Detection, Tracking, and Counting Meets Drones in Crowds: A Benchmark

Longyin Wen[1,*], Dawei Du[2,*], Pengfei Zhu[3,†], Qinghua Hu[3], Qilong Wang[3], Liefeng Bo[1], Siwei Lyu[2]

[1]JD Finance America Corporation, Mountain View, CA, USA

[2]University at Albany, State University of New York, Albany, NY, USA

[3]Tianjin University, Tianjin, China

https://github.com/VisDrone/DroneCrowd

Abstract

To promote the developments of object detection, tracking and counting algorithms in drone-captured videos, we construct a benchmark with a new drone-captured large-scale dataset, named as DroneCrowd, formed by 112 video clips with 33,600 HD frames in various scenarios. Notably, we annotate 20,800 people trajectories with 4.8 million heads and several video-level attributes. Meanwhile, we design the Space-Time Neighbor-Aware Network (STNNet) as a strong baseline to solve object detection, tracking and counting jointly in dense crowds. STNNet is formed by the feature extraction module, followed by the density map estimation heads, and localization and association subnets. To exploit the context information of neighboring objects, we design the neighboring context loss to guide the association subnet training, which enforces consistent relative position of nearby objects in temporal domain. Extensive experiments on our DroneCrowd dataset demonstrate that STNNet performs favorably against the state-of-the-arts.

1. Introduction

Drones, or general unmanned aerial vehicles (UAVs), equipped with cameras have been fast deployed to a wide range of applications, such as video surveillance for crowd control [45] and public safety [22]. In recent years, many massive stampedes have taken place around the world that claimed many victims, making the automatic density map estimation, counting and tracking in crowds on drones important tasks, which draw great attention from the computer vision community.

Despite significant progress, crowd counting and tracking algorithms still have room for improvement to deal with drone-captured videos due to various challenges, such as view point and scale variations, background clutter, and small scales. Developing and evaluating these algorithms for drones are impeded by the lack of publicly available large-scale benchmarks. Some recent efforts [40, 44, 10, 41, 5, 35] have devoted to construct datasets for crowd counting. However, the majority of them focus on crowd counting with still images or inconsistent frames by surveillance cameras, due to difficulties in data collection and annotation for drone-based crowd counting and tracking.

To fill this gap, we collect a large-scale drone-based dataset for density map estimation, crowd localization and tracking. Our DroneCrowd dataset consists of 112 video clips formed by total 33,600 frames, captured by various drone-mounted cameras, in 70 different scenarios across 4 different cities in China (*i.e.*, Tianjin, Guangzhou, Daqing, and Hong Kong). These video clips are annotated with more than 4.8 million head annotations and several video-level attributes. To the best of our knowledge, this is the largest and most thoroughly annotated density map estimation, localization, and tracking dataset to date, see Table 1.

To handle this challenging dataset, we design a Space-Time Neighbor-Aware Network (STNNet) as a strong baseline, which solves the density map estimation, localization, and tracking simultaneously. Specifically, the proposed STNNet is formed by four modules, *i.e.*, the feature extraction subnetwork, followed by the density map estimation heads, the localization, and the association subnets. The feature extraction subnetwork first uses two-branch CNNs to extract multi-scale features, and then computes the correlations between the extracted features in consecutive two frames to exploit the temporal relations. Using density map estimation heads, we can estimate the density of objects in video frames to perform crowd counting. Inspired by object detection [25, 26, 42], we introduce the localization subnet, formed by the classification and regression branches, to output accurate locations of targets in each individual frames.

Table 1. Comparison between the DroneCrowd dataset and existing datasets.

Dataset	Type	Trajectory	Resolution	Frames	Max count	Min count	Ave count	Total count	Year
UCF_CC_50 [9]	image		-	50	4,543	94	1,279.5	63,974	2013
Shanghaitech A [44]	image		-	482	3,139	33	501.4	241,677	2016
Shanghaitech B [44]	image		768 × 1024	716	578	9	123.6	88,488	2016
AHU-Crowd [7]	image		576 × 720	107	2,201	58	420.6	45,000	2016
CARPK [6]	image		1280 × 720	1,448	188	1	62.0	89,777	2017
Smart-City [41]	image		1920 × 1080	50	14	1	7.4	369	2018
UCF-QNRF [10]	image		-	1,535	12,865	49	815.4	1,251,642	2018
NWPU [35]	image		2191 × 3209	5,109	20,033	0	418.0	2,133,375	2020
UCSD [3]	video		158 × 238	2,000	46	11	24.9	49,885	2008
Mall [19]	video		640 × 480	2,000	53	13	31.2	62,316	2013
WorldExpo [40]	video		576 × 720	3,980	253	1	50.2	199,923	2015
FDST [5]	video		1920 × 1080	15,000	57	9	26.7	394,081	2019
DroneCrowd	video	✓	1920 × 1080	33,600	455	25	144.8	4,864,280	2021

To exploit the temporal consistency, the association subnet is designed to predict motion offests of targets in consecutive frames for tracking. Besides, we develop the neighboring context loss by integrating spatial-temporal context of neighboring targets to guide the training of association subnet. Specifically, the neighboring context loss penalizes large displacements of the relative positions of adjacent objects in temporal domain, and guides the association subnet to generate accurate motion offsets. The whole network is trained in an end-to-end manner with the multi-task loss and Adam optimizer [12]. After that, multi-object tracking methods [24, 1] are used to predict long trajectories of targets. Compared with 12 state-of-the-art algorithms, extensive experiments on our DroneCrowd dataset demonstrate the effectiveness of the proposed STNNet method for density map estimation, crowd localization and tracking tasks.

Contributions. (1) We collect a large-scale drone captured dataset for density map estimation, localization, and tracking in dense crowd, which significantly surpasses existing datasets in terms of data type and volume, annotation quality, and difficulty. (2) We propose a space-time neighbor-aware network to solve the density map estimation, localization and tracking tasks simultaneously. (3) To exploit the spatial-temporal context, we design the neighboring context loss to penalize large displacements of the relative positions of adjacent objects in temporal domain for network training.

2. Related Work

Existing datasets. To date, there only exists a handful of crowd counting, crowd localization, or crowd tracking datasets. UCF_CC_50 [9] is formed by 50 images containing 64,000 annotated humans, with the head counts ranging from 94 to 4,543. Shanghaitech [44] includes 1,198 images with a total number of 330,165 labeled people. Recently, UCF-QNRF [10] is released with 1,535 images and 1.25 million annotated people's heads in various scenarios. Hsieh et al. [6] present a drone-based car counting dataset,

which approximately contains $90K$ cars captured in different parking lots. Recently, Wang et al. [35] collect a large-scale congested crowd counting and localization dataset, which includes more than $5K$ images and 2 million annotated heads with points and boxes. However, these datasets are still limited in sizes and scenarios covered.

To evaluate counting algorithms in videos, Chan et al. [3] present the UCSD counting dataset including low density crowd and counting difficulty. Similar to the UCSD dataset, Mall [19] is collected by the surveillance camera in a single location. Zhang et al. [40] present the WorldExpo dataset with 3,980 annotated frames in total, which is captured in 108 different scenes during 2010 Shanghai World-Expo. Fang et al. [5] collect a video dataset with $15K$ frames and $394K$ annotated heads captured from 13 different scenes. In contrast to the aforementioned datasets, our DroneCrowd dataset is a large-scale drone-captured dataset for density map estimation, crowd localization and tracking, which consists of 112 sequences with more than 4.8 million head annotations on $20,800$ people trajectories.

Crowd counting and density map estimation. Modern crowd counting methods [14, 44, 29, 2, 15, 17, 20, 34] formulate crowding counting as density map estimation. Lempitsky and Zisserman [14] learn to infer the density estimation by a minimization of a regularized risk quadratic cost function. Zhang et al. [44] use the multi-column CNN network to estimate the crowd density map, which learns the features for different head sizes by each column CNN. Sam et al. [29] develop the switching CNN model to handle the variations of crowd density. Cao et al. [2] propose an encoder-decoder network, where the encoder extracts multi-scale features with scale aggregation and the decoder generates high-resolution density maps using transposed convolutions. Li et al. [15] employ dilated convolution layers to enlarge receptive fields and extract deeper features without losing resolutions. Liu et al. [17] adaptively encodes the scale of the contextual information for accurate crowd
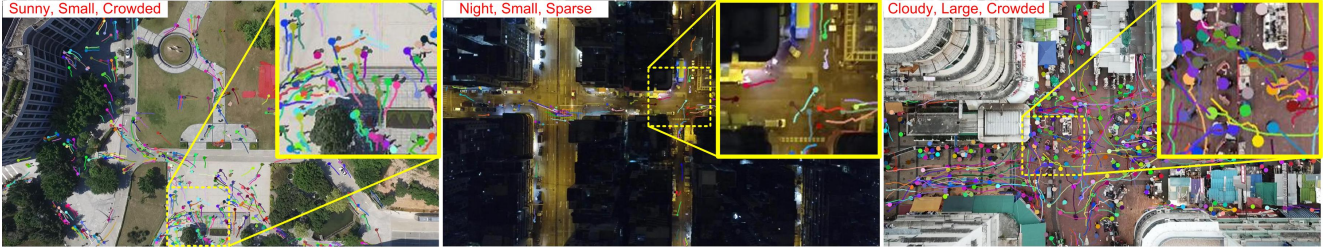
Figure 1. Some annotated example frames in the DroneCrowd dataset. Different color indicates different object instance and the corresponding trajectory. The video-level attributes are presented on the top-left corner in each video frame.

density prediction. In [16], the physically-inspired temporal consistency constraints are considered in the network to handle the viewpoint changes by drones. Besides, Luo *et al.* [20] propose the hybrid graph neural network to capture dependencies among multi-scale counting and localization features. To avoid hurting the generalization bound of a model, Wang *et al.* [34] propose the optimal transport to measure the similarity between the normalized predicted and ground-truth density maps.

In terms of crowd counting in videos, spatio-temporal information is critical to improve the counting accuracy. Xiong *et al.* [38] design a convolutional LSTM model to fully capture both spatial and temporal dependencies for crowd counting. Zhang *et al.* [43] combine fully convolutional neural networks and LSTM by residual learning to perform vehicle counting. Liu *et al.* [18] first compute people flows between consecutive frames and then estimate the densities from these flows. Different from existing methods, our STNNet can output both crowd density and target locations in crowds using the proposed localization subnet.

Crowd localization and tracking. Besides crowd counting, crowd localization and tracking are also important tasks in safety control scenarios. Rodriguez *et al.* [27] formulate an energy minimization framework by jointly optimizing the density and location, with the temporal-spatial constraints of person tracks in video. Ma *et al.* [21] first obtain local counts from sliding windows over the density map and then use integer programming to recover the locations of individual objects. In [10], crowd counting and localization tasks are simultaneously solved with a CNN model trained by a composition loss. In contrast, our method captures context information among neighbouring targets and estimate motion offsets of targets between consecutive frames, trained by the proposed neighboring context loss.

3. DroneCrowd Dataset

3.1. Data Collection and Annotation

Our DroneCrowd dataset is captured by drone-mounted cameras (*i.e.*, DJI Phantom 4, Phantom 4 Pro and Mavic), covering a wide range of scenarios, *e.g.*, campus, street,

Table 2. Statistics of each attribute in DroneCrowd.

Attribute	Min count	Max count	Avg count	Frames
Small	26	455	143.8	20, 700
Large	25	436	146.3	12, 900
Cloudy	25	436	144.9	18, 300
Sunny	26	455	153.3	12, 300
Night	40	167	109.1	3, 000
Crowded	129	455	225.6	13, 500
Sparse	25	170	90.5	20, 100

park, parking lot, playground and plaza[1]. The videos are recorded at 25 frames per seconds (FPS) with a resolution of 1920×1080 pixels. As presented in Figure 2 (a) and (b), the maximal and minimal numbers of people in each video frame are 455 and 25 respectively, and the average number of objects is 144.8. Moreover, more than 20 thousands of head trajectories of people are annotated with more than 4.8 million head points in individual frames of 112 video clips. Over 20 domain experts annotate and double-check the dataset using the vatic software [33] for more than two months. Figure 1 shows some frames of video clips with annotated trajectories of people heads.

We divide DroneCrowd into the `training` and `testing` sets, with 82 and 30 sequences, respectively. Notably, training videos are taken at different locations from testing videos to reduce the chances of algorithms to overfit to particular scenes. It contains video sequences with large variations in scale, viewpoint, and background clutters. To analyze the performance of algorithms thoroughly, we define three video-level attributes of the dataset, described as follows. (1) *Illumination*: under different illumination conditions, the objects are assumed to be different in appearance. Three categories of illumination conditions are considered in our dataset, including *Cloudy*, *Sunny*, and *Night*. (2) *Scale* indicates the size of objects. Two categories of scales are defined, including *Large* (the diameter of objects > 15 pixels) and *Small* (the diameter of objects ≤ 15 pix-

---

[1]We strictly comply with local laws and regulations in China when using unmanned aircraft/drones, and avoid restricted areas to capture videos. Since the scales of objects are extremely small, no identity information such as faces and vehicle plates could be retrieved. After careful check, we confirm that all data in our dataset would not leak any personal information.
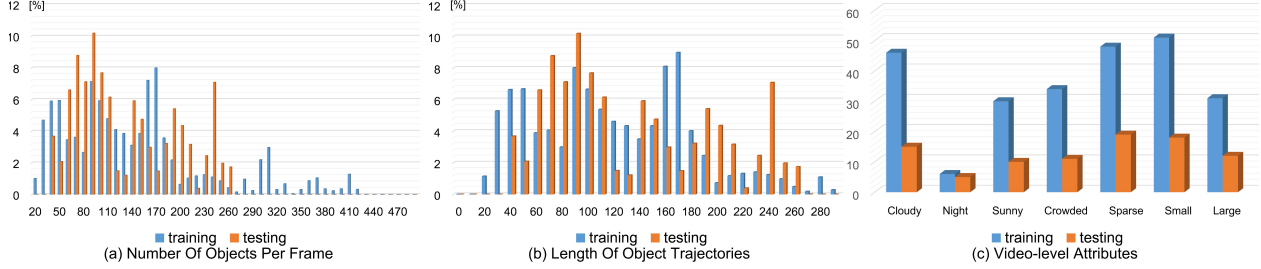
Figure 2. (a) The distribution of the number of objects per frame, (b) the distribution of the length of object trajectories, and (c) the attribute statistics, of the `training` and `testing` sets in the DroneCrowd dataset.

els). (3) *Density* indicates the number of objects in each frame. Based on the average number of objects in each frame, we divide the dataset into two density levels, *i.e.*, *Crowded* (with the number of objects in each frame larger than 150), and *Sparse* (with the number of objects in each frame less than 150). The statistics on different attributes are shown in Figure 2 (c) and Table 2.

3.2. Evaluation Metrics and Protocols

Density map estimation. Following the previous works [40, 44, 10], the density map estimation task aims to compute per-pixel density at each location in the image, while preserving spatial information about distribution of people. We use the mean absolute error (MAE) and mean squared error (MSE) to evaluate the performance, *i.e.*, $\text{MAE} = \frac{1}{\sum_{i=1}^{M} N_i} \sum_{i=1}^{M} \sum_{j=1}^{N_i} |z_{i,j} - \hat{z}_{i,j}|$, and $\text{MSE} = \sqrt{\frac{1}{\sum_{i=1}^{M} N_i} \sum_{i=1}^{M} \sum_{j=1}^{N_i} |z_{i,j} - \hat{z}_{i,j}|^2}$, where M is the number of video clips, N_i is the number of frames in the i-th video. $z_{i,j}$ and $\hat{z}_{i,j}$ are the ground-truth and estimated number of people in the j-th frame of the i-th video clip, respectively. As stated in [44], MAE and MSE describe the accuracy and robustness of the estimation respectively.

Crowd localization. The goal of crowd localization is to detect the locations of all people in an image. Each evaluated crowd localization algorithm is required to output a series of detected points with confidence scores for each test image. The estimated locations determined by the confidence threshold are associated to the ground-truth locations using greedy method. Then, we compute the L-mAP at various distance thresholds $(1, 2, 3, \cdots, 25$ pixels) to evaluate the localization results. We also report the performance with three specific distance thresholds, *i.e.*, L-AP@10, L-AP@15, and L-AP@20 pixels. These criteria penalize missing detection of people as well as duplicate detections.

Crowd tracking. Crowd tracking requires an algorithm to recover the trajectories of people in video sequence, which is evaluated on the metric in [23]. Specifically, each tracker is required to output a series of head points with confidence scores and the corresponding identities. We sort the tracklets, formed by the locations with the same identity, based

on the average confidence of their detections. A tracklet is considered to be correct if the matched ratio between the predictions and ground-truth tracklets is larger than a threshold. We use 3 thresholds in evaluation, *i.e.*, 0.10, 0.15, and 0.20. The matching distance threshold between the predicted and ground-truth locations on the tracklets is set to 25 pixels. The T-mAP scores over different thresholds (*i.e.*, T-AP@0.10, T-AP@0.15, and T-AP@0.20) are used to measure the performance.

4. Our Method

Our STNNet sequentially takes a pair of frames as input, and outputs the density maps, the locations, and the motion offsets of objects in these two frames, see Figure 3. After that, the association method is used to generate long trajectories of objects in videos.

4.1. Network Architecture

As shown in Figure 3, the Siamese feature extraction subnetwork in our STNNet is constructed on the first 4 groups of convolution layers in the parameters shared two-branch VGG-16 network [31] to extract multi-scale features. Inspired by [28], the U-Net style architecture is used to fuse multi-scale features for prediction. Using density map estimation heads, we can determine the number of targets based on multi-scale features. Meanwhile, the correlation operation [11] is conducted on the extracted features to exploit the temporal coherence at different stage. In addition, the localization and association subnets are introduced to predict the locations of target points and the corresponding motion offsets, which are described as follows.

Localization subnet. The localization subnet consists of the classification and regression branches. To generate accurate locations of objects, we tile the object proposal in each pixel. The classification branch aims to predict the probability of each proposal to be an object, and the regression branch aims to generate the accurate locations of the positive proposals. As shown in Figure 4, we fuse multi-scale feature maps (*i.e.*, f_1, f_2, f_3) with both channel and spatial attention [36] for each branch. After that, we resize multi-scale feature maps and then output the fused classifi-
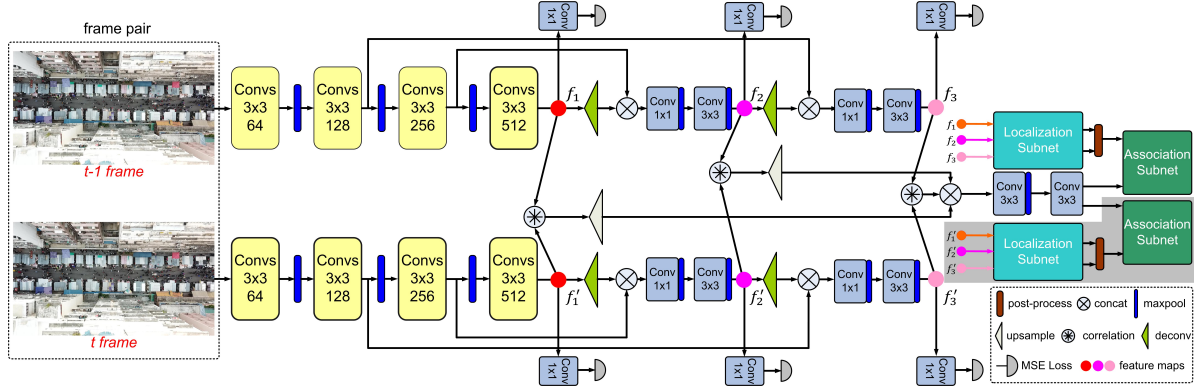
Figure 3. The architecture of our STNNet. The yellow rectangles indicate the convolution groups in the VGG-16 backbone. The blue and green rectangles indicate the localization subnet (see Figure 4) and association subnet (see Figure 5) respectively. Colourful circles indicate feature maps at different stage. Note that the modules in the grey regions are removed in the testing phase.
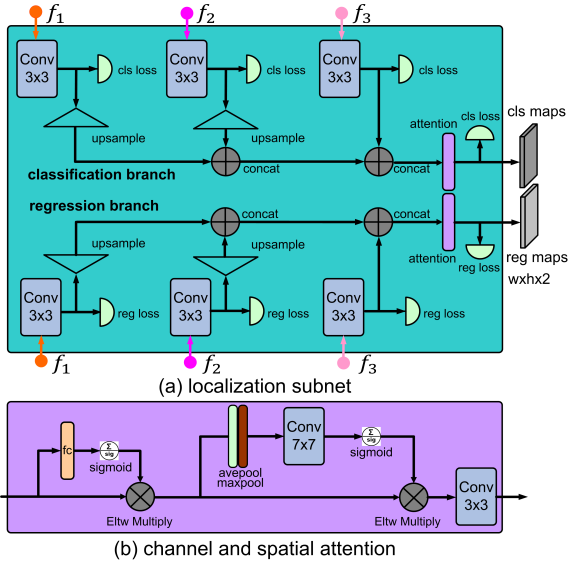

Figure 4. (a) the localization subnet based on (b) channel and spatial attention.

cation and regression maps. The classification map denotes the probability that each proposal contains an object and the regression map contains the regressed offsets of the positive proposals. Finally, we perform non-maximal suppression to predict the accurate object locations.

Association subnet. As mentioned above, we introduce the association subnet to predict the motion offsets of each object to complete the tracking task. As shown in Figure 5(a), given the M top scored post-processed object proposals generated by the localization subnet in the $(t-1)$-th frame and the fused multi-scale correlation features, we use 3 stacked PointConv [37] and multi-layer perceptron (MLP) operations to construct the association subnet to generate the motion offsets in a circle, *i.e.*, from the $(t-1)$-th frame to t-th frame and vice versa. Note that, only the nearest β

points are considered in each PointConv operation.

4.2. Multi-Task Loss Function

We use the multi-task loss to guide the training of our STNNet method, which consists of three terms, including the neighboring context loss $L_{\text{ass}}(\hat{P}_k, P_k^*, O_k)$, the localization loss $L_{\text{loc}}(\hat{C}_k, C_k^*, \hat{R}_k, R_k^*)$, and the density loss $L_{\text{den}}(\hat{\Phi}_k, \Phi_k^*)$, *i.e.*,

$$
\begin{aligned}
\mathcal{L} &= \frac{1}{K} \sum_{k=1}^{K} \left(L_{\text{den}}(\hat{\Phi}_k, \Phi_k^*) + L_{\text{loc}}(\hat{C}_k, C_k^*, \hat{R}_k, R_k^*) \right. \\
&\quad \left. + \sum_k^K L_{\text{ass}}(\hat{P}_k, P_k^*, O_k) \right),
\end{aligned}
\tag{1}
$$

where k is the index of the batch, K is the batch size. $\hat{\Phi}_k$ and Φ_k^* are the predicted and ground-truth density maps. $\hat{C}_k$ and C_k^* are the predicted and ground-truth labels (*i.e.*, objects or background) of the object proposals, $\hat{R}_k$ and R_k^* are the predicted and ground-truth offsets of the object proposals. $\hat{P}_k$ and P_k^* are the predicted and ground-truth locations of the objects, and O_k is the prediction motion offsets of the objects. In the following sections, we would like to discuss each loss term in details.

Density loss. Inspired by [44], we use the pixel-wise Euclidean loss for the density loss. The geometry-adaptive Gaussian kernel method is used to generate the ground-truth density map Φ_k^*. The density loss term is computed as

$$
\begin{aligned}
L_{\text{den}}(\hat{\Phi}_k, \Phi_k^*) &= \frac{1}{2 \cdot L} \sum_{t=1}^{2} \sum_{l=1}^{L} \sum_{i=1}^{W_l} \\
&\quad \sum_{j=1}^{H_l} \omega_l \cdot \|\hat{\phi}_{k,t}(i,j,l) - \phi_{k,t}^*(i,j,l)\|_2^2,
\end{aligned}
\tag{2}
$$

where $\hat{\phi}_{k,t}(i,j,l)$ and $\phi_{k,t}^*(i,j,l)$ are the values of the predicted and ground-truth density maps at (i,j) of layer l at time t of the k-th batch, and ω_l is the parameter to balance the influence of each layer.

Localization loss. Motivated by object detection [25, 26, 42], the localization loss is formed by the classification loss and regression loss. We tile the point proposals on each
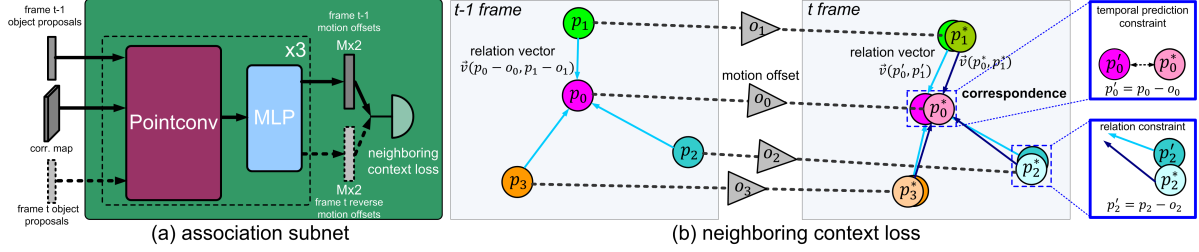
Figure 5. (a) the association subnet using (b) the neighboring context loss. Notably, the dashed modules in (a) are only used in the training phase. For clarity, we only display the calculation of the terms from time $t-1$ to time t in the neighboring context loss.

pixel and match the proposal to the ground-truth points. If the proposal locates in the neighboring regions of the ground-truth points, we assign it to be the positive proposal (i.e., $s_k(i, j, l) = 1$ for the proposal at (i, j) in the l-th layer in the k-th batch); otherwise the background (i.e., $s_k(i, j, l) = 0$). Thus, the localization loss is computed as

$$
\begin{aligned}
&L_{\text{loc}}(\hat{C}_k, C_k^*, \hat{R}_k, R_k^*) \\
&= \frac{1}{L} \sum_{l=1}^{L} \sum_{i=1}^{W_l} \sum_{j=1}^{H_l} \big(L_{\text{cls}}(\hat{c}_k(i,j,l), c_k^*(i,j,l)), \\
&+ s_k(i,j,l) \cdot L_{\text{reg}}(\hat{r}_k(i,j,l), r_k^*(i,j,l)) \big),
\end{aligned}
\tag{3}
$$

where $\hat{c}_k(i, j, l)$ and $c_k^*(i, j, l)$ are the predicted and ground-truth labels at (i, j) of layer l. $\hat{r}_k(i, j, l)$ and $r_k^*(i, j, l)$ are the predicted and ground-truth offsets at (i, j) of layer l. We use the log loss to compute L_{cls}, and the squared loss to compute L_{reg}. Notably, the regression loss L_{reg} is only activated for the positive proposals.

Neighboring context loss. In crowded scenes, the objects are generally clustered in a small region and usually share similar motion patterns in consecutive frames. To exploit the motion consistency of neighboring objects, we design a neighboring context loss, which is formed by two parts, i.e., the temporal prediction constraint, and the relation constraint, see Figure 5(b).

Specifically, the temporal prediction constraint enforces the proposals in the consecutive frames projected by the predicted motion offsets to approach the ground-truth points. Let $p_{i,t-1}$ be the location of the i-th proposal at time $t-1$, $p_{j,t-1} \in \mathcal{N}_{p_{i,t-1}}$ be the object location in the neighboring region of the proposal at $p_{i,t-1}$, and $o_{i,t-1}$ be the predicted offset corresponding to the proposal at $p_{i,t-1}$ from time $t-1$ to t. Thus, the temporal prediction constraint aims to minimize the ℓ_1-norm of the differences, i.e., $\|(p_{i,t-1} - o_{i,t-1}) - p_{i,t}^*\|_1$. Meanwhile, the relation constraint enforces the relation vectors between the target and neighboring objects to approach to the relation vectors of their corresponding associated ground-truth points. Let $\vec{v}(p_{i,t-1} - o_{i,t-1}, p_{j,t-1} - o_{j,t-1})$ be the relation vector[2] between the target and neighboring objects projected to the second frame, and $\vec{v}(p_{i,t}^*, p_{j,t}^*)$

---

[2]The relation vector is computed as $\vec{v}(p_{i,t-1} - o_{i,t-1}, p_{j,t-1} - o_{j,t-1}) = (p_{j,t-1} - o_{j,t-1}) - (p_{i,t-1} - o_{i,t-1})$.

be the relation vector between the ground-truth points at $p_{i,t}^*$ and $p_{j,t}^*$. Thus, the relation constraint aims to minimize $\sum_{p_{j,t-1} \in \mathcal{N}_{p_{i,t-1}}} \|\vec{v}(p_{i,t-1} - o_{i,t-1}, p_{j,t-1} - o_{j,t-1}) - \vec{v}(p_{i,t}^*, p_{j,t}^*)\|_1$. The cycle strategy is used to compute the neighboring context loss, i.e.,

$$
\begin{aligned}
&L_{\text{ass}}(\hat{P}_k, P_k^*, O_k) = \frac{1}{2 \cdot M} \sum_{i=1}^{M} \big(\|p_{i,t-1}' - p_{i,t}^*\|_1 \\
&+ \sum_{p_{j,t-1} \in \mathcal{N}_{p_{i,t-1}}} \|\vec{v}(p_{i,t-1}', p_{j,t-1}') - \vec{v}(p_{i,t}^*, p_{j,t}^*)\|_1 \\
&+ \|p_{i,t}' - p_{i,t-1}^*\|_1 \\
&+ \sum_{p_{j,t} \in \mathcal{N}_{p_{i,t}}} \|\vec{v}(p_{i,t}', p_{j,t}') - \vec{v}(p_{i,t-1}^*, p_{j,t-1}^*)\|_1 \big),
\end{aligned}
\tag{4}
$$

where $p_{i,t}' = p_{i,t} - o_{i,t}$ and $p_{j,t}' = p_{j,t} - o_{j,t}$ are the projected targets.

4.3. Optimization

To increase diversity in training data, we randomly flip and crop the training images. Due to limited computation resources, we equally divide each frame into 2×2 patches, and use the divided 4 patches with the resolution of 960×540 for training. For the Pointconv layer, we use $\beta = 8$ nearest points to capture the context information. In (2), the pre-set weights ω_l are set to $\{2.0, 0.5, 0.05\}$. The matching threshold between the proposals and ground-truth points is set to 10 pixels. Meanwhile, the threshold used to determine the neighboring regions of pixels $\mathcal{N}_{p_i}$ is set to 50 pixels. The total number of proposal objects M is set to 128. In addition, we set the batch size $K = 4$ in the training phase. **Two-stage training.** We use the two stage strategy to train our network. For the first stage, we remove the association subnet and train the network to generate accurate density map and object proposals. After that, we fixed the parameters in the density map estimation heads, and add the association subnet to fine-tune the whole network. We use the Adam optimization algorithm [12] with the learning rate 10^{-6} in both stages.

5. Experiment

As discussed above, we conduct the experiment on our DroneCrowd for crowd counting, localization and tracking. We report the density map estimation results and speeds of STNNet and 12 existing methods. Meanwhile, the ablation
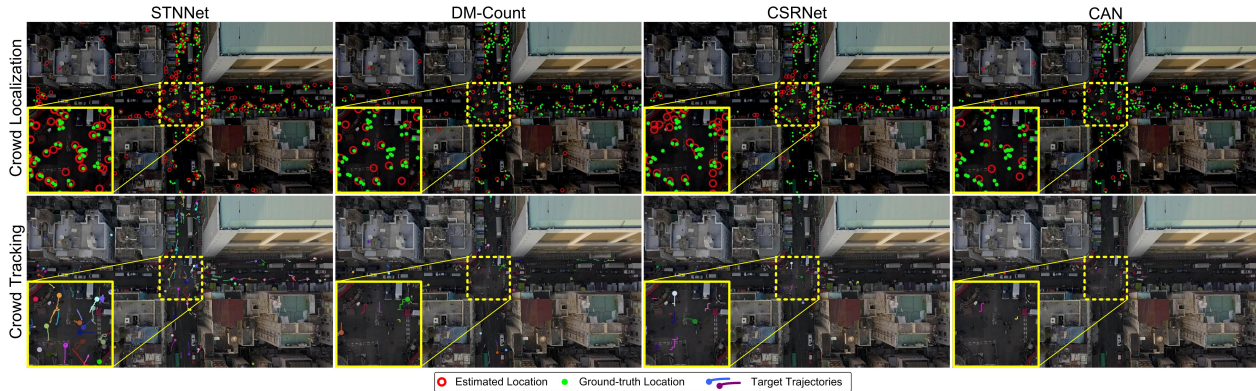
Figure 6. Qualitative results of DM-Count [34], CSRNet [15], CAN [17], and our STNNet on DroneCrowd. Best view in color version.

study is conducted to verify the effectiveness of important components in our method. Besides, some visual results are shown in Figure 6.

Density map estimation. As shown in Table 3, our STNNet performs favorably against the state-of-the-art methods, with an improvement of 2.6 MAE and 8.3 MSE in comparison to the second best DM-Count [34] in the overall `testing` set. It indicates that our method generates more accurate and robust density maps in different scenarios. To further analyze the results, we report the performance on several subsets based on the video-level attributes (see Section 3.1). LCFCN [13] and AMDCN [4] perform not well in the *Crowd* subset, producing the two worst MAE and MSE scores. This is maybe because LCFCN [13] uses a loss function to encourage the network to output a segmentation blob for each object in crowd counting. However, in drone-captured scenarios, each object may contain only few pixels, making it difficult to separate objects accurately. AMDCN [4] uses multiple columns of large dilation convolution operations, which inevitably integrates considerable background noise, affecting the accuracy in density map estimation. In contrast, MCNN [44] uses multi-column CNNs to learn the features adaptive to variations in object size due to perspective effect or image resolution, resulting in better performance. CAN [17] achieves the best performance in both *Cloudy* and *Crowded* subsets by exploiting multi-scale contextual information in density maps. DM-Count [34] obtains the best MAE and MSE scores in the *Sunny* subset without imposing Gaussians to annotations. Our STNNet achieves the best result in other four subsets, which demonstrates the effectiveness and importance of exploiting multi-scale features in density map estimation.

Furthermore, to study the effectiveness of the localization subnet in STNNet for density map estimation, we construct a variant of STNNet, *i.e.*, STNNet (w/o loc), which removes the localization subnet from STNNet. As shown in Table 3, our STNNet achieves better results than STNNet (w/o loc) by decreasing 2.8 MAE score and 3.5 MSE score,

which validates the importance of the localization subnet.

Crowd localization. As presented in Table 4, we compare the localization results of 4 methods with top density estimation results (*i.e.*, MCNN [44], CSRNet [15], CAN [17], and DM-Count [34]) and our STNNet variants, *i.e.*, STNNet (w/o loc), STNNet (w/o ass) and STNNet (w/o cyc). STNNet (w/o loc) denotes the method that removes both the association and localization subnets from STNNet, STNNet (w/o ass) denotes the method that removes the association subnet from STNNet, and STNNet (w/o cyc) denotes the method that only considers the forward motion offsets in neighboring context loss computation. Meanwhile, for the density map estimation based methods such as MCNN, CSRNet, CAN, DM-Count, and STNNet (w/o loc), similar to [10], we post-process the predicted density maps to find local peaks using a preset threshold.

As shown in Table 4, we find that STNNet achieves the best accuracy with 40.45% L-mAP and surpasses the second best DM-Count [34] 22.28% L-mAP. It indicates that our method can generate more accurate localizations of each target. Compared to STNNet (w/o cyc), STNNet improves the localization accuracy by 0.22%, which shows the effectiveness of cycle strategy in the neighboring context loss for the localization task. Without the association subnet, the L-mAP score decreases 0.68% (40.45% of STNNet *vs.* 39.77%), indicating that temporal coherence facilitates improve the localization accuracy. If we remove both association and localization subnets, the L-mAP score decreases more than 8%. It demonstrates that the localization subnet enforces the network to focus on more discriminative features to localize people's heads.

Crowd tracking. For object tracking, two association methods, *i.e.*, the min-cost flow method [24] and the social-LSTM method [1], are used to generate long trajectories of objects. To validate the effectiveness of STNNet for crowd tracking, we compare it to several methods including MCNN, CSRNet, CAN, DM-Count, STNNet (w/o loc), STNNet (w/o ass), STNNet (w/o cyc) and STNNet. It is

Table 3. Estimation errors of density maps on DroneCrowd.

Method	Speed FPS	Overall MAE	Overall MSE	Large MAE	Large MSE	Small MAE	Small MSE	Cloudy MAE	Cloudy MSE	Sunny MAE	Sunny MSE	Night MAE	Night MSE	Crowded MAE	Crowded MSE	Sparse MAE	Sparse MSE
MCNN [44]	**28.98**	34.7	42.5	36.8	44.1	31.7	40.1	21.0	27.5	39.0	43.9	67.2	68.7	29.5	35.3	37.7	46.2
C-MTL [32]	2.31	56.7	65.9	53.5	63.2	61.5	69.7	59.5	66.9	56.6	67.8	48.2	58.3	81.6	88.7	42.2	47.9
MSCNN [39]	1.76	58.0	75.2	58.4	77.9	57.5	71.1	64.5	85.8	53.8	65.5	46.8	57.3	91.4	106.4	38.7	48.8
LCFCN [13]	3.08	136.9	150.6	126.3	140.3	152.8	164.8	147.1	160.3	137.1	151.7	105.6	113.8	208.5	211.1	95.4	110.0
SwitchCNN [29]	0.01	66.5	77.8	61.5	74.2	74.0	83.0	56.0	63.4	69.0	80.9	92.8	105.8	67.7	79.8	65.7	76.7
ACSCP [30]	1.58	48.1	60.2	57.0	70.6	34.8	39.7	42.5	46.4	37.3	44.3	86.6	106.6	36.0	41.9	55.1	68.5
AMDCN [4]	0.16	165.6	167.7	166.7	168.9	163.8	165.9	160.5	162.3	174.8	177.1	162.3	164.3	165.5	167.7	165.6	167.8
StackPooling [8]	0.73	68.8	77.2	68.7	77.1	68.8	77.3	66.5	75.9	74.0	83.4	65.2	67.4	95.7	101.1	53.1	59.1
DA-Net [46]	2.52	36.5	47.3	41.5	54.7	28.9	33.1	45.4	58.6	26.5	31.3	29.5	34.0	56.5	68.3	24.9	28.7
CSRNet [15]	3.92	19.8	25.6	17.8	25.4	22.9	25.8	12.8	16.6	19.1	22.5	42.3	45.8	20.2	24.0	19.6	26.5
CAN [17]	7.12	22.1	33.4	18.9	26.7	26.9	41.5	**11.2**	**14.9**	14.8	17.5	69.4	73.6	**14.4**	**17.9**	26.6	39.7
DM-Count [34]	10.04	18.4	27.0	19.2	29.6	17.2	22.4	11.4	16.3	**12.6**	**15.2**	51.1	55.7	17.6	21.8	18.9	29.6
STNNet (w/o loc)	3.65	18.6	22.2	17.1	20.5	21.0	24.6	14.7	19.9	21.4	23.3	24.7	26.3	24.2	27.3	15.4	18.7
STNNet	3.41	**15.8**	**18.7**	**16.0**	**18.4**	**15.6**	**19.2**	14.1	17.2	19.9	22.5	**12.9**	**14.4**	18.5	21.6	**14.3**	**16.9**

Table 4. Localization accuracy on DroneCrowd.

Methods	L-mAP	L-AP@10	L-AP@15	L-AP@20
MCNN [44]	9.05%	9.81%	11.81%	12.83%
CAN [17]	11.12%	8.94%	15.22%	18.27%
CSRNet [15]	14.40%	15.13%	19.77%	21.16%
DM-Count [34]	18.17%	17.90%	25.32%	27.59%
STNNet (w/o loc)	32.19%	33.88%	39.56%	43.22%
STNNet (w/o ass)	39.77%	42.06%	50.00%	54.88%
STNNet (w/o rel)	40.00%	42.29%	50.31%	55.11%
STNNet (w/o cyc)	40.23%	42.57%	50.64%	55.42%
STNNet	**40.45**%	**42.75**%	**50.98**%	**55.77**%

Table 5. Tracking accuracy on DroneCrowd in terms of min-cost flow/social-LSTM.

Methods	T-mAP	T-AP@0.10	T-AP@0.15	T-AP@0.20
MCNN [44]	9.16/8.96	11.47/10.45	9.65/9.91	6.36/6.51
CAN [17]	4.39/4.13	6.97/5.48	4.72/5.26	1.48/1.65
CSRNet [15]	12.15/11.66	17.34/14.63	12.85/13.74	6.26/6.16
DM-Count [34]	17.01/16.54	22.38/19.72	18.34/19.13	10.29/10.77
STNNet (w/o loc)	28.72/28.55	32.52/32.50	30.84/30.65	22.80/22.51
STNNet (w/o ass)	31.44/30.90	34.59/34.08	32.94/32.32	26.77/26.30
STNNet (w/o rel)	32.26/**31.60**	35.20/34.78	33.78/**33.12**	27.80/26.89
STNNet (w/o cyc)	**32.50**/31.44	**35.45**/34.53	**33.99**/32.79	**28.05**/**26.99**
STNNet	32.32/31.58	35.29/**34.82**	33.78/33.00	27.90/26.92

worth mentioning that STNNet (w/o loc) performs crowd tracking based on the localized points from density maps, similar to MCNN, CSRNet, CAN, and DM-Count. Without predicting motion offsets, STNNet (w/o ass) directly associates the targets from the localization results. STNNet (w/o cyc) and STNNet first connect short tracklets in two consecutive frames based on the predicted offsets, and then generate long trajectories using the same data association methods [24, 1].

From Table 5, we notice that STNNet achieves 32.50% T-mAP score, which is 15.49% higher than the second best DM-Count. Meanwhile, STNNet (w/o cyc) produces 0.11% higher T-mAP score than our method. STNNet (w/o ass) produces inferior results than STNNet, *i.e.*, 31.44% vs. 32.32%. The T-mAP score of STNNet (w/o loc) decreases 3.60% compared to STNNet (w/o ass). These results indicate that association and localization subnets are critical in crowd tracking. However, these results are still far from satisfactory. Besides, we find that the method using social-LSTM [1] performs comparably with that using min-cost flow [24]. It indicates that it is possible to predict the motion patterns of objects based on the observed trajectories. In summary, our DroneCrowd dataset is extremely challenging for crowd tracking and much effort is needed to develop more effective methods in real scenarios.

Effectiveness of neighboring context loss. To further demonstrate the effectiveness of the relation constraint in the neighboring context loss, we construct a variant STNNet (w/o rel) by removing the relation constraint in STNNet (w/o cyc). As shown in Table 4 and 5, STNNet (w/o rel) produces 40.00% and 32.26% L-mAP and T-mAP scores, respectively. STNNet (w/o cyc) improves 0.23% and 0.24% L-mAP and T-mAP scores compared with STNNet (w/o rel).

6. Conclusion

In this work, we propose the STNNet method to jointly solve density map estimation, localization, and tracking in drone-captured crowded scenes. Notably, we design the neighboring context loss to capture relations among neighboring targets in consecutive frames, which is effective for localization and tracking. To better evaluate the performances on drones, we collect and annotate a new dataset, DroneCrowd. To the best of our knowledge, it is the largest dataset to date in terms of annotated trajectories of heads for density map estimation, crowd localization, and tracking on drones. We hope the dataset and the proposed method can facilitate the research and development in crowd localization, tracking and counting on drones.

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Fei-Fei Li, and Silvio Savarese. Social LSTM: human trajectory prediction in crowded spaces. In *CVPR*, pages 961–971, 2016. 2, 7, 8

[2] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. Scale aggregation network for accurate and efficient crowd counting. In *ECCV*, pages 757–773, 2018. 2

[3] Antoni B. Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *CVPR*, 2008. 2

[4] Diptodip Deb and Jonathan Ventura. An aggregated multicolumn dilated convolution network for perspective-free counting. In *CVPRW*, pages 195–204, 2018. 7, 8

[5] Yanyan Fang, Biyun Zhan, Wandi Cai, Shenghua Gao, and Bo Hu. Locality-constrained spatial transformer network for video crowd counting. In *ICME*, pages 814–819, 2019. 1, 2

[6] Meng-Ru Hsieh, Yen-Liang Lin, and Winston H. Hsu. Drone-based object counting by spatially regularized regional proposal network. In *ICCV*, 2017. 2

[7] Yaocong Hu, Huan Chang, Fudong Nian, Yan Wang, and Teng Li. Dense crowd counting from still images with convolutional neural networks. *J. Visual Communication and Image Representation*, 38:530–539, 2016. 2

[8] Siyu Huang, Xi Li, Zhiqi Cheng, Zhongfei Zhang, and Alexander G. Hauptmann. Stacked pooling: Improving crowd counting by boosting scale invariance. *CoRR*, abs/1808.07456, 2018. 8

[9] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In *CVPR*, pages 2547–2554, 2013. 2

[10] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Máadeed, Nasir M. Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *ECCV*, pages 544–559, 2018. 1, 2, 3, 4, 7

[11] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, pages 1647–1655, 2017. 4

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 2, 6

[13] Issam H. Laradji, Negar Rostamzadeh, Pedro O. Pinheiro, David Vázquez, and Mark W. Schmidt. Where are the blobs: Counting by localization with point supervision. In *ECCV*, 2018. 7, 8

[14] Victor S. Lempitsky and Andrew Zisserman. Learning to count objects in images. In *NeurIPS*, pages 1324–1332, 2010. 2

[15] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *CVPR*, pages 1091–1100, 2018. 2, 7, 8

[16] Weizhe Liu, Krzysztof Lis, Mathieu Salzmann, and Pascal Fua. Geometric and physical constraints for drone-based head plane crowd density estimation. In *IROS*, pages 244–249, 2019. 3

[17] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *CVPR*, pages 5099–5108, 2019. 2, 7, 8

[18] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Estimating people flows to better count them in crowded scenes. In *ECCV*, volume 12360, pages 723–740, 2020. 3

[19] Chen Change Loy, Shaogang Gong, and Tao Xiang. From semi-supervised to transfer counting of crowds. In *ICCV*, pages 2256–2263, 2013. 2

[20] Ao Luo, Fan Yang, Xin Li, Dong Nie, Zhicheng Jiao, Shangchen Zhou, and Hong Cheng. Hybrid graph neural networks for crowd counting. In *AAAI*, pages 11693–11700, 2020. 2, 3

[21] Zheng Ma, Lei Yu, and Antoni B. Chan. Small instance detection by integer programming on object density maps. In *CVPR*, pages 3689–3697, 2015. 3

[22] Naser Hossein Motlagh, Miloud Bagaa, and Tarik Taleb. Uav-based iot platform: A crowd surveillance use case. *IEEE Communications Magazine*, 55(2):128–134, 2017. 1

[23] Eunbyung Park, Wei Liu, Olga Russakovsky, Jia Deng, Fei-Fei Li, and Alex Berg. Large Scale Visual Recognition Challenge 2017. http://image-net.org/challenges/LSVRC/2017. 4

[24] Hamed Pirsiavash, Deva Ramanan, and Charless C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, pages 1201–1208, 2011. 2, 7, 8

[25] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *NeurIPS*, pages 91–99, 2015. 1, 5

[26] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *TPAMI*, 39(6):1137–1149, 2017. 1, 5

[27] Mikel Rodriguez, Ivan Laptev, Josef Sivic, and Jean-Yves Audibert. Density-aware person detection and tracking in crowds. In *ICCV*, pages 2423–2430, 2011. 3

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015. 4

[29] Deepak Babu Sam, Shiv Surya, and R. Venkatesh Babu. Switching convolutional neural network for crowd counting. In *CVPR*, pages 4031–4039, 2017. 2, 8

[30] Zan Shen, Yi Xu, Bingbing Ni, Minsi Wang, Jianguo Hu, and Xiaokang Yang. Crowd counting via adversarial cross-scale consistency pursuit. In *CVPR*, pages 5245–5254, 2018. 8

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4

[32] Vishwanath A. Sindagi and Vishal M. Patel. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *AVSS*, pages 1–6, 2017. 8

[33] Carl Vondrick, Donald J. Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation - A set

of best practices for high quality, economical video labeling. *IJCV*, 101(1):184–204, 2013. 3

[34] Boyu Wang, Huidong Liu, Dimitris Samaras, and Minh Hoai. Distribution matching for crowd counting. In *NeurIPS*, 2020. 2, 3, 7, 8

[35] Qi Wang, Junyu Gao, Wei Lin, and Xuelong Li. Nwpu-crowd: A large-scale benchmark for crowd counting. *CoRR*, abs/2001.03360, 2020. 1, 2

[36] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. In *ECCV*, pages 3–19, 2018. 4

[37] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, pages 9621–9630, 2019. 5

[38] Feng Xiong, Xingjian Shi, and Dit-Yan Yeung. Spatiotemporal modeling for crowd counting in videos. In *ICCV*, pages 5161–5169, 2017. 3

[39] Lingke Zeng, Xiangmin Xu, Bolun Cai, Suo Qiu, and Tong Zhang. Multi-scale convolutional neural networks for crowd counting. In *ICIP*, pages 465–469, 2017. 8

[40] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, pages 833–841, 2015. 1, 2, 4

[41] Lu Zhang, Miaojing Shi, and Qiaobo Chen. Crowd counting via scale-adaptive convolutional neural network. In *WACV*, pages 1113–1121, 2018. 1, 2

[42] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z. Li. Single-shot refinement neural network for object detection. In *CVPR*, pages 4203–4212, 2018. 1, 5

[43] Shanghang Zhang, Guanhang Wu, João P. Costeira, and José M. F. Moura. Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In *ICCV*, pages 3687–3696, 2017. 3

[44] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, pages 589–597, 2016. 1, 2, 4, 5, 7, 8

[45] Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *CVPR*, pages 2871–2878, 2012. 1

[46] Zhikang Zou, Xinxing Su, Xiaoye Qu, and Pan Zhou. Danet: Learning the fine-grained density distribution with deformation aggregation network. *Access*, 6:60745–60756, 2018. 8