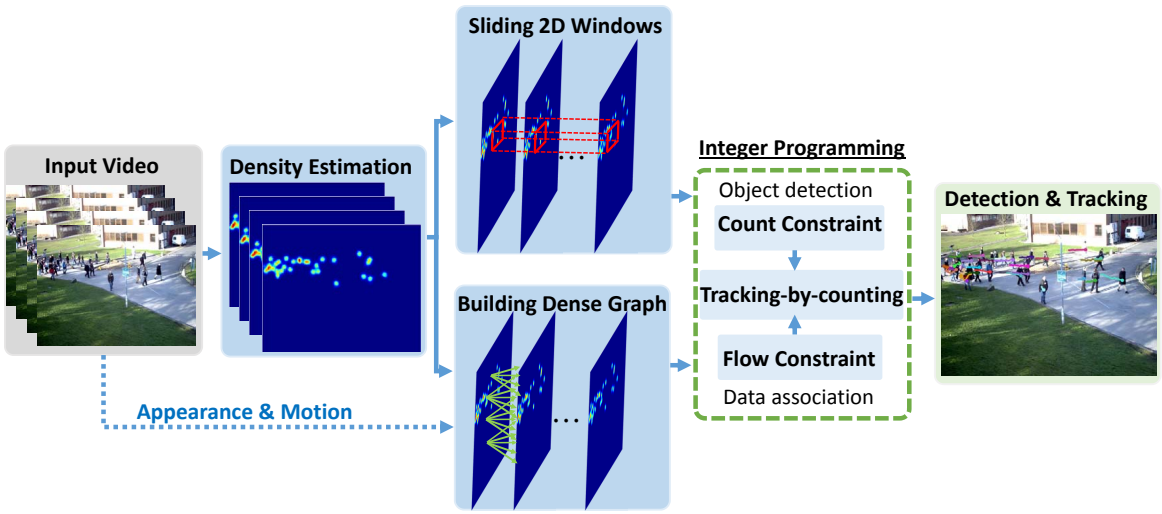


Fig. 2: The proposed tracking-by-counting model. We first estimate the density maps from the input video. Sliding 2D windows are generated over the density map video to obtain object count constraints for detection. Meanwhile we also build a dense graph with flow constraints for data association. Combining the two constraints, we model the tracking-by-counting problem as an Integer Program, whose solution simultaneously generates the detections and tracks.

MOT17) between two extremes of the scene and can reflect the scale changes of an object in a scene. For sequences taken with fixed cameras, we only need to estimate one perspective map for all the frames. However, for those with moving cameras (e.g., *MOT17*), we estimate a perspective map for each frame. Referring to [70], the bounding box for a detection in frame t is estimated by

$$\mathbf{b}^* = \arg\min_{\mathbf{b}_i} \left| \sum_{p \in \mathbf{b}_i} \boldsymbol{\mathcal{D}}(p,t) - c \right| + \lambda_b \Delta(\mathbf{b}_i, \mathbf{b}_0), \quad (4)$$

where $\Delta(\mathbf{b}_i, \mathbf{b}_0)$ is the total difference between the estimated bounding box $\mathbf{b}_i$ and the reference box $\mathbf{b}_0$, which is set using the perspective map. Parameter λ_b controls the weight of the prior from $\mathbf{b}_0$, and c is the target density value, which is between 1 and 0.8, which respectively correspond to a looser or tighter bounding box. We have also tried directly using object detectors for scale estimation, but they often fail in these crowded scenes. We solve (4) using a simple exhaustive search, within a range of plausible bounding boxes ($\pm 20\%$) consistent with the perspective map.

2) Density map estimation: We adopt the approach of [64] to estimate density maps using a CNN with 3 convolutional layers and 3 fully connected layers. Note that we estimate a high-resolution density map for tracking by using a sliding window CNN to predict the density for each pixel in the tracking image patch.

B. Tracking-by-counting: joint detection and tracking with density maps

Object detectors often fail to localize objects in crowded scenes, whereas density maps are more robust in these scenarios. By incorporating the count constraint (for object detection) with flow constraint (for tracking) on crowd density maps, the multi-object tracking problem can be cast as a *joint* optimization, which simultaneously predicts people detections and connections between them across video frames. Formally, we incorporate the data association cost and flow constraints used in flow-tracking methods with the object count constraint in (3), resulting in the *tracking-by-counting* model:

$$\begin{aligned} \min_{\mathbf{x}} & \sum_{t=1}^{\mathrm{T}} \sum_{k=1}^{\mathrm{K}_t} \left| \left(\mathbf{w}_k^t\right)^T \mathbf{x} - \hat{n}_k^t \right| + \sum_{ij \in E} c_{ij} x_{ij} + \sum_i c_{si} x_{si} + \sum_i c_{it} x_{it} \\ \text{s.t.} & \sum_{i:ij \in E} x_{ij} + x_{sj} = x_j = \sum_{i:ji \in E} x_{ji} + x_{jt} \\ & \sum_i x_{it} = \sum_i x_{si}, \quad x_i, x_{ij} \in \{0,1\}, \end{aligned} \quad (5)$$

where $\mathbf{x} = [x_1, \cdots, x_N]$. Note that here the binary indicator variables $\{x_i\}$ denote potential detections on all pixel locations densely sampled from the density maps, such that $x_i = 1$ if a location i is selected as a detection and appears in some track. $x_{ij} \in \{0,1\}$ is also a binary indicator variable, where $x_{ij} = 1$ when location i and location j are both selected as detections for the same track in consecutive frames. The set of possible connections between all the potential object detections, built using spatial proximity, is represented as E. The connection variables x_{si} and x_{it} represent the start and end of tracks respectively (s is the "source" node and t is the "sink" node), and c_{si} and c_{it} are the cost of the track start and terminate respectively. The variable c_{ij} is the edge cost for associating detections at locations i and j, which incorporates geometric location, target appearance and motion direction:

$$c_{ij} = -\alpha e^{-\lambda \|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\|_2} - \beta H(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j) - \gamma \cos(V_i, V_j), \quad (6)$$

where $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$ are the 2D coordinates of locations i and j, respectively, $H(\phi_i, \boldsymbol{\phi}_j)$ is the histogram intersection between the histograms of image patches extracted from locations i and j, and $\cos(V_i, V_j)$ is the cosine similarity between the velocities V_i and V_j, which are estimated through optical flow [71]. The tracking-by-counting model optimizes the object detection and data association at the same time. Furthermore, the flow constraint on the density maps will make the outputs of the detection term in (3) consistent between frames, which will greatly reduce missed detections. In turn, the enhanced

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

detections benefits the tracking performance. The proposed model makes the first attempt towards the joint multi-object detection, tracking, and counting, and can potentially bridge the gap between MOT and video-based object counting.

To solve the energy function in (5), we further rewrite it as a standard linear form. We introduce an auxiliary variable z_k^t, and let

$$\left|\left(\mathbf{w}_k^t\right)^T \mathbf{x} - n_k^t\right| \leq z_k^t, \tag{7}$$

where implicitly $z_k^t \geq 0$. Thus, the minimization of $\left|\left(\mathbf{w}_k^t\right)^T \mathbf{x}^d - n_k^t\right|$ can be regarded as the minimization of z_k^t. Equation 5 can then be rewritten as

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{z}} & \sum_{t=1}^{\mathrm{T}} \sum_{k=1}^{\mathrm{K}_t} z_k^t + \sum_{ij \in E} c_{ij} x_{ij} + \sum_i c_{si} x_{si} + \sum_i c_{it} x_{it} \\
\text{s.t.} & \sum_{i:ij \in E} x_{ij} + x_{sj} = x_j = \sum_{i:ji \in E} x_{ji} + x_{jt} \\
& \sum_i x_{it} = \sum_i x_{si} \\
& \left(\mathbf{w}_k^t\right)^T \mathbf{x} - n_k^t - z_k^t \leq 0 \\
& -\left(\mathbf{w}_k^t\right)^T \mathbf{x} + n_k^t - z_k^t \leq 0 \\
& x_i, x_{ij} \in \{0,1\}, 0 \leq z_k^t.
\end{aligned} \tag{8}$$

At the optimum of (8), we have $z_k^t = \left|\left(\mathbf{w}_k^t\right)^T \mathbf{x}^d - n_k^t\right|$ since z_k^t is minimized in the objective. The above formulation is a Mixed Integer Linear Programming problem (MILP) and can be directly solved through a optimization toolbox, such as CPLEX/MOSEK. Although (5) as a mixed integer linear program is NP-complete, there are approximation algorithms that can provide quality solutions with moderate computation. CPLEX uses branch-and-cut search, which works well in practice [8, 28, 30, 35, 70]. When solving (5), we drop candidate points with low density values. In all experiments CPLEX was able to find an efficient and stable solution to (8) (using tolerance gap of 0.001) without any special initialization of $\mathbf{x}$. The small tolerance gap between the primal and dual solutions indicates that a global optimum was likely found.

IV. Experiments

In this section, we evaluate our tracking-by-counting model on MOT using five small-scale datasets featuring crowded scenes, *UCSD* [72], *LHI* [73], *Fish* [70], *Cell* [74, 75] and *PETS2009* [76]. In Section V, we test on three large-scale datasets, *MOT17* [77], *MOT20*, and *DukeMTMC* [78]. As for the UCF benchmark in [9], it only has annotations for part of the people in the highly crowded scene. Besides, even the IP method [70] is not able to localize objects in that scene, since density maps cannot be accurately estimated for UCF. This is also the reason why [9] needs to be initialized with the ground truth annotations in the first frame and is not able to detect new objects coming into the scene.

A. Datasets

For the *UCSD* dataset, we choose the most crowded video clip of 200 frames (238×158) featuring 63 pedestrians. The *LHI* dataset comprises a color video of 400 frames (288×352) with 43 tracks. The *Fish* dataset has 129 frames (300×410) with 279 intersecting paths. The *Cell* dataset contains 92 frames (350×550), and features 265 tracks where cells deform frequently. For *PETS2009* dataset, we choose the S2-L2 sequence of 436 frames (576×768) with 42 tracks. Examples of the five datasets and their corresponding density maps are shown in Fig. 3.

B. Evaluation metrics

The trackers are evaluated using the CLEAR MOT [79] metrics on the 2D plane. Multiple Object Tracking Accuracy (MOTA) accounts for false positives (FP), false negatives (FN), and identity switches (IDS), while the Multiple Object Tracking Precision (MOTP) measures the average distance between the ground truth and the tracker output. A track is considered as a Mostly Tracked (MT) one if the ground truth trajectory is covered by this track for $\geqslant$80% of its time span, while a Mostly Lost (ML) track is the one with $\leqslant$20%. Track fragmentation (FM) counts the number of trajectory fragmentations, and FAF represents the number of false alarms per frame. RCLL and PRCN are the recall and precision of the detections used as the input for a tracker. Finally, IDF1 [80] means the ratio of correctly identified detections over the average number of ground-truth and predicted detections.

C. Experiment setup

We denote our tracking-by-counting model (Eq. 5) as "TBC". We set c_{si} = c_{it} = 10 in (5), and $\lambda = \beta = \gamma = 1$ in (6). To see the effect of the joint detection-tracking framework, we also consider a variant (denoted as "TBC3") that separates object detection and tracking – TBC (Eq. 5) is used as a temporally-regularized object detector for every three frames, and the detections are input into a general flow-tracking method [34], by replacing the first term in (5) with $c_i x_i$ for data association (denote as "FT"). Here, c_i denotes the cost of selecting detection i; it is set to 1, indicating high confidence for the detections from TBC3. Our TBC model is implemented using Matlab with the CPLEX toolbox on a PC (i7 3.4GHz CPU, 8 GB memory).

To estimate density maps, we adopt the approach of [64], except that we estimate a high-resolution density map for tracking by using a sliding window CNN to predict the density for each pixel. For each dataset, we only train the density estimation network on its own training set, and did not use other extra information. If a dataset (e.g., *MOT17*) has different types of sequences, we fine-tuned each sequence using the pretrained model on the whole training set. Note that the fine-tuning is only performed on the training sequences. For a test sequence, we use a model fine-tuned on a similar training scene. For the datasets *Fish* and *Cell*, we use the traditional method [67] to estimate the density maps due to the small size of the training set.

We compare our TBC with four state-of-the-art multi-object trackers, DCEM [7], GOGA [34], MHT [81] and density-aware (DA) [8]. DCEM, GOGA and MHT are typical tracking-by-detection trackers, for which the first step is to obtain a

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

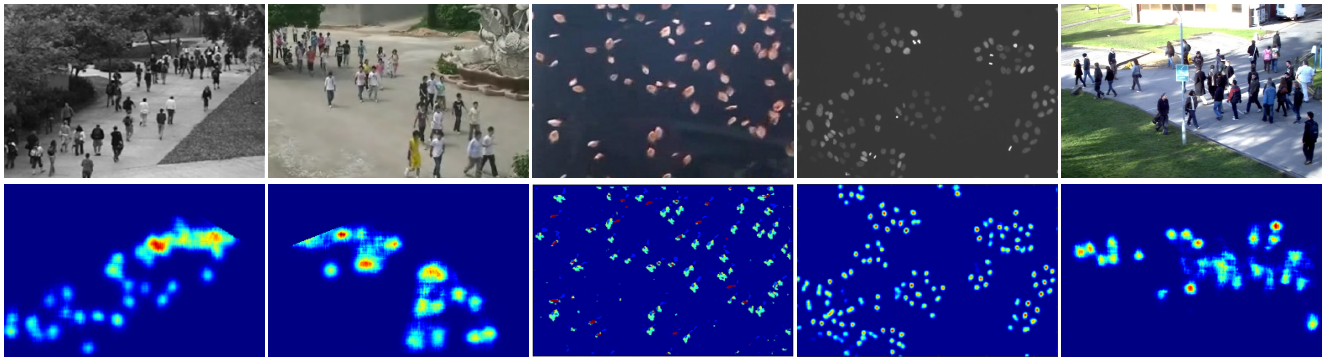


Fig. 3: Example frames from left to right on the top row: *UCSD*, *LHI*, *Fish*, *Cell* and *PETS2009*, and their corresponding crowd density maps on the bottom. *UCSD*, *Fish* and *Cell* have the higher crowd densities (average 40-80 objects per frame), and *LHI* and *PETS* have relatively smaller crowds (average 20 people per frame). Crowd levels also changes within each video (e.g., 12-34 in PETS).

set of object detections for each frame independently. In our experiments, we consider three object detectors: the retrained DPM [1], the retrained Faster-RCNN [2], and the Integer Programming method based on density map (IP) [70]. To well retrain the DPM and Faster-RCNN, we upsample the datasets to an appropriate scale for object detections. DA achieves good detection results using DPM detection score maps and density maps, but the original formulation relies on a simple association method. For fairer comparison, we use DA as a detector and apply the flow-tracking (FT) for data association. In the text, we denote combinations of tracker and detectors as "tracker-detector".

D. Evaluation on UCSD *and* LHI

Both *UCSD* and *LHI* are low resolution datasets but contain crowded scenes, which are challenging for multi-object tracking. In Table I, we report the tracking results for different combinations of trackers and detectors on *UCSD*. Using the overall MOTA scores, our TBC performs better than DCEM-Faster, GOGA-Faster, MHT-Faster and FT-Faster (e.g., MOTA of 37.1 vs 23.7, 33.8, 16.6, and 15.6), which indicates that jointly imposing object count constraint and flow constraint on crowd density maps will greatly improve the tracking results in crowded scenes.

TBC3 uses the model in (5) with flow constraint for object detection, while FT-IP only uses IP method for object detection. Though the two baselines have the same data association model FT, the detection result of TBC3 is much better than FT-IP (e.g., PRCN of 80.5 vs 64.7), which means that flow constraint on density maps can improve object detection performance by making the detections consistent between frames. In addition, TBC3 also performs better than FT-IP in terms of tracking results (MOTA) (e.g., MOTA of 36.4 vs 21.1). For the FT tracker, using the IP detector performs better than using the traditional trackers DPM and Faster-rcnn (e.g., MOTA of 21.1 vs -12.1 and 15.6), which indicates that density maps are effective for object detection in crowded scenes.

TBC jointly optimizes object detection and tracking over all the frames, while TBC3 separates object detection and tracking. From the tracking results, TBC achieves higher MOTA than TBC3, which indicates that using flow constraint on all the frames is more helpful for detection and tracking than only using flow constraint on part of the frames. However, TBC takes more time for optimization compared with TBC3.

In Table II, we summarize the tracking results on *LHI*. Our TBC still performs better compared to DCEM-IP, GOGA-Faster, MHT-Faster and FT-DA, showing higher MT and MOTA, e.g., MT of 29 vs 13, 23, 26 and 5, and MOTA of 73.1 vs 25.7, 62.7, 25.7, and 39.0. It also demonstrates that min-cost network flow tracking methods perform better than other methods in crowded scenes. For example, GOGA-Faster has higher MOTA than DCEM-Faster and MHT-Faster (e.g., MOTA of 62.7 vs 20.8 and 25.7). DA has a relatively better detection result than that of Faster-RCNN and DPM, which also indicates that density maps are effective for object detection in crowded scenes. From IDF1, our tracker can also achieve good tracking performance on both *UCSD* and *LHI*.

E. Evaluation on Fish *and* Cell

The *Fish* and *Cell* datasets both feature crowded scenes with small objects that deform or change appearance, which lead to the failures of DPM and Faster-RCNN. We thus only use the IP detector to generate detection results for evaluating all the competing methods. Quantitative results on *Fish* and *Cell* are summarized in Table III. Both DCEM and MHT assume that object trajectories are smooth and continuous, and adopt a motion model to recover the potential missed detections that follow the trajectory assumption. Although this recovery strategy works well for people tracking, it violates the fact that objects like fish and cell usually change their moving directions randomly and suddenly. This is the reason why DCEM and MHT have high FPs which in turn results in low MOTAs. In terms of MOTA, the flow-based method GOGA performs better than DCEM and MHT (e.g., for *cell*, MOTA of 73.9 vs 34.1 and 49.1), but worse than the TBC3 model. Our TBC model achieve the best results among all the methods by simultaneously optimizing detection and tracking.

F. Evaluation on PETS2009

The *PETS2009* S2-L2 sequence features many intersecting trajectories, making it challenging for multi-object tracking. Table IV reports the tracking results on *PETS2009*. Here, we

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

TABLE I: Multi-object tracking results on *UCSD*.

Tracker	Detector	RCLL(%)↑	PRCN(%)↑	FAF↓	GT	MT↑	PT	ML↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	IDF1↑	MOTP↑
DCEM	DPM	24.6	66.2	6.57	62	0	30	32	1313	6616	277	282	6.5	7.4	63.1
	Faster	36.4	77.9	4.54	62	2	44	16	909	5581	210	281	23.7	31.1	61.1
	IP	34.1	74.0	5.27	62	1	45	16	1054	5782	236	317	19.4	18.5	60.5
GOGA	DPM	11.2	39.0	7.71	62	0	12	50	1543	7791	159	230	-8.2	7.3	61.9
	Faster	**57.4**	72.7	9.43	62	9	45	8	1886	**3742**	184	485	33.8	**38.9**	63.2
	IP	54.3	73.7	8.52	62	6	52	**4**	1703	4008	799	879	25.8	28.8	67.6
MHT	DPM	13.7	36.6	10.39	62	0	18	44	2078	7575	93	209	-11.1	5.6	61.3
	Faster	51.6	65.3	18.29	62	12	40	10	3657	4247	289	530	16.6	23.4	60.3
	IP	40.3	62.0	16.34	62	7	36	19	3268	5238	291	434	10.2	19.2	62.9
FT	DPM	13.6	36.8	10.21	62	0	17	45	2043	7586	211	340	-12.1	6.8	62.2
	Faster	46.0	61.2	12.81	62	7	43	12	2563	4735	108	593	15.6	34.6	61.2
	IP	51.6	64.7	12.35	62	8	47	7	2470	4248	207	702	21.1	31.2	62.1
	DA	20.3	76.9	**2.69**	62	0	26	36	**537**	6991	**114**	**302**	12.9	14.4	70.1
TBC3 (ours)		50.5	80.5	5.38	62	**14**	34	14	1077	4341	166	419	36.4	36.1	**74.1**
TBC (ours)		49.5	**82.1**	4.74	62	11	37	14	948	4435	140	389	**37.1**	38.7	74.0

TABLE II: Multi-object tracking results on *LHI*.

Tracker	Detector	RCLL(%)↑	PRCN(%)↑	FAF↓	GT	MT↑	PT	ML↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	IDF1↑	MOTP↑
DCEM	DPM	23.4	63.1	3.19	43	1	19	23	1275	7147	274	313	6.8	6.6	67.6
	Faster	45.6	66.0	5.46	43	7	23	13	2186	5077	126	198	20.8	31.1	68.4
	IP	58.4	65.8	7.08	43	13	24	6	2830	3878	224	238	25.7	31.6	70.9
GOGA	DPM	34.3	89.2	0.97	43	4	26	13	388	6129	307	251	26.8	33.4	74.1
	Faster	74.5	88.3	2.31	43	23	19	1	924	2379	173	295	62.7	51.9	70.0
	IP	60.8	81.0	3.33	43	10	28	5	1331	3660	503	554	41.1	42.5	71.4
MHT	DPM	43.0	66.2	5.12	43	5	23	15	2046	5313	90	**69**	20.1	29.6	74.5
	Faster	**79.6**	61.1	11.84	43	26	16	1	4734	**1903**	291	206	25.7	38.8	69.5
	IP	58.2	63.0	7.97	43	12	26	5	3187	3897	232	220	21.6	36.3	72.1
FT	DPM	39.4	82.2	1.99	43	5	25	13	797	5656	402	399	26.5	36.4	73.6
	Faster	77.1	78.4	4.95	43	26	16	1	1979	2139	121	297	54.6	56.9	75.9
	IP	72.6	95.3	0.84	43	16	26	1	337	2559	167	242	67.2	52.4	77.0
	DA	61.6	82.0	3.15	43	5	35	3	1258	3584	848	888	39.0	21.5	72.6
TBC3 (ours)		73.3	**95.9**	**0.73**	43	18	24	1	**293**	2490	76	132	69.3	63.5	**81.6**
TBC (ours)		79.1	93.6	1.26	43	**29**	13	1	504	1945	**63**	110	**73.1**	**64.6**	81.3

TABLE III: Multi-object tracking results on *Fish* and *Cell*.

Dataset	Tracker	Detector	RCLL(%)↑	PRCN(%)↑	FAF↓	GT	MT↑	PT	ML↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	IDF1↑	MOTP↑
Fish	DCEM	IP	37.5	59.5	30.51	279	11	196	72	3936	4280	602	592	8.7	16.7	65.9
	GOGA	IP	35.5	**71.5**	**7.51**	279	12	170	97	**969**	4415	474	478	14.5	23.7	68.3
	MHT	IP	**44.2**	58.2	38.01	279	**26**	192	**61**	4903	**3819**	663	541	7.0	14.4	66.7
	FT	IP	43.1	62.9	13.49	279	15	183	81	1740	3894	262	604	13.9	**31.3**	67.0
	TBC3 (ours)		32.5	67.3	8.41	279	16	134	129	1085	4621	**141**	**423**	14.6	28.0	**68.6**
	TBC (ours)		37.9	65.3	10.67	279	19	155	105	1377	4254	164	446	**15.4**	29.9	68.3
Cell	DCEM	IP	61.5	70.7	23.96	265	**91**	109	65	2204	3329	161	237	34.1	42.1	73.6
	GOGA	IP	77.7	**98.0**	**1.52**	265	165	59	41	**140**	1928	184	161	73.9	**69.3**	83.8
	MHT	IP	81.8	74.4	26.48	265	181	56	**28**	2436	1576	389	165	49.1	47.0	82.0
	FT	IP	80.6	97.4	2.01	265	185	51	29	185	1673	216	268	76.0	61.8	83.8
	TBC3 (ours)		81.2	96.4	2.88	265	185	51	29	265	1623	207	236	75.7	62.7	**88.0**
	TBC (ours)		**85.4**	95.4	3.87	265	199	36	30	356	**1263**	**153**	**138**	**79.5**	63.7	**88.0**

TABLE IV: Comparisons with tracking-by-detection models on S2L2: (top) using MOT evaluation server; (bottom) using ground-truth from Milan et al. [18]

Tracker	RCLL(%)↑	PRCN(%)↑	FAF↓	GT	MT↑	PT	ML↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	MOTP↑
MHT [81]	-	-	2.10	42	8	31	3	933	3667	**142**	**201**	50.8	70.4
AMIR [82]	-	-	**1.40**	42	5	33	4	**616**	4236	254	397	47.0	**70.5**
TBC3 (ours)	71.8	83.7	3.23	42	15	26	1	1409	2701	558	467	51.4	65.8
TBC (ours)	78.5	85.3	2.99	42	**19**	23	**0**	1303	**2072**	531	562	**59.5**	67.2
Milan et al. [18]	65.5	89.8	1.43	74	28	34	12	622	2881	99	73	56.9	59.4
Berclaz et al. [25]	26.8	92.1	**0.44**	74	7	27	40	**193**	6117	**22**	**38**	24.2	60.9
Andriyenko et al. [83]	53.9	93.7	0.69	74	15	45	14	301	3850	152	128	48.5	62.0
Andriyenko et al. [84]	52.6	94.7	0.56	74	15	48	11	245	3957	143	125	48.0	61.6
Pirsiavash et al. [85]	49.0	**95.4**	0.46	74	7	50	17	199	4257	137	216	45.0	64.1
Wen et al. [86]	71.2	90.3	1.47	74	27	44	3	640	2402	125	175	62.1	52.7
Wen et al. [87]	74.4	89.8	1.62	74	30	42	2	708	2141	136	235	**64.2**	57.3
TBC3 (ours)	75.6	83.1	2.94	74	32	40	2	1281	2036	327	397	56.4	**69.5**
TBC (ours)	**78.4**	81.9	3.32	74	**40**	33	**1**	1448	**1805**	300	352	57.5	66.6

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

TABLE V: Effects of different parameters on *UCSD*.

Parameter	Setting	RCLL(%)↑	PRCN(%)↑	FAF↓	GT	MT↑	PT	ML↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	IDF1↓	MOTP↑
Edge Cost	Location	**52.7**	77.4	6.75	62	**13**	37	12	1351	**4150**	268	518	34.3	37.6	73.5
	Appearance	52.3	77.2	6.80	62	12	38	12	1360	4184	388	580	32.4	36.1	73.8
	Motion	51.7	76.1	7.14	62	10	39	13	1428	4242	975	663	24.3	29.4	73.6
	All	49.5	**82.1**	**4.74**	62	11	37	14	**948**	4435	**140**	**389**	**37.1**	**38.7**	**74.0**
Track Cost	c_{si}=0	44.7	**85.4**	**3.35**	62	6	38	18	**671**	4852	**127**	**267**	35.6	35.6	**74.5**
	c_{si}=5	48.7	81.8	4.77	62	10	34	18	954	4504	145	373	36.2	38.3	74.1
	c_{si}=10	49.5	82.1	4.74	62	**11**	37	**14**	948	**4435**	140	389	**37.1**	**38.7**	74.0
	c_{si}=15	48.7	82.2	4.64	62	9	35	18	929	4501	141	379	36.3	38.5	74.1
	c_{si}=20	48.5	82.5	4.51	62	9	35	18	902	4520	133	373	35.3	**38.7**	74.1
Window size	w=$\frac{1}{3}w_0$	47.1	79.7	5.28	62	5	37	20	1055	4646	156	393	33.3	34.1	73.8
	w=$\frac{1}{2}w_0$	48.7	81.5	4.84	62	10	34	18	968	4503	141	388	36.1	37.9	74.0
	w=w_0	49.5	82.1	4.74	62	**11**	37	**14**	948	**4435**	140	389	**37.1**	**38.7**	74.0
	w=2w_0	45.2	81.8	4.41	62	8	34	20	881	4812	154	378	33.4	33.9	**74.2**
	w=3w_0	42.9	**82.6**	**3.98**	62	3	37	22	**796**	5011	**129**	**339**	32.4	32.2	74.1

directly use the reported results from MHT and AMIR on S2L2 (see Tab. IV (top)), and our result is also evaluated through the MOT server. TBC achieves higher MOTA than MHT and AMIR (59.5 vs 50.8 and 47.0) on S2L2 (Note that AMIR ranks third on MOT15). We also compare with the results from a recent paper [87] on an ROI of S2L2 (using ground truth from Milan et al. [18]). Even without using the provided detection results, our model can achieve comparable results with these baselines, but is able to find more tracks (higher MT and Recall). Note that the number of GT trajectories are different due to using different annotations.

The top two rows of Fig. 4 show the tracking results of three competing trackers DCEM, GOGA, MHT and our TBC using detections generated by Faster-RCNN. Though the retrained Faster-RCNN can recognize most of the people in the scene, it still misses some objects (marked with red arrows) when they are heavily occluded, which suggests that crowd density maps are very useful for object detection in crowded scenes. In the last two rows of Fig. 4, we show the tracking results of *Fish* and *Cell*, both of which have many intersecting paths. For *Fish*, DCEM and MHT generate many FPs. GOGA achieves good tracking results but also discards many true positives. Using global optimization of object detection and data association, our TBC model achieves less IDS and FM. Consistent with *Fish*, the TBC model also works the best among all the trackers. The video results can be found in our supplementary material.

G. Running time

The running time of TBC is affected by the image resolution and density maps used for building a graph. The average per-frame running times for *UCSD*, *LHI*, *Fish*, *Cell* and *PETS2009* are 2.4, 4.1, 12.9, 16.3 and 6.4 sec, respectively. The reported running time only includes solving the energy function in (5). Note that the reported time is for TBC, which jointly optimizes detection and tracking over the whole sequence. For TBC3 (which jointly optimizes 3 frames at a time), the average time (for all the scenes) can be reduced to less than 0.03s. The tracking speed is directly correlated with the number of sliding windows, which are then determined by the number of the targets and the number of frames considered in the optimization. Taking *UCSD* as an example, a crowded scene (average 38 people) with 200 frames has an average running time of 2.4s, while a sparse scene (average 17 people) with the same number of frames has an average running time of 0.6s. Furthermore, for *LHI* (average 22 people) with 400 frames, the running time is 4.1s.

H. Effects of parameter settings on tracking performance

In this subsection, we analyze the effects of parameter settings on tracking performance. The edge cost c_{ij} in (6) consists of three types of input information, and we first analyze the effect of each one by removing the other two. In addition, the track start cost c_{si} and terminate cost c_{it} also affect the tracking performance, and we test them by using different settings. Finally, we analyze the performance of TBC model by using different sliding window sizes in (5).

All the experiments are carried on *UCSD* using TBC model, and the results are reported in Tab. V. The edge cost c_{ij} incorporates geometric location, target appearance and motion information for association, and each of them contributes to the tracking performance. Large track cost c_{si} or c_{it} reduces IDS and thus can improve IDF1, but this will degrade the tracking accuracy. E.g., using $c_{si} = 20$ achieves less IDS compared with using $c_{si} = 10$ (133 vs. 140), but has lower MOTA (35.3 vs. 37.1).

The base sliding window is set as the average target size (i.e., w_0). With the base window, the sliding windows used for detections will be adjusted by perspective map. A window with either too large spatial size or too small spatial size will degrade the detection results, and further affects the tracking performance. For a small spatial window, the sum within the window is far less than 1, resulting in missed detection. For a large window, the window will contain too many objects that are hard to be accurately localized. As observed, when we shrink and expand the base window to 1/3 and 3 times of its original size, the MOTA drops from 37.1 to 33.3 and 32.4 respectively. On the other hand, the MOTP is not affected much by the base window size.

To reduce computational complexity, we dropped the candidate points with low density value ($<$0.005) when solving (5). Here, we draw a plot of computation vs. performance for different MILP problem sizes (i.e., different threshold settings). From 0.001 to 0.009, we choose 9 thresholds with step size

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

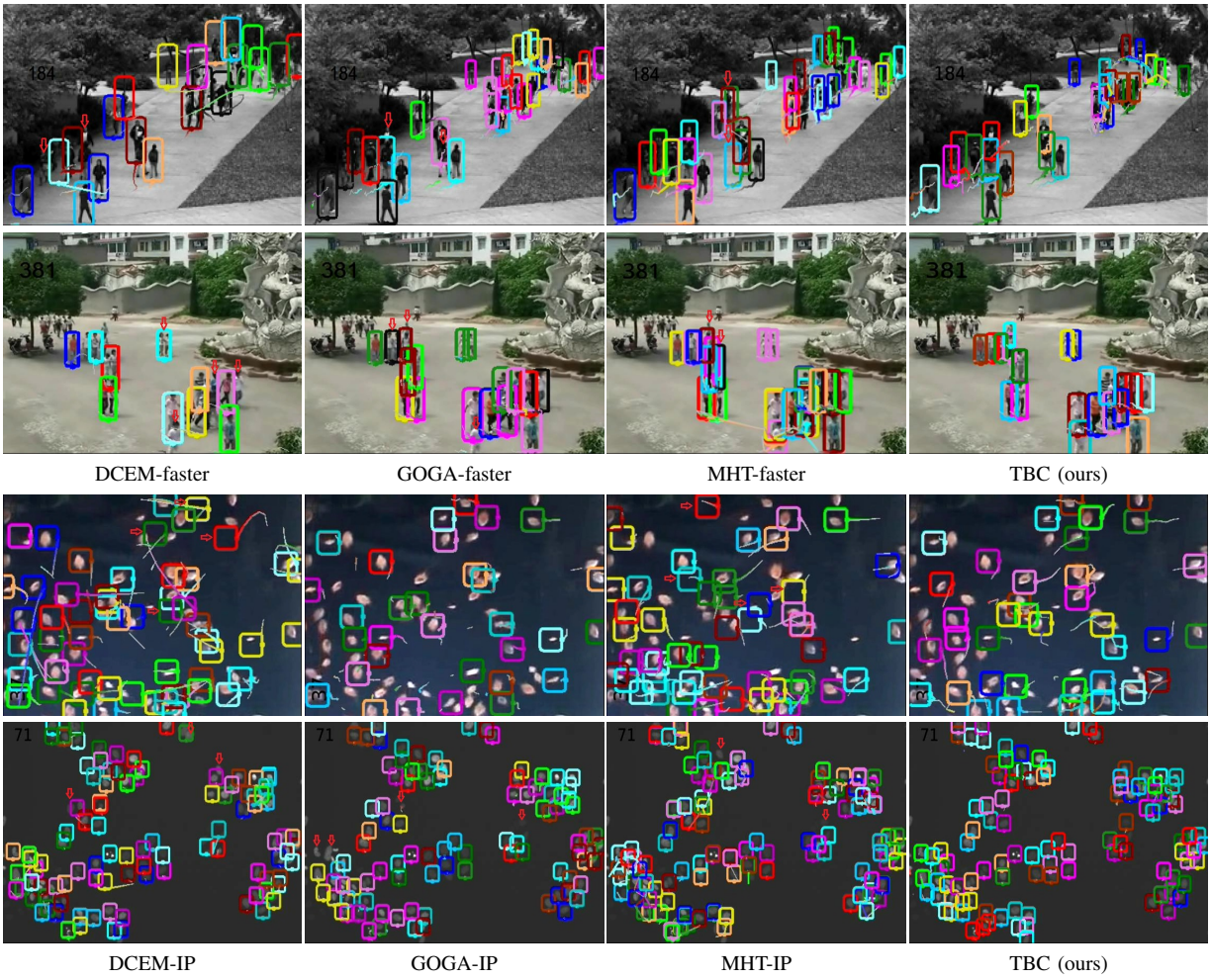


Fig. 4: Qualitative results of our tracking-by-counting model. From top to bottom, the results are on *UCSD*, *LHI*, *Fish* and *Cell*, respectively.

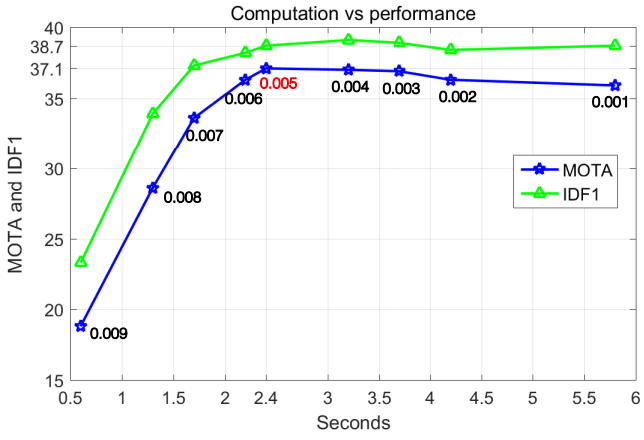


Fig. 5: Plot of computation vs. performance for different MILP problem sizes (i.e., different threshold settings).

0.001 to separately evaluate TBC model on *UCSD*. As shown in Fig. 5, when the threshold is set to 0.005 (marked red in the plot), a good balance between computation and tracking performance is achieved (MOTA=37.1, IDF1=38.7 and time = 2.4s). Besides, the threshold 0.005 also works well for other datasets.

V. Evaluation on Large Scale Datasets

Our previous experiments are conducted on crowd scenes with relatively low resolution, where the traditional object detector often fails to localize objects. Using crowd density maps, our *tracking-by-counting* model has advantage of detecting and tracking small objects over other methods in these scenes. For large scale datasets with high-resolution, the traditional object detector can achieve good detection results. To further evaluate our model on large scale scenes, we extend our *tracking-by-counting* model, by incorporating detection results produced by object detectors:

$$\begin{aligned}
\min_{\mathbf{x}} \; & \mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{m}^T\mathbf{x} + \sum_{t=1}^{\mathrm{T}}\sum_{k=1}^{\mathrm{K}_t}\left|\left(\mathbf{w}_k^t\right)^T\mathbf{x} - \hat{n}_k^t\right| + \sum_{ij\in E} c_{ij}x_{ij} + \\
& \sum_i c_{si}x_{si} + \sum_i c_{it}x_{it} \\
\text{s.t.} \; & \sum_{i:ij\in E} x_{ij} + x_{sj} = x_j = \sum_{i:ji\in E} x_{ji} + x_{jt} \\
& \sum_i x_{it} = \sum_i x_{si}, \\
& x_i, x_{ij} \in \{0,1\},
\end{aligned} \tag{9}$$

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

where the first term ensures that only non-overlapping locations are selected, by setting $\mathbf{H}_{ij} = \infty$ if locations i and j have significant overlap ratio (which we set to be 0.65), and 0 otherwise. The second term is the total detection score, where the vectorized score map $\mathbf{m}$ is obtained by placing Gaussian radial basis functions at each detected location, with height equal to its negative detection score. By adding the first two terms, our model can combine object detections with estimated density maps together to jointly optimize detections and trajectories.

Similar to (8), we also use auxiliary variable z_k^t to replace $\left|\left(\mathbf{w}_k^t\right)^T \mathbf{x}^d - n_k^t\right|$, and rewrite (9) as a quadratically mixed integer least square problem which also can be solved with optimization toolboxes:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{z}} \; & \mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{m}^T\mathbf{x} + \sum_{t=1}^{\mathrm{T}}\sum_{k=1}^{\mathrm{K}_t} z_k^t + \sum_{ij\in E} c_{ij}x_{ij} + \\
& \sum_i c_{si}x_{si} + \sum_i c_{it}x_{it} \\
\text{s.t.} \; & \sum_{i:ij\in E} x_{ij} + x_{sj} = x_j = \sum_{i:ji\in E} x_{ji} + x_{jt} \\
& \sum_i x_{it} = \sum_i x_{si} \\
& \left(\mathbf{w}_k^t\right)^T \mathbf{x} - n_k^t - z_k^t \le 0 \\
& -\left(\mathbf{w}_k^t\right)^T \mathbf{x} + n_k^t - z_k^t \le 0 \\
& x_i, x_{ij} \in \{0,1\}, 0 \le z_k^t.
\end{aligned} \tag{10}$$

A. Experiments

The average results of *MOT17* are reported in Tab. VI, and detailed results can be found on the MOT challenge website (our tracker is "MOT_TBC"). Here, we only report the results of TBC3, since some sequences are too long, which will consume excessive memory to build a graph over the whole sequence. In terms of MOTA, TBC3 performs better than other *tracking-by-detection* methods. TBC3 has lower FN, showing that our model can recover the missed objects using estimated density maps. To validate the efficiency of our model in (9), we also create two variants, by removing the first two detection terms ("TBC3-Count") and by removing the third counting term ("TBC3-Det"). As observed, both the detection term and the counting term contribute to improving the tracking performance. For the running time, TBC3 model runs the fastest among all the trackers.

Also, we evaluate our model on *DukeMTMC*, and the results are summarized in Tab. VII (our tracker is "MTMC_TBC" in the MOT challenge). For the easy test set, our TBC3 tracker achieves the same IDF1 value (89.2) with DeepCC, but has higher MOTA and MT. For the hard test set, TBC3 has higher IDF1 and MOTA, and lower FP and FN, which indicates that density maps can improve both detection and tracking performance especially in crowd scenes. The running time of TBC3 on *DukeMTMC* is similar on *MOT17*.

Finally, we evaluate our model on *MOT20* dataset where the average number of targets per frame can reach 100. In Tab. IX, we compare our TBC3 model with three released works UnsupTrack [95], SORT [96] and Fair [97] from MOT20. Note that Fair [97] used private detections for tracking, while our method, UnsupTrack, and SORT used public detections. Fair [97] adopted a variant DLA-34 proposed in [98] as the backbone, and the parameters are pretrained on COCO detection dataset [99]. Thus, it achieves very high MOTA and IDF1. [95] trained a ReID network to cluster tracklets, and thus achieves relatively higher IDF1. However, our TBC3 model can find more missed detecions using density maps, and thus has higher MOTA and MT than [95]. These results indicates the effectiveness of our TBC model on crowd scenes. See MOT20 challenge for more details (our tracker is denoted as "MOT20-TBC").

In Fig. 6, we show tracking results of one example from *MOT17*. JCC [88] changes the object identity from "2" to "3" when two objects meet, while Tracktor17 [60] causes identity switches. Using both object detections and density maps, our TBC model can effectively reduce IDS. Besides, it can recover missed detections (marked as red arrows).

B. Adding detections for UCSD, LHI and PETS2009

In the experiments on crowd scenes in Section IV-D, we only used the model in (8) for tracking, and did not use the detections for *UCSD*, *LHI* and *PETS2009*. Here, we introduce detections for them (denoted as "TBC+Det"), and analyze if the model in (10) can further improve the tracking performance. The tracking results when adding detections to TBC are summarized in Tab. VIII. For *LHI* and *PETS2009*, TBC+Det model can further improve TBC, e.g., MOTA is improved from 57.5 to 59.6 on *PETS2009*. For *UCSD*, TBC+Det yields slightly lower MOTA compared with TBC, but has higher MT and IDF1. In summary, tracking results can be further improved when combining TBC model with object detections as in (10).

VI. Conclusions

In this paper, we propose a novel MOT approach explicitly designed for crowded scenes. Unlike existing tracking-by-detection MOT methods that focus on associating per-frame detections and existing density-based tracking approaches that rely on heuristic short point-tracks, our model explicitly accounts for the object counts inferred from density maps and simultaneously solves multi-object detection and tracking over the whole video sequence. This is achieved by modeling the joint problem as a network flow program for which the optimal solutions can be found using standard commercial solvers. Our method achieves promising results on public datasets featuring people-, cell-, and fish-tracking scenarios. We also extend our model to use object detections, which can further improve performance on large-scale scenes.

Acknowledgements

The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. [T32-101/15-R] and CityU 11212518), by a Strategic Research Grant from City University of Hong Kong (Project No. 7004887), and

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

TABLE VI: Multi-object tracking results on *MOT17*.

Tracker	MOTA↑	IDF1↑	MOTP↑	FAF↓	MT↑	ML↓	FP↓	FN↓	IDS↓	FM↓	Time↑
JCC [88]	51.2	54.5	75.9	1.5	20.9%	37.0%	25,937	247,822	1,802	2,984	1.8 Hz
BLSTM [89]	47.5	51.9	**77.5**	1.5	18.2%	41.7%	25,981	268,042	2,069	3,124	1.9 Hz
DMAN [90]	48.2	**55.7**	75.7	1.5	19.3 %	38.3%	26,218	263,608	2,194	5,378	0.3 Hz
MHT [81]	50.7	47.2	**77.5**	**1.3**	20.8%	36.9%	**22,875**	252,889	2,314	2,865	0.9 Hz
SAS_MOT17 [57]	44.2	57.2	76.4	1.7	16.1%	44.3%	29,473	283,611	**1,529**	**2,644**	4.8 Hz
MTDF17 [91]	49.6	45.2	75.5	2.1	18.9%	**33.1%**	37,124	241,768	5,567	9,260	1.2 Hz
Tracktor17 [60]	53.5	52.3	78.0	0.7	19.5%	36.6%	12,201	248,047	2,072	4,611	1.5 Hz
TBC3 (ours)	**53.9**	50.0	76.8	1.4	20.2%	**36.7%**	24,584	**232,670**	2,945	4,612	6.7 Hz
TBC3-Det	42.8	41.9	76.9	1.9	15.4%	42.1%	33,670	28,6397	2,815	5,196	10.9 Hz
TBC3-Count	38.3	38.2	73.1	5.3	17.6%	38.9%	9,5178	247,542	5,439	13,101	8.4 Hz

TABLE VII: Multi-object tracking results on *DukeMTMC* dataset.

Dataset	Tracker	IDF1↑	IDP↑	IDR↑	MOTA↑	MOTP↑	FAF↓	MT↑	ML↓	FP↓	FN↓	IDS↓	FM↓
Test-easy	DeepCC [92]	**89.2**	**91.7**	86.7	87.5	77.1	**0.05**	1,103	29	**37,280**	94,399	**202**	**753**
	TAREIDMTMC [93]	75.7	77.7	73.8	84.1	73.8	0.08	1,080	**7**	56,063	110,038	2,068	10,098
	PT_BIPCC [94]	71.2	84.8	61.4	59.3	**78.7**	0.09	666	234	68,634	361,589	290	783
	TBC3 (ours)	**89.2**	90.4	**88.0**	**87.8**	78.1	0.07	**1,131**	22	50,771	**78,263**	222	833
Test-hard	DeepCC [92]	79.0	87.4	72.0	70.0	75.0	0.15	524	66	43,989	170,104	**236**	**777**
	TAREIDMTMC [93]	68.2	74.4	63.0	68.0	72.6	0.20	515	**38**	57,995	167,625	2,485	9,219
	PT_BIPCC [94]	65.0	81.8	54.0	54.4	**77.1**	0.14	335	104	40,978	283,704	661	1,054
	TBC3 (ours)	**82.4**	**90.0**	**76.0**	**75.2**	76.4	**0.12**	**569**	52	**32,813**	**143,883**	298	898

TABLE VIII: Tracking results on *UCSD*, *LHI* and *S2L2* when adding detections.

Dataset	Tracker Detector	RCLL(%)↑	PRCN(%)↑	FAF↓	GT	MT↑	PT	ML↓	FP↓	FN↓	IDS↓	FM↓	MOTA↑	IDF1↓	MOTP↑
UCSD	GOGA Faster	57.4	72.7	9.43	62	9	45	8	1886	3742	184	485	33.8	38.9	63.2
	TBC (ours)	49.5	**82.1**	**4.74**	62	11	37	14	**948**	4435	**140**	**389**	**37.1**	38.7	**74.0**
	TBC+Det	**58.7**	72.6	9.72	62	**16**	39	**7**	1944	**3627**	171	508	35.6	**39.2**	69.6
LHI	GOGA Faster	74.5	88.3	2.31	43	23	19	1	924	2379	173	295	62.7	51.9	70.0
	TBC (ours)	79.1	**93.6**	**1.26**	43	29	13	1	**504**	1945	**63**	**110**	73.1	64.6	**81.3**
	TBC+Det	**81.1**	93.0	1.42	43	**30**	12	1	569	**1759**	86	132	**74.1**	**65.3**	81.2
S2L2	Wen et al. [86]	71.2	90.3	1.47	74	27	44	3	640	2402	125	175	62.1	-	52.7
	Wen et al. [87]	74.4	89.8	1.62	74	30	42	2	708	2141	136	235	**64.2**	-	57.3
	TBC(ours)	78.4	81.9	3.32	74	40	33	**1**	1448	1805	300	352	57.5	-	66.6
	TBC+Det	**79.5**	82.6	3.22	74	**41**	32	**1**	1404	**1710**	257	281	59.6	-	**69.7**

TABLE IX: Multi-object tracking results on *MOT20*.

Tracker	detector	MOTA↑	IDF1↑	MT↑	ML↓	IDS↓
UnsupTrack [95]	public	53.6	**50.6**	376	311	**2,178**
SORT [96]	public	42.7	45.1	208	326	4,470
TBC3	public	**54.5**	50.1	**415**	**245**	2,445
Fair [97]	private	61.8	67.3	855	94	5,243

by the Natural Science Foundation of China under Grant 61991413. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

REFERENCES

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

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015, pp. 91–99.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[4] S. Tang, M. Andriluka, and B. Schiele, "Detection and tracking of occluded people," *International Journal of Computer Vision*, vol. 110, no. 1, pp. 58–69, 2014.

[5] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2129–2137.

[6] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2903–2910.

[7] A. Milan, K. Schindler, and S. Roth, "Multi-target tracking by discrete-continuous energy minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2054–2068, 2016.

[8] M. Rodriguez, I. Laptev, J. Sivic, and J.-Y. Audibert, "Density-aware person detection and tracking in crowds," in *IEEE International Conference on Computer Vision*, 2011, pp. 2423–2430.

[9] A. Dehghan and M. Shah, "Binary quadratic programing for online tracking of hundreds of people in extremely crowded scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 568–581, 2017.

[10] W. Luo, J. Xing, X. Zhang *et al.*, "Multiple object tracking: A literature review," *arXiv:1409.7618*, 2014.

[11] J. Black, T. Ellis, and P. Rosin, "Multi view image surveillance and tracking," in *Motion and Video Computing, 2002. Proceedings. Workshop on*, 2002, pp. 169–174.

[12] X. Li, K. Wang, W. Wang, and Y. Li, "A multiple object tracking method using kalman filter," in *Proc. IEEE ICIA*, 2010, pp. 1862–1866.

[13] J. Giebel, D. M. Gavrila, and C. Schnörr, "A bayesian framework for multi-cue 3d object tracking," in *European Conference on Computer Vision*, 2004, pp. 241–252.

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

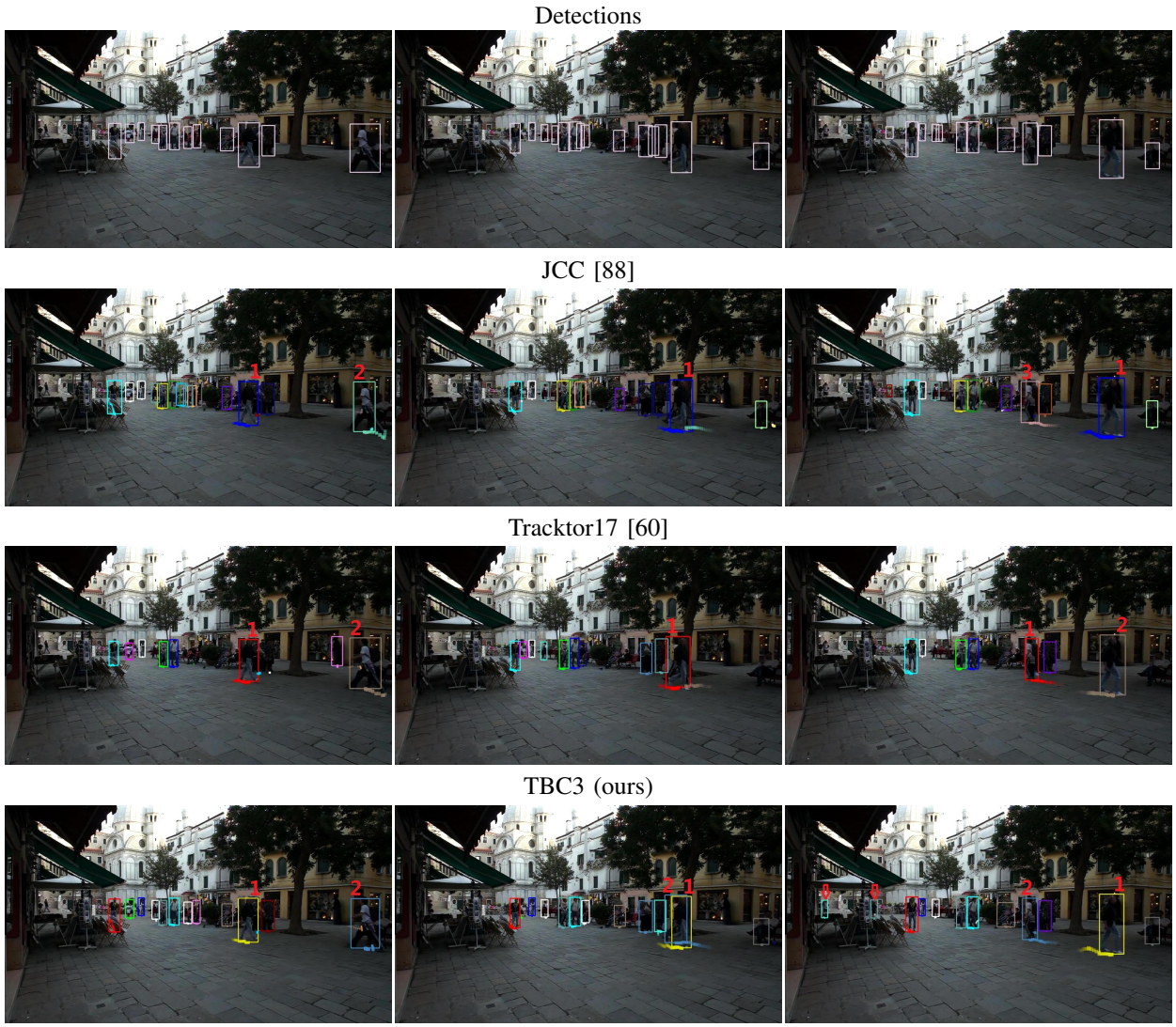


Fig. 6: Qualitative results on *MOT17*. From left to right, frames 45, 85 and 110. Using both object detections and density maps, our TBC model can effectively reduce IDS. Besides, it can recover missed detections (marked with red arrows)

[14] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *European Conference on Computer Vision*, 2004, pp. 28–39.
[15] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multi-target tracking," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 481–497, 2009.
[16] R. Girshick, "Fast r-cnn," in *IEEE International Conference on Computer Vision*, 2015.
[17] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1265–1272.
[18] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multitarget tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 58–72, 2014.
[19] S.-I. Yu, D. Meng, W. Zuo, and A. Hauptmann, "The solution path algorithm for identity-aware multi-object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3871–3879.
[20] A. Maksai, X. Wang, and P. Fua, "What players do with the ball: A physically constrained interaction modeling," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 972–981.
[21] L. Lan, X. Wang, G. Hua, T. S. Huang, and D. Tao, "Semi-online multi-people tracking by re-identification," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1937–1955, 2020.
[22] A. Maksai, X. Wang, F. Fleuret, and P. Fua, "Non-markovian globally consistent multi-object tracking," in *IEEE International Conference on Computer Vision*, 2017, pp. 2563–2573.
[23] L. Lan, X. Wang, S. Zhang, D. Tao, W. Gao, and T. S. Huang, "Interacting tracklets for multi-object tracking," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4585–4597, 2018.
[24] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
[25] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1806–1819, 2011.
[26] X. Wang, Z. Li, and D. Tao, "Subspaces indexing model on grassmann manifold for image search," *IEEE Transactions on Image Processing*, vol. 20, no. 9, pp. 2627–2635, 2011.
[27] A. A. Butt and R. T. Collins, "Multi-target tracking by lagrangian relaxation to min-cost network flow," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1846–1853.
[28] V. Chari, S. Lacoste-Julien, I. Laptev, and J. Sivic, "On pairwise costs for network flow multi-object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5537–5545.
[29] X. Wang, B. Fan, S. Chang, Z. Wang, X. Liu, D. Tao, and T. S. Huang, "Greedy batch-based minimum-cost flows for tracking multiple objects," *IEEE Transactions on Image Processing*, vol. 26, no. 10, pp. 4765–4776, 2017.
[30] X. Wang, E. Türetken, F. Fleuret, and P. Fua, "Tracking interacting objects using intertwined flows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2312–2326, 2016.
[31] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Multi-person tracking by multicut and deep matching," in *European Conference on Computer Vision*, 2016, pp. 100–111.

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

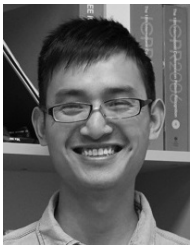
This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

[32] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.
[33] B. Yang and R. Nevatia, "An online learned crf model for multi-target tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2034–2041.
[34] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1201–1208.
[35] X. Wang, E. Türetken, F. Fleuret, and P. Fua, "Tracking interacting objects optimally using integer programming," in *European Conference on Computer Vision*, 2014, pp. 17–32.
[36] H. Jiang, S. Fels, and J. J. Little, "A linear programming approach for multiple object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.
[37] S. Wang and C. C. Fowlkes, "Learning optimal parameters for multi-target tracking with contextual interactions," *International Journal of Computer Vision*, vol. 122, no. 3, pp. 484–501, 2017.
[38] A. Dehghan, Y. Tian, P. H. Torr, and M. Shah, "Target identity-aware network flow for online multiple target tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1146–1154.
[39] S. Wang and C. C. Fowlkes, "Learning optimal parameters for multi-target tracking." in *British Machine Vision Conference*, vol. 1, no. 2, 2015, p. 6.
[40] R. Henschel, Y. Zou, and B. Rosenhahn, "Multiple people tracking using body and joint detections," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.
[41] L. Kratz and K. Nishino, "Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 987–1002, 2011.
[42] X. Zhao, D. Gong, and G. Medioni, "Tracking using motion patterns for very crowded scenes," in *European Conference on Computer Vision*, 2012, pp. 315–328.
[43] A. Bera, N. Galoppo, D. Sharlet, A. Lake, and D. Manocha, "Adapt: real-time adaptive pedestrian tracking for crowded scenes," in *Proc. IEEE ICRA*, 2014, pp. 1801–1808.
[44] W. Luo and T.-K. Kim, "Generic object crowd tracking by multi-task learning." in *British Machine Vision Conference*, 2013.
[45] H. Fradi, V. Eiselein, J.-L. Dugelay, I. Keller, and T. Sikora, "Spatio-temporal crowd density model in a human detection and tracking framework," *Signal Processing: Image Communication*, vol. 31, pp. 100–111, 2015.
[46] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *European Conference on Computer Vision*, 2016, pp. 749–765.
[47] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," *arXiv:1708.02973*, 2017.
[48] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for correlation filter based tracking," *arXiv:1704.06036*, 2017.
[49] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler, "Learning by tracking: Siamese cnn for robust target association," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 33–40.
[50] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, K. Luk Chan, and G. Wang, "Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 1–8.
[51] Y. Yang, Z. Feng, M. Song, and X. Wang, "Factorizable graph convolutional networks," in *Neural Information Processing Systems*, 2020.
[52] Q. Chu, W. Ouyang, H. Li *et al.*, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," in *IEEE International Conference on Computer Vision*, Oct 2017, pp. 4846 – 4855.
[53] J. Son, M. Baek, M. Cho, and B. Han, "Multi-object tracking with quadruplet convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017, pp. 1063–6919.
[54] Y. Yang, J. Qiu, M. Song, D. Tao, and X. Wang, "Distilling knowledge from graph convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7072–7081.
[55] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks." in *Proc. AAAI*, 2017, pp. 4225–4232.
[56] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *IEEE International Conference on Computer Vision*, Oct 2017, pp. 300–311.
[57] A. Maksai and P. Fua, "Eliminating exposure bias and metric mismatch in multiple object tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4639–4646.
[58] K. Bozek, L. Hebert, A. S. Mikheyev *et al.*, "Towards dense object tracking in a 2d honeybee hive," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4185–4193.
[59] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015, pp. 234–241.
[60] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, "Tracking without bells and whistles," in *IEEE International Conference on Computer Vision*, 2019.
[61] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Proc. NIPS*, 2010, pp. 1324–1332.
[62] L. Fiaschi, U. Koethe, R. Nair, and F. A. Hamprecht, "Learning to count with regression forest and structured labels," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2685–2688.
[63] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2547–2554.
[64] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 833–841.
[65] V. A. Sindagi and V. M. Patel, "Generating high-quality crowd density maps using contextual pyramid cnns," in *IEEE International Conference on Computer Vision*, 2017, pp. 1879–1888.
[66] D. B. Sam, S. Surya, and R. V. Babu, "Switching convolutional neural network for crowd counting," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 3, 2017, p. 6.
[67] D. Kang, D. Dhar, and A. Chan, "Incorporating side information by adaptive convolution," in *NIPS*, 2017, pp. 3870–3880.
[68] F. Xiong, X. Shi, and D.-Y. Yeung, "Spatiotemporal modeling for crowd counting in videos," in *IEEE International Conference on Computer Vision*, 2017, pp. 5161–5169.
[69] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura, "Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras," in *IEEE International Conference on Computer Vision*, 2017, pp. 3687–3696.
[70] Z. Ma, L. Yu, and A. B. Chan, "Small instance detection by integer programming on object density maps," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3689–3697.
[71] C. Liu, "Beyond pixels: exploring new representations and applications for motion analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.
[72] A. B. Chan and N. Vasconcelos, "Counting people with low-level features and bayesian regression," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2160–2177, 2012.
[73] B. Yao, X. Yang, and S.-C. Zhu, "Introduction to a large-scale general purpose ground truth database: methodology, annotation tool and benchmarks," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2007, pp. 169–183.
[74] M. Maška, V. Ulman, D. Svoboda *et al.*, "A benchmark for comparison of cell tracking algorithms," *Bioinformatics*, vol. 30, no. 11, pp. 1609–1617, 2014.
[75] V. Ulman, M. Maška, K. E. Magnusson *et al.*, "An objective comparison of cell-tracking algorithms," *Nature methods*, vol. 14, no. 12, p. 1141, 2017.
[76] J. Ferryman and A. Shahrokni, "Pets2009: Dataset and challenge," in *Performance Evaluation of Tracking and Surveillance, IEEE International Workshop on*, 2009, pp. 1–6.
[77] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.
[78] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *ECCV workshop on Benchmarking Multi-Target Tracking*, 2016.
[79] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *Journal on Image and Video Processing*, vol. 2008, p. 1, 2008.
[80] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision Workshops*, 2016, pp. 17–35.
[81] C. Kim, F. Li, A. Ciptadi *et al.*, "Multiple hypothesis tracking revisited," in *IEEE International Conference on Computer Vision*, 2015, pp. 4696–4704.
[82] A. A. Amir Sadeghian and S. Savarese1, "Tracking the untrackable:

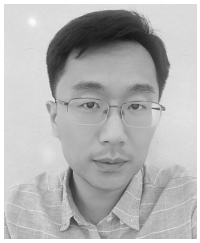
Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2020.3044219

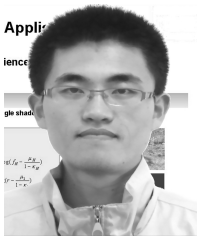
Learning to track multiple cues with long-term dependencies," in *IEEE International Conference on Computer Vision*, 2017, pp. 300–311.
[83] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1265–1272.
[84] K. S. Andriyenko, Anton and S. Roth, "Discrete-continuous optimization for multi-target tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1926–1933.
[85] D. R. Pirsiavash, Hamed and C. C. Fowlkes, "Discrete-continuous optimization for multi-target tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1201–1208.
[86] L. Wen, W. Li, J. Yan, Z. Lei *et al.*, "Multiple target tracking based on undirected hierarchical relation hypergraph," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1282–1289.
[87] L. Wen, Z. Lei, S. Lyu, S. Z. Li, and M. Yang, "Exploiting hierarchical dense structures on hypergraphs for multi-object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 1983–1996, 2016.
[88] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele, "Motion segmentation & multiple object tracking by correlation co-clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
[89] C. Kim, F. Li, and J. M. Rehg, "Multi-object tracking with neural gating using bilinear lstm," in *European Conference on Computer Vision*, 2018, pp. 200–215.
[90] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *European Conference on Computer Vision*, 2018, pp. 366–382.
[91] Z. Fu, F. Angelini, J. Chambers, and S. M. Naqvi, "Multi-level cooperative fusion of gm-phd filters for online multiple human tracking," *IEEE Transactions on Multimedia*, 2019.
[92] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6036–6046.
[93] Y. X. C. X. Z. Z. Na Jing, ShiChen Bai and W. Wu, "Online inter-camera trajectory association exploiting person re-identification and camera topology," in *Proc. ACM MM*, 2018, pp. 1457–1465.
[94] F. F. Andrii Maksai, Xinchao Wang and P. Fua, "Globally consistent multi-people tracking using motion patterns," in *IEEE International Conference on Computer Vision*, 2017, pp. 2544–2554.
[95] S. Karthik, A. Prabhu, and V. Gandhi, "Simple unsupervised multi-object tracking," *arXiv preprint arXiv:2006.02609*, 2020.
[96] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *IEEE International Conference on Image Processing*. IEEE, 2016, pp. 3464–3468.
[97] Y. Zhang, C. Wang, X. Wang, W. Zeng, and WenyuLiu, "A simple baseline for multi-object tracking," *arXiv preprint arXiv:2004.01888v2*, 2020.
[98] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2403–2412.
[99] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014, pp. 740–755.



Weihong Ren received the B.E. degree in Automation and Electronic Engineering from Qingdao University of Science and Technology, Qingdao, China, in 2013. He received the Ph.D. degree in the Department of Computer Science, City University of Hong Kong, in 2020. He also reveived the Ph.D. degree in Pattern Recognition and Intelligent System at Chinese Academy of Sciences, China, in 2020. His current research interests include people tracking, image restoration and deep learning.



Xinchao Wang has been a tenure-track Assistant Professor at Stevens Institute of Technology, New Jersey, United States, since November 2017. Before joining Stevens, he was an SNSF postdoctoral fellow at University of Illinois Urbana-Champaign (UIUC) with Prof. Thomas S. Huang. He received a PhD from Ecole Polytechnique Federale de Lausanne (EPFL) in 2015, and a first-class honorable degree from Hong Kong Polytechnic University (HKPU) in 2010. His research interests include artificial intelligence, computer vision, machine learning, medical image analysis, and multimedia. His articles have been published in major venues including CVPR, ICCV, ECCV, NeurIPS, AAAI, IJCAI, MICCAI, TPAMI, IJCV, TIP, TMI, and TNNLS. He is an associate editor of IEEE Transactions on Circuits and Systems for Video Technology (TCSVT) and Journal of Visual Communication and Image Representation (JVCI). He has been or will be serving as an area chair of CVPR, ICCV, ICIP, ICME, and as a senior program committee member of AAAI and IJCAI.



Jiandong Tian received his B.S. degree in the department of automation, Heilongjiang University, P.R. China, in 2005. In 2011 he received a doctor's degree in Shenyang Institute of Automation, Chinese Academy of Sciences. Currently he is a professor in Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests include Illumination and Reflectance Modeling,image processing, and pattern recognition.

Yandong Tang received the B.S. and M.S. degrees in mathematics from Shandong University, China, in 1984 and 1987, respectively, and the Ph.D. degree in applied mathematics from the University of Bremen, Germany, in 2002. He is currently a Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests include numerical computation, image processing, and computer vision.

Antoni B. Chan received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, in 2000 and 2001, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), San Diego, in 2008. He is currently an Associate Professor in the Department of Computer Science, City University of Hong Kong. His research interests include computer vision, machine learning, pattern recognition, and music analysis.

Copyright (c) 2020 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.