Fusing Crowd Density Maps and Visual Object Trackers for People Tracking in Crowd Scenes

Weihong Ren[1,2,3], Di Kang[1], Yandong Tang[2], Antoni B. Chan[1]
[1]Department of Computer Science, City University of Hong Kong;
[2]State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences;
[3]University of Chinese Academy of Sciences
{weihonren2-c, dkang5-c}@my.cityu.edu.hk, ytang@sia.cn, abchan@cityu.edu.hk

Abstract

While visual tracking has been greatly improved over the recent years, crowd scenes remain particularly challenging for people tracking due to heavy occlusions, high crowd density, and significant appearance variation. To address these challenges, we first design a Sparse Kernelized Correlation Filter (S-KCF) to suppress target response variations caused by occlusions and illumination changes, and spurious responses due to similar distractor objects. We then propose a people tracking framework that fuses the S-KCF response map with an estimated crowd density map using a convolutional neural network (CNN), yielding a refined response map. To train the fusion CNN, we propose a two-stage strategy to gradually optimize the parameters. The first stage is to train a preliminary model in batch mode with image patches selected around the targets, and the second stage is to fine-tune the preliminary model using the real frame-by-frame tracking process. Our density fusion framework can significantly improves people tracking in crowd scenes, and can also be combined with other trackers to improve the tracking performance. We validate our framework on two crowd video datasets.

1. Introduction

In recent years, visual object tracking methods [1, 7, 12, 14, 17, 20] have focused on developing an effective appearance model in sparsely crowd scenes. Few works can effectively track people in crowded scenes due to heavy occlusions, high crowd density, and appearance variations. Though people tracking in crowded scenes is challenging, it is necessary for a wide range of applications including surveillance, event detection and group behaviour modelling. Rather than focus on individual people, crowd counting methods aim to predict the number of people in an image without explicitly detecting or tracking the people. One effective method is to estimate a crowd density map [18], where the sum over a region in the image corresponds to the number of people in that region. The crowd density map has been used for people counting [28, 30, 32], as well as for small object detection [21, 25, 28]. In this paper, we propose a framework that can effectively combine crowd density maps with generic visual object trackers to address the problem of people tracking in crowd scenes.

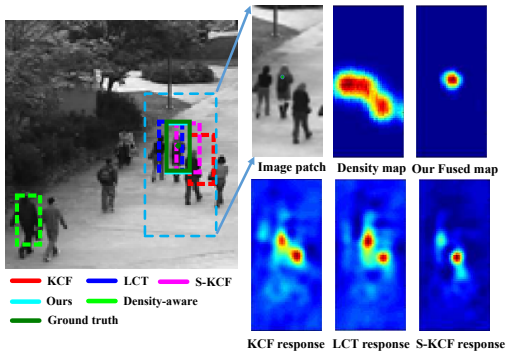

Figure 1. An example of people tracking in crowd scene using different trackers: KCF [14], S-KCF (ours), long-term correlation tracker (LCT) [20], density-aware [25], and our proposed fusion tracker. The response maps of the raw trackers (bottom-right) are greatly affected by the objects surrounding the target due to their similar appearances. Compared to KCF and LCT, our S-KCF partially suppresses the spurious responses using a sparsity constraint. Our density fusion framework combines the S-KCF response map with the crowd density map, and effectively suppresses the irrelevant responses and detects the target accurately (top-right).

Fig. 1 presents an example for people tracking using different trackers: KCF [14], S-KCF (ours), long-term correlation tracker (LCT) [20], density-aware [25], and our proposed fusion tracker. KCF and LCT are designed to address the problem of target appearance variation, but they do not perform well in crowd scenes due to the presence of many objects with similar appearance as the target, which results in drift. In addition, the target in crowd scenes is usually heavily occluded by other objects, and it is difficult to differentiate appearances between the target and other objects.

Although originally developed for counting, crowd den-

sity maps contain information about the location of people in the image, which makes them useful for people detection and tracking. Previous work [25] uses crowd density maps to improve people detection and tracking in crowd scenes. [25] first detects the head locations of all the people in a video through the optimization of a joint energy function that encourages the detected locations in the score maps to be consistent with the estimated density map. For tracking, [25] uses a nearest-neighbor rule to associate the head detections in individual frames into people tracks. However, one disadvantage of [25] is its simple association rule, which does not use an appearance model, and thus could fail in crowd scenes when people walk closely together.

In this paper, we address the problem of people tracking in crowded scenes by combining the visual tracker response map with the crowd density map to produce a new fused response map. The contributions of this paper are 3-fold:

1. To make the appearance-based tracker robust in crowded scenes, we propose a sparse KCF (S-KCF) tracker that uses a sparsity constraint on tracker response map to suppress variations caused by occlusions and illumination changes, and distractor objects.
2. We propose a density fusion framework, based on a CNN, that combines the S-KCF tracker response and the estimated crowd density map to improve people tracking in crowd scenes. Our density fusion framework can also be used with other appearance-based trackers to improve their accuracy in crowded scenes.
3. To train the fusion CNN, we propose a two-stage training strategy to gradually optimize the parameters in an end-to-end fashion. The first stage trains a preliminary model based on image patches selected around the targets, and the second stage fine-tunes the preliminary model on real tracking cases.

2. Related work

Here, we give a brief review of tracking methods closely related to our work. Comprehensive reviews on visual tracking methods can be found in [19, 26].

2.1. Correlation Filters Based Trackers

Correlation filters (CF) have been widely used in online visual tracking due to its high computational efficiency. [3] presented a Minimum Output Sum of Squared Error (MOSSE) filter for visual tracking on gray images. To make MOSSE a fast tracker, correlation is computed in an element-wise multiplication in Fast Fourier Transform (FFT), and the computational efficiency for MOSSE can reach several hundreds fps. [13] exploited the circulant structure of matrices for tracking-by-detection with kernels (CSK), which was later derived as the Kernelized Correlation Filter (KCF) tracker that supports multi-channel features [14]. To incorporate color information into CSK tracker, [9] proposed a low-dimensional adaptive extension for color distributions, and extended the learning scheme to multi-channel color features. [17] proposed a method to limit circular boundary effects of shifted examples, and used all possible patches densely extracted from training examples during the learning process. By embedding spatio-temporal context information into filter learning, [31] proposed a model to handle appearance changes of target. [20] proposed a Long-term Correlation Tracker (LCT) to address the problem of long-term visual tracking. The LCT tracker decomposes the tracking task into translation and scale estimation of target objects in conjunction with a online re-detection scheme, in order to address target drift during long-term tracking when the scene is not too crowded. [8] proposed a continuous convolution framework to integrate multi-resolution feature maps into correlation filter.

To make the CF response map more robust under appearance changes, [27] exploited the anisotropy of the CF response by using three sparsity-related *loss* functions: l_1, $l_1 l_2$, and $l_{2,1}$ norm functions. Their motivation is to tolerate large errors in the response map caused by appearance changes, but the prediction of the target location may be confused by the multi-peak response map. In contrast, our proposed S-KCF model uses a sparsity constraint on the *response map*, in order to suppress the irrelevant responses caused by other objects or occlusions, making it more robust in crowded scenes.

2.2. Crowd Density Maps

Crowd density maps were proposed in [18] to solve the object counting task. The crowd density at each pixel is predicted from the low-level image features, and the predicted region count is the sum of the density map over that region. Density map estimation is usually regarded as a general regression problem, and previous methods mainly focus on feature extraction and loss function design [11, 15, 18] to make the estimation robust to scene changes. Recently, CNNs have also been used [28, 30, 32], and have achieved good performance for a wide range of scenes. Besides people counting [28, 30, 32], crowd density maps can also be used for object detection [21, 25, 28].

2.3. Detection and Tracking in Crowd Scenes

To detect small instances in a scene, [21] first estimated the object density maps, and then proposed a joint object detection and counting framework using the density maps based on 2D integer programming. [22] proposed an alternative formulation of multi-target tracking as minimization of a continuous energy function. Besides the image evidence, their model also takes into account physical constraints, and thus it performs well for general crowded scenes. [24] adopted the Correlated Topic Model (CTM) [2] to track individual targets in high-density unstruc-

tured crowd scenes. In [25], a "density-aware" detection and tracking model was proposed that combines individual person detection with crowd density maps. [25] solves an energy minimization problem where the candidate detections should have high scores in the detection score map, while also encouraging the density map produced from the candidate detections to be similar to an estimated crowd density map. However, their tracking framework does not contain appearance models for each target, and uses simple nearest-neighbors correspondence between frames. In contrast to [25], our method fuses the crowd density map and response map of the visual tracker into a refined response map, and the target can be localized without solving an energy minimization problem. In our work, targets are associated between frames using the appearance model of the visual tracker.

3. Methodology

Our density fusion framework has three main parts: visual tracking model (S-KCF), crowd density estimation, and fusion neural network. Our framework is shown in Fig. 2. An image patch in the current frame is first cropped based on the predicted location of previous frame, and then passed to S-KCF to generate a response map. At the same time, a crowd density map of the image patch is estimated using a CNN. The response map, image patch and the corresponding density map are fused together using a fusion CNN, yielding a final fused response map. The predicted target location is the maximum value in the final response map. Using the predicted location, the S-KCF filters are updated.

3.1. Sparse Kernelized Correlation Filter

Kernelized Correlation Filter (KCF) has achieved great success in real-time tracking [13, 14]. By exploiting the circulant structure, [13] formulated visual tracking as a correlation filtering problem. Cyclic shifts of the target region in the initial frame are first used to train a KCF model. The KCF model is used to predict a response map on the next frame, and the location with the highest response value is the new target position. As tracking proceeds, the KCF model is gradually updated to adapt to background changes. However, for crowd scenes, the traditional KCF model may lose track easily due to many similar distractor objects, heavy occlusions and illumination changes, which all cause irrelevant responses around the target.

To make the KCF model robust to background changes, we propose to use an l_0 sparse term to regularize the *response map*, thus suppressing high responses due to distractors, occlusions and illumination changes. The typical correlation filter is a ridge regression problem

$$\min_{\mathbf{w}} \sum_i \left(f\left(\mathbf{x}_i\right) - y_i\right)^2 + \lambda \|\mathbf{w}\|_2^2, \tag{1}$$

where regression function $f\left(\mathbf{x}_i\right) = \mathbf{w}^T \phi\left(\mathbf{x}_i\right)$ is trained with a feature-space projector $\phi\left(\cdot\right)$, $\{y_i\}$ is the Gaussian-shaped response map, and $\lambda > 0$ is a parameter that controls overfitting. The Gaussian-shaped target response map is sparse, but the tracker usually generates multi-peak response maps due to distractors having similar appearance in crowd scenes. Hence, we add a sparsity regularization term on the response map in order to suppress the irrelevant response and retain the target response,

$$\min_{\mathbf{w}} \|\Phi^T \mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 + \tau \|\Phi^T \mathbf{w}\|_0, \tag{2}$$

where $\Phi = [\phi\left(x_1\right), \cdots, \phi\left(x_i\right)]$, $\mathbf{y} = [y_1, \cdots, y_i]^T$; $\tau > 0$ is a parameter that controls the sparsity of response map. During training, the sparse constraints make the tracker filters $\mathbf{w}$ consider the context changes, and push the weaker positive responses of the distractors to zero. Thus, the filters trained with the sparse constraint can generate sparse response maps on the test set.

Eq. 2 is an NP-hard problem, since it has a l_0 term. To make (2) tractable, we add an auxiliary quadratic constraint [29], and rewrite (2) as:

$$\min_{\mathbf{w},\mathbf{r}} \|\Phi^T \mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 + \tau \|\mathbf{r}\|_0 + \beta \|\Phi^T \mathbf{w} - \mathbf{r}\|_2^2, \tag{3}$$

where β is a parameter controlling the similarity between $\mathbf{r}$ and $\Phi^T \mathbf{w}$. When β is large enough, $\mathbf{r}$ approximately equals to $\Phi^T \mathbf{w}$. The solution to (3) can be found by alternatively solving for $\mathbf{w}$ and $\mathbf{r}$ (see supplementary for derivations),

$$\min_{\mathbf{w}} \|\Phi^T \mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 + \beta \|\Phi^T \mathbf{w} - \mathbf{r}\|_2^2, \tag{4}$$

$$\min_{\mathbf{r}} \tau \|\mathbf{r}\|_0 + \beta \|\Phi^T \mathbf{w} - \mathbf{r}\|_2^2. \tag{5}$$

For kernel (nonlinear) regression, $\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i)$, and thus variables under optimization are $\boldsymbol{\alpha}$. Referring to [14, 23], $\boldsymbol{\alpha}$ can be obtained by a closed-form solution

$$\hat{\boldsymbol{\alpha}} = \frac{\hat{\mathbf{y}} + \beta \hat{\mathbf{r}}}{\left(\beta + 1\right) \hat{\mathbf{k}} + \lambda}, \tag{6}$$

where $\mathbf{k}$ is the first row of the kernel matrix $\mathbf{K}$ whose elements $k_{ij} = \phi\left(\mathbf{x}_i\right)^T \phi\left(\mathbf{x}_j\right)$, the hat $\hat{}$ denotes the DFT of a vector, and the fraction means element-wise division. The optimal solution $\mathbf{r}$ in (5) is obtained from

$$\mathbf{r} = \sigma\left(\frac{\tau}{2\beta}, \mathcal{F}^{-1}\left(\hat{\boldsymbol{\alpha}} \odot \hat{\mathbf{k}}\right)\right), \tag{7}$$

where $\sigma(\cdot)$ is a soft-thresholding function,

$$\sigma\left(\varepsilon, x\right) = \text{sign}(x) \max(0, |x| - \varepsilon). \tag{8}$$

After we obtain $\hat{\boldsymbol{\alpha}}$, the response map for a candidate region $\mathbf{z}$ can be computed in frequency domain from

$$\hat{\mathbf{f}}\left(\mathbf{z}\right) = \hat{\mathbf{k}}^{\mathbf{xz}} \odot \hat{\boldsymbol{\alpha}}, \tag{9}$$

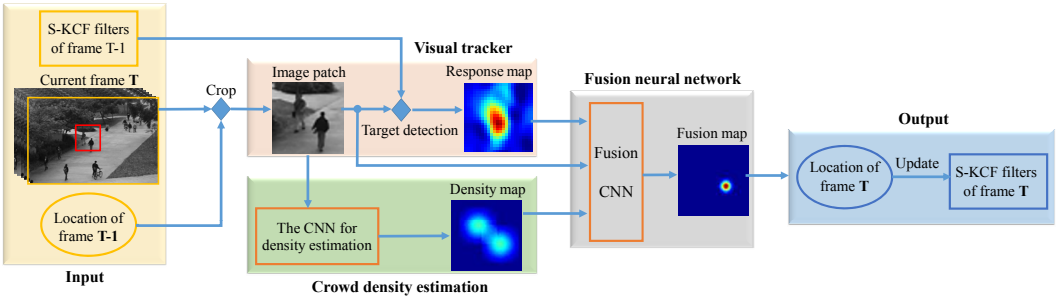

Figure 2. The proposed density fusion framework. The input of the framework is the current frame **T**, S-KCF filters in frame **T-1** and target location of frame **T-1**. The output is target location of frame **T** and the updated S-KCF filters. An image patch in current frame is first cropped based on previous target location, and then is input into S-KCF to generate a response map. At the same time, a crowd density map is estimated from the image patch. The fusion CNN takes in the image patch, S-KCF response map, and crowd density map and produces a refined fused response map, whose maximum value indicates the location of the target. The S-KCF filters are then updated based on the predicted target location.

where $\mathbf{x}$ is the latest target region and $\mathbf{k}^{\mathbf{xz}}$ is the first row of the kernel matrix $\mathbf{K}^{\mathbf{xz}}$ whose element $k_{ij}^{\mathbf{xz}} = \phi(\mathbf{x}_i)^T \phi(\mathbf{z}_j)$. In our experiments, we use 5 iterations of the alternating solver. The overall complexity of the sparse KCF model is $\mathcal{O}(n \log n)$ [14, 27], and thus remains as efficient as standard KCF. Here, n is the number of pixels in the image patch. For UCSD (158x238), the actual running times for KCF, S-KCF, LCT and DSST are 178.2 fps, 112.6 fps, 43.57 fps and 137.67 fps, respectively.

Fig. 3 shows an example comparing the response maps of KCF and S-KCF. The candidate region is presented in Fig. 3(a), where the target object is the person in the middle. The response maps for KCF and S-KCF trackers appear in Figs. 3(b) and 3(c), respectively. The KCF response generates a bias caused by other objects, and thus the detected location drifts to another object. In contrast, the response map of our S-KCF has a clear peak in the response map, which makes it less likely to drift.

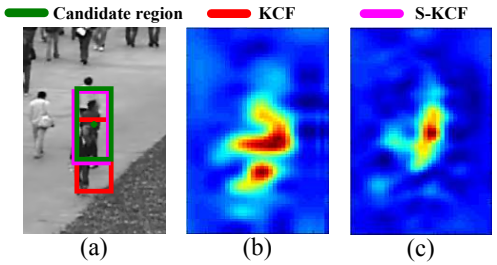

Figure 3. Correlation filter response using KCF and our S-KCF: (a) the candidate region and the detection results; the response maps for (b) KCF, and (c) our S-KCF. The response map for our S-KCF has a clearer peak, which prevents drifting in crowded scenes.

3.2. Crowd Density Map Estimation

The CNN structure for crowd density map estimation is shown in Fig. 4, and loosely follows [30], except 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. The input for the network is 33×33 image patch, and the output is the density value at the center pixel of the input patch. The network has 2 convolutional layers and 5 fully-connected layers. The first convolutional layer contains 64 5×5×1 filters, followed by a 2×2 max-pooling layer with 2×2 stride. The second convolutional layer has 64 5×5×64 filters and is followed by a 3×3 max-pooling layer with stride 2×2. The parameters of fully connected layers are shown in Fig. 4, and the rectified linear unit (ReLU) activation function, which is not shown in the figure, is applied after each convolutional/fully-connected layer.

For training, the ground truth density map is created by convolving the annotation map with Gaussian kernels

$$D(p) = \sum_{\mu_i \in \mathcal{P}} \mathcal{N}\left(p; \mu_i, \sigma^2 I\right), \tag{10}$$

where p denotes a pixel location, $\mathcal{P}$ is the set of annotated positions for an image, and $\mathcal{N}\left(p; \mu_i, \sigma^2 I\right)$ is a Gaussian response with mean μ_i and isotropic variance $\sigma^2 I$.

Similar to [30], we train the CNN using two tasks, density estimation and people counting. Both tasks share the same CNN feature extraction layers. People counting is an auxiliary task that helps guide the network to find good image features. The loss function for density estimation is the pixel-wise squared error,

$$\ell_{density} = \frac{1}{N} \sum_{j=1}^{N} \|\hat{D}_j - D_j\|_2^2, \tag{11}$$

where $\hat{D}_j$ is the estimated density value, and N is the input batch size. For simplicity, we treat people counting as a multi-class classification task where each class is the people count within the image patch, and the categorical cross entropy is used as the loss function

$$\ell_{count} = -\frac{1}{N} \sum_{j=1}^{N} \sum_{k=1}^{K} p_{jk} \log \hat{p}_{jk}, \tag{12}$$

where K is the number of classes (the count range), $p_{\cdot k}$ is the true probability of class k and $\hat{p}_{\cdot k}$ is the predicted probability of class k. The two tasks can be combined into a weighted loss function $\ell = \gamma \cdot \ell_{density} + \ell_{count}$, where we set $\gamma = 100$ in our implementation.

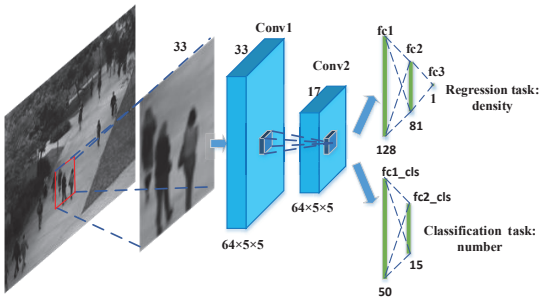

Figure 4. CNN for crowd density map estimation. The CNN contains 2 convolutional and 5 fully connected layers. Two tasks are used for training, density estimation and people counting. People counting is an auxiliary task to guide the network to find good image features.

3.3. Fusion CNN

The fusion CNN combines the tracker response map, the crowd density map, and the image patch to produce a refined (fused) response map, where the maximum value indicates the target position. Note that the image patch is included as input, since it can provide context/localization information (e.g., edges) that are not visible in the response/density maps (see Fig. 7 for ablation studies). The structure of our fusion CNN is shown in Fig. 5. The input has 3 channels: image patch, crowd density map, and response map. The size of the input is selected according to the average target size in a video. Our fusion network has 3 convolutional layers (Conv1-Conv3). Conv1 has 64 $9\times9\times3$ filters, Conv2 has 32 $1\times1\times64$ filters, and the last Conv3 has 1 $5\times5\times32$ filters. The ReLU activation function is applied after each convolutional layer. Note that no max-pooling layer is used, since the goal is to produce a fused response map with the same resolution as the input.

For training, the ground truth fused response map is generated based on the annotated point of the target position as $R(p) = \mathcal{N}(p; \mu, \sigma^2 I)$, where p denotes a pixel location, and μ is the target position. The loss function is the pixel-wise squared error,

$$\ell = \frac{1}{N} \sum_{j=1}^{N} \|\hat{R}_j - R_j\|_2^2, \tag{13}$$

where $\hat{R}_j$ is the estimated response map, and N is the input batch size.

3.4. Two-stage Training Strategy for Fusion CNN

Training the fusion CNN requires the response maps generated by KCF, but the KCF model is also updated according to the fusion response maps (see Fig. 2). Because of the interplay between the output of one frame with the input in the next frame, we adopt a two-stage training procedure to gradually optimize the fusion CNN. The first stage trains a preliminary model based on image patches selected around the targets, and the second stage fine-tunes the preliminary model using the actual tracking process in Fig. 2.

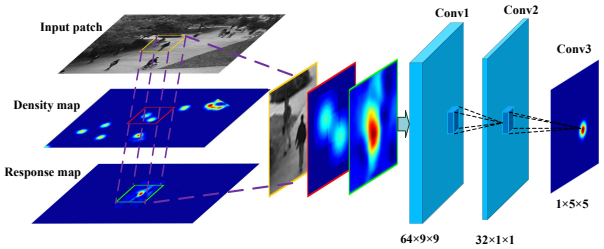

Figure 5. The structure of the fusion CNN. The fusion network has three input channels, and can effectively fuse the appearance information (image patch), with the crowd density map, and the visual tracker response map.

In the first stage, we train the fusion CNN in "batch" mode, where each frame is treated independently. I.e., the predicted fused response map is not used to update the KCF filters. To collect the training data, we first choose a time point for a given target in the video clip, and then initialize the S-KCF to track the target for 50 frames. For each frame, we sample 8 windows randomly around the target's ground truth position. Using each window, we generate the input patches (response map, crowd density map, and image patch) and output (ground-truth response map). The window shift is limited to the maximum target shift in the dataset. For correlation filter trackers, the image patch size is usually larger than the target in order to provide background context – we set the image patch size as 2.5 times average target size. To augment more training samples, we choose 5 starting points for each target in a video clip.

In the second stage, we run the fusion CNN, and use the fusion response map to predict target position. The predicted position is used to update S-KCF, and to generate a new training sample for the next frame (input window & output GT response). This is iterated over frames, and the samples used for fine-tune training the fusion CNN.

4. Experiments

In this section, we evaluate our framework on the task of people tracking in crowd scenes on the *UCSD* dataset [5] and *PETS2009* [22] datasets.

4.1. Dataset

For the *UCSD* dataset, we use the 1200 frames test set for crowd counting [4], containing 152 unique people for tracking. The frames are split into six video clips, based on the crowd density (sparse, medium, high). All video clips are grayscale with dimensions 238×158 at 10 fps.

Table 1. Distance precisions on *UCSD* dataset for location error threshold 10 (P@10).

Scene	*Sparse density*					*Medium density*					*High density*					*Average performance*				
Model \ Tracker	KCF	S-KCF	LCT	DSST	DPM	KCF	S-KCF	LCT	DSST	DPM	KCF	S-KCF	LCT	DSST	DPM	KCF	S-KCF	LCT	DSST	DPM
Raw	0.3073	**0.3341**	0.2979	0.2336	-	0.5907	0.5429	0.5602	**0.5923**	-	0.3604	**0.4101**	0.3188	0.3592	-	0.4235	**0.4356**	0.3932	0.4058	-
FusionCNN-v1 (ours)	**0.4902**	0.4895	0.4244	0.3109	-	**0.6854**	0.6723	0.5950	0.5644	-	0.4802	**0.5535**	0.3060	0.3708	-	0.5501	**0.5772**	0.4293	0.4202	-
FusionCNN-v2 (ours)	**0.5546**	0.4873	0.4939	0.3268	-	0.7138	**0.7312**	0.6633	0.6712	-	0.4890	**0.5623**	0.4293	0.5350	-	0.5786	**0.5999**	0.5217	0.5300	-
Density-aware [25]	0.2740	0.2487	-	-	0.3203	0.2388	0.2294	-	-	0.1357	0.1926	0.1386	-	-	0.1410	0.2273	0.1948	-	-	0.1821

Figure 6. Example frames from *UCSD* and *PETS2009* and the corresponding crowd density maps. The datasets have low-resolution images and various crowd densities.

In the *PETS2009* dataset, we use the people tracking subdataset S2 for evaluation. The subdataset S2 contains three video clips: sparse density L1 (795 frames), medium density L2 (436 frames) and high density L3 (240 frames). Since our focus is on people tracking in crowd scenes, we only select the first 201 frames in L1 for evaluation. In total, there are 877 frames with 44 unique people for tracking. The original image resolution for *PETS2009* is 576×768. We reduce the image resolution to half and use the gray-scale images to evaluate our proposed framework.

We randomly select 80% of the unique people for training the fusion CNN, and the remaining 20% are held out for testing. To train the density map CNN on the *UCSD* dataset, we use 800 frames that are specified for training crowd counting methods, which are distinct from the 1200 frames used for evaluating tracking. For the *PETS2009* dataset, the density map CNN is only trained on the *PETS2009* data, while the fusion CNN is fine-tuned from the network learned from the *UCSD* dataset. We train three separate models on L1, L2 and L3, since the 3 sequences have different properties: L1 has occlusions, L2 has intersecting paths, and L3 has many distractors. Some example frames from the two datasets and the corresponding estimated crowd density maps are shown in Fig. 6.

4.2. Experiment setup

We compare our S-KCF with three other recent visual trackers, KCF [14], LCT [20] and DSST [6], on the tracking datasets using HOG features (denoted as "Raw" model). To show the general effectiveness of using density map fusion, we train a separate fusion CNN for each tracker, using the two-stage training procedure (denoted as "FusionCNN-v2"), and evaluate the tracking performance of the fused response map. To show the effectiveness of two-stage training, we compare with a fusion CNN using only the first stage of training, denoted as "FusionCNN-v1".

We also compare our CNN fusion method with the density-aware method [25], which uses the crowd density map to regularize the detections/tracks found in the detection score map, produced using deformable parts model (DPM) [10]. As the detection score maps do not contain association information between frames, we also test [25] using the response maps of the KCF and S-KCF trackers in place of the detection score map, where the trackers are updated after each frame using the predicted location. Trackers are evaluated by distance-precision, which show the percentage of frames whose estimated location is within the given distance threshold of the ground truth.

The S-KCF paramaters are set to $\lambda = 10^{-4}$, $\tau = 10^{-4}$, and $\beta = 0.05$. The learning rate for S-KCF is 0.02. Currently, we update S-KCF in all frames regardless of occlusion, and the update process is the same as KCF. For scale estimation, we construct a patch pyramid around the estimated target location, and the patch with maximum response value is regarded as the current scale. We implement the fusion CNN using the Caffe [16] framework. The standard stochastic gradient descent with momentum is employed for training, where the initial learning rate, momentum and weight decay are set to 10^{-4}, 0.9 and 10^{-3}. The network converges after approximately 300K for stage-1 training, and 50K iterations for stage-2 training, using mini-batches of 64 samples.

4.3. Evaluation of People Tracking

Table 1 presents the precision for location error threshold 10 (P@10) on the *UCSD* dataset. Overall, our S-KCF performs better than KCF, LCT and DSST (average P@10 of 0.4356 vs 0.4235, 0.3932 and 4058), which demonstrates that the sparsity constraint on response map suppresses the spurious responses. When combined with crowd density using our density fusion framework (FusionCNN-v2), all the trackers can be improved significantly (e.g., KCF improves P@10 from 0.4235 to 0.5501, while S-KCF improves from 0.4356 to 0.5999). Comparing the training strategies, using the two-stage training (FusionCNN-v2) consistently improves over using the first stage (FusionCNN-v1). The density-aware method [25] for fusion does not perform as well as our fusion method. Although most of the pedestrians in an image can be detected, the density-aware method has problems associating the same target together between frames in crowd scenes. Here, performance of each tracker is better on medium/crowded scenes due to more context information (e.g, people in groups) than sparse scenes.

Tracking results P@10 on the *PETS2009* dataset are

Table 2. Distance precisions on *PETS2009* dataset for location error threshold 10 (P@10).

Scene	*Sparse density*					*Medium density*					*High density*					*Average performance*				
Model \ Tracker	KCF	S-KCF	LCT	DSST	DPM	KCF	S-KCF	LCT	DSST	DPM	KCF	S-KCF	LCT	DSST	DPM	KCF	S-KCF	LCT	DSST	DPM
Raw	0.3446	**0.3705**	0.3316	0.2565	-	0.1040	0.1345	0.1125	**0.1481**	-	**0.4937**	0.4810	0.3924	0.4709	-	0.2251	**0.2447**	0.2048	0.2378	-
FusionCNN-v1 (ours)	0.3782	0.3964	0.4404	**0.4896**	-	0.1293	**0.1823**	0.1256	0.1125	-	0.4848	0.4924	0.4241	**0.5101**	-	0.2432	**0.2813**	0.2335	0.2514	-
FusionCNN-v2 (ours)	0.5078	0.5466	0.4611	**0.6995**	-	0.1842	**0.2188**	0.1776	0.1631	-	0.5342	0.5354	0.4304	**0.5570**	-	0.3054	**0.3326**	0.2710	0.3196	-
Density-aware [25]	0.0415	0.0466	-	-	0.1865	0.1115	0.0947	-	-	0.0642	0.0304	0.0304	-	-	0.0304	0.0840	0.0737	-	-	0.0704

summarized in Table 2. S-KCF achieves better results, compared to KCF, LCT and DSST (average P@10 of 0.2447 vs 0.2251, 2048 and 0.2378). All the trackers perform worse on the medium density (L2) scene. The L2 scene has many intersecting paths, which makes tracking more challenging. In contrast, almost all the people in the heavy density (L3) scene are moving in the same direction at the same speed, which causes very few intersections. Similar to the UCSD dataset, all the trackers improve step-by-step using our fusion framework (e.g., DSST improves P@10 from 0.2378 to 0.2514 to 0.3196).

We also evaluate tracking performance using Intersection-over-Union (IoU), and the average success rates at IoU = 0.5 are summarized in Table 3. Overall, all trackers are improved significantly, except for DSST tracker on *PETS2009*. However, DSST tracker can be also improved from 0.3468 to 0.3695 at IoU = 0.3.

Tracking results for KCF [14], LCT [20], DSST [6] density-aware [25], S-KCF and our FusionCNN-v2 are shown in Fig. 8 (see supplementary for video results). The density-aware method [25] drifts easily to other objects, since there are no effective features to associate targets together between frames. The KCF, LCT and DSST trackers fail to track the target when encountering heavy occlusions. Combining S-KCF with crowd density, our FusionCNN-v2 works well for people tracking in crowd scenes, and prevents drift during occlusions and intersecting paths. On UCSD, the running times of fusion CNN with KCF, S-KCF, LCT and DSST are 19, 18, 10, 23 fps.

To further compare S-KCF with KCF, we use a χ^2 significance test on the numbers of tracked and lost frames. Raw S-KCF is significantly better than raw KCF on *PETS* (p=0.03), but similar on *UCSD* (p=0.19). Fusion S-KCF is better than Fusion KCF on both datasets (p=0.02, p=0.02). Fusion is always better than Raw ($p<10^{-4}$).

Table 3. The average success rates at IoU = 0.5.

Dataset	*UCSD*					*PETS2009*				
Model \ Tracker	KCF	S-KCF	LCT	DSST	DPM	KCF	S-KCF	LCT	DSST	DPM
Raw	0.4963	0.5148	0.3847	**0.5568**	-	0.2580	**0.2807**	0.2233	0.2604	-
FusionCNN-v1	0.6208	**0.6213**	0.4347	0.4721	-	0.2743	**0.2755**	0.1940	0.2202	-
FusionCNN-v2	**0.6561**	0.6378	0.4546	0.5951	-	0.3480	**0.3559**	0.2752	0.2326	-
Density-aware [25]	0.2515	0.2349	-	-	0.2231	0.1227	0.1012	-	-	0.1193

4.4. Tracking with Color Cues

The previous experiment only uses edge intensity cues (HOG), while color cues could also help to distinguish nearby pedestrians to improve tracking performance in crowds. Here we test tracking using intensity and RGB color cues together (denoted as HOG+A). Table 4 summarizes the tracking results, and all the trackers can be indeed improved when incorporated with color cues and intensity (e.g., KCF improves from 0.4235 to 0.5317 for *UCSD*, and from 0.2251 to 0.2520 for *PETS2009*). The fusion model can improve visual trackers more than incorporating color and intensity cues (e.g., for *UCSD*, DSST tracker improves from 0.4058 to 0.4364 using HOG+A features, while it improves from 0.4085 to 0.5300 when fused with our fusion CNN). The last row of the table (Fusion / HOG+A) shows the fusion results using trackers with HOG+A features. For *UCSD* and *PETS*, all the trackers can be further improved (e.g., KCF improves from 0.5786 to 0.6495 on *UCSD*, and improves from 0.3054 to 0.3257 on *PETS2009*).

Table 4. Tracking results using color cues and intensity for location error threshold 10 (P@10).

Dataset	*UCSD*				*PETS2009*			
Model \ Tracker	KCF	S-KCF	LCT	DSST	KCF	S-KCF	LCT	DSST
Raw / HOG	0.4235	**0.4356**	0.3932	0.4058	0.2251	**0.2447**	0.2048	0.2378
Raw / HOG + A	**0.5317**	0.5259	0.5146	0.4364	0.2520	0.2529	0.2350	**0.2550**
Fusion / HOG	0.5786	**0.5999**	0.5217	0.5300	0.3054	**0.3326**	0.2710	0.3196
Fusion / HOG + A	**0.6495**	0.6376	0.5956	0.5341	0.3257	**0.3511**	0.2967	0.3245

4.5. Cross-crowd and Cross-scene Generalization

The experiment in Section 4.3 trained a separate fusion model for each crowd level in *PETS2009*, since they have uniquely different properties. Here, we report the tracking results when using uniform fusion model trained on all crowd levels in Table 5. For simplicity, we only take S-KCF tracker for an example. Overall, the uniform fusion model performs a little worse than the separate model (the average P@10 of 0.3245 vs 0.3326, and the average IoU = 0.5 of 0.3381 vs 0.3559). However, the uniform fusion model can still significantly improve S-KCF tracker (the average P@10 improves from 0.2447 to 0.3245, and the average IoU = 0.5 improves from 0.2807 to 0.3381). L2 accounts for about half of the whole training dataset. In order to train the uniform model, we augment L1 and L3 for balancing the training dataset. The augment results in the performance of the uniform model degradation on L2, but the uniform model can still improve S-KCF on L2 (average P@10 improves from 0.1345 to 0.1781). Also, the uniform model can improve S-KCF more than separate model on L1 (average P@10 of 0.7332 of 0.5466).

We also evaluate training and testing across crowd-levels, where the fusion model is trained only on either L1, L2 or L3. Each model performs well on its own training scene (e.g., "L1-trained" performs well on L1), but may

perform bad on other scenes. “L2-trained" and “L3-trained model" also can improve the performance on L1, while they perform bad between each other. Since each scene in *PETS2009* has different properties, thus training a separate model for each scene may be a good choice to improve the tracking performance.

Finally, we evaluate the cross-scene generalization ability of the fusion model. We also take S-KCF tracker for example. We first use the model trained on *UCSD* to test tracking on *PETS2009*, and the performance becomes worse compared to the raw tracker (average P@10 of 0.1163 vs 0.2447). However, the uniform model trained on *PETS2009* improves the tracking on *UCSD*, from P@10 of 0.4356 to 0.4551. This suggests that intersecting paths in *PETS2009* make it more difficult than *UCSD*.

Table 5. Evaluations using cross-crowd-level models.

Scene	Sparse density L1		Medium density L2		High density L3		Average	
Model \ Metric	P@10	IoU=0.5	P@10	IoU=0.5	P@10	IoU=0.5	P@10	IoU=0.5
Raw	0.3705	0.4974	0.1345	0.1425	0.4810	0.5481	0.2447	0.2807
Separate model	0.5466	0.7124	**0.2188**	**0.2071**	**0.5354**	**0.5835**	**0.3326**	**0.3559**
Uniform model	**0.7332**	**0.9585**	0.1781	0.1678	0.5203	0.4949	0.3245	0.3381
L1-trained model	0.5466	0.7124	0.0670	0.0567	0.2608	0.4291	0.1692	0.2221
L2-trained model	0.4896	0.5855	**0.2188**	**0.2071**	0.3873	0.5114	0.2906	0.3239
L3-trained model	0.5984	0.7073	0.1092	0.1143	**0.5354**	**0.5835**	0.2680	0.2955

4.6. Comparison of fusion CNN architectures

In this subsection we compare different variations of the fusion CNN architecture. To evaluate the effectiveness of the three-layer CNN, we tested a two-layer CNN by removing Conv2, and a four-layer CNN by adding a new convolutional layer with 32 3×3×64 filters after Conv1. We also tested the effect of using different input modalities by using only two of the input channels, by removing either the image patch channel, the crowd density channel, or the response map channel. To test the effectiveness of the crowd-density map, we also tried replacing the crowd density map with the DPM detection score map. Another baseline comparison is to directly (element-wise) multiply the response map by the crowd density to obtain a fused response map. Finally, a three-layer CNN is trained only using the real-tracking procedure (stage-2 of the two-stage training) to test the effectiveness of our proposed two-stage training strategy. We tested all these variations on the *UCSD* dataset using the S-KCF tracker.

Fig. 7 shows the average tracking results on the *UCSD* dataset. The three-layer CNN performs better than the two or four layer versions. Using all three input channels performs better than using any of the two channels as input. This demonstrates that the three input modalities are complementary and each provides useful information. Finally, the model trained using the two-stage training strategy achieves better results than using just one-stage training, either "batch" (stage-1) or "real-tracking" (stage-2).

5. Conclusions

In this paper, we address the problem of people tracking in crowd scenes. Using a sparsity constraint on the response map, we first design a sparse KCF tracker to suppress response variations caused by occlusions and illumination changes, as well as similar distractor objects. We fuse the appearance-based tracker and crowd density map together with a three-layer fusion CNN to produce a refined response map. Experimental results show that our fusion framework can improve the people tracking performance of appearance-based trackers in crowd scenes.

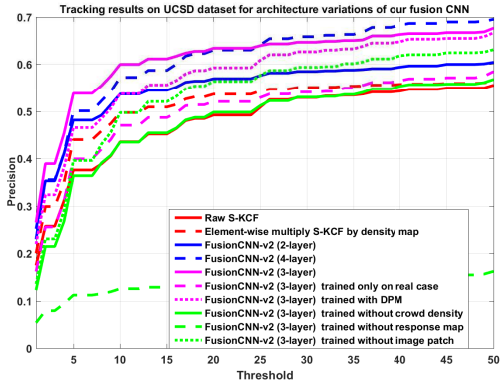

Figure 7. Tracking results on *UCSD* dataset for architecture variations of our fusion CNN.

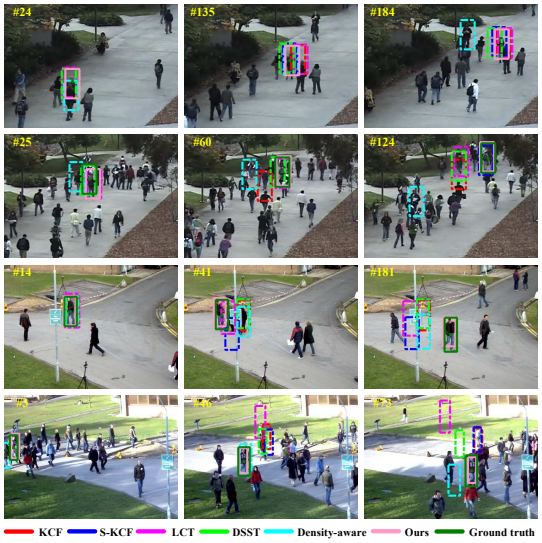

Figure 8. Tracking results on *UCSD* dataset (the first two rows) and *PETS2009* dataset (the last two rows). We show the results using color images for clear visualization.

6. 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]), by a Strategic Research Grant from City University of Hong Kong (Project No. 7004887), and by the Natural Science Foundation of China under Grant 61333019. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

[1] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr. Staple: Complementary learners for real-time tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1401–1409, 2016.

[2] D. M. Blei and J. D. Lafferty. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35, 2007.

[3] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2544–2550. IEEE, 2010.

[4] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE, 2008.

[5] A. B. Chan and N. Vasconcelos. Counting people with low-level features and bayesian regression. *IEEE Transactions on Image Processing*, 21(4):2160–2177, 2012.

[6] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[7] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4310–4318, 2015.

[8] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*, pages 472–488. Springer, 2016.

[9] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014.

[10] 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*, 32(9):1627–1645, 2010.

[11] L. Fiaschi, U. Koethe, R. Nair, and F. A. Hamprecht. Learning to count with regression forest and structured labels. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2685–2688. IEEE, 2012.

[12] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European conference on computer vision*, pages 702–715. Springer, 2012.

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[15] H. Idrees, I. Saleemi, C. Seibert, and M. Shah. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2547–2554, 2013.

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[17] H. Kiani Galoogahi, T. Sim, and S. Lucey. Correlation filters with limited boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4630–4638, 2015.

[18] V. Lempitsky and A. Zisserman. Learning to count objects in images. In *Advances in Neural Information Processing Systems*, pages 1324–1332, 2010.

[19] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel. A survey of appearance models in visual object tracking. *ACM transactions on Intelligent Systems and Technology (TIST)*, 4(4):58, 2013.

[20] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5388–5396, 2015.

[21] Z. Ma, L. Yu, and A. B. Chan. Small instance detection by integer programming on object density maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3689–3697, 2015.

[22] A. Milan, S. Roth, and K. Schindler. Continuous energy minimization for multitarget tracking. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):58–72, 2014.

[23] R. Rifkin, G. Yeo, and T. Poggio. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154, 2003.

[24] M. Rodriguez, S. Ali, and T. Kanade. Tracking in unstructured crowded scenes. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1389–1396. IEEE, 2009.

[25] M. Rodriguez, I. Laptev, J. Sivic, and J.-Y. Audibert. Density-aware person detection and tracking in crowds. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2423–2430. IEEE, 2011.

[26] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014.

[27] Y. Sui, Z. Zhang, G. Wang, Y. Tang, and L. Zhang. Real-time visual tracking: Promoting the robustness of correlation filter learning. In *European Conference on Computer Vision*, pages 662–678. Springer, 2016.

[28] W. Xie, J. A. Noble, and A. Zisserman. Microscopy cell counting with fully convolutional regression networks. In *MICCAI 1st Workshop on Deep Learning in Medical Image Analysis*, 2015.

[29] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via l0 gradient minimization. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2011.

[30] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 833–841, 2015.

[31] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang. Fast visual tracking via dense spatio-temporal context learning. In *European Conference on Computer Vision*, pages 127–141. Springer, 2014.

[32] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 589–597, 2016.