Crowd Tracking with Dynamic Evolution of Group Structures

Feng Zhu[1], Xiaogang Wang[2,3], and Nenghai Yu[1]

[1] Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China
[2] Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, China
[3] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

Abstract. Crowd tracking generates trajectories of a set of particles for further analysis of crowd motion patterns. In this paper, we try to answer the following questions: what are the particles appropriate for crowd tracking and how to track them robustly through crowd. Different than existing approaches of computing optical flows, tracking keypoints or pedestrians, we propose to discover distinctive and stable mid-level patches and track them jointly with dynamic evolution of group structures. This is achieved through the integration of low-level keypoint tracking, mid-level patch tracking, and high-level group evolution. Keypoint tracking guides the generation of patches with stable internal motions, and also organizes patches into hierarchical groups with collective motions. Patches are tracked together through occlusions with spatial constraints imposed by hierarchical tree structures within groups. Coherent groups are dynamically updated through merge and split events guided by keypoint tracking. The dynamically structured patches not only substantially improve the tracking for themselves, but also can assist the tracking of any other target in the crowd. The effectiveness of the proposed approach is shown through experiments and comparison with state-of-the-art trackers.

1 Introduction

Crowd motion analysis has recently drawn many attentions because of its important applications in crowd video surveillance including recognizing different crowd events and traffic modes [29, 34–36, 11], detecting abnormal crowd behaviours [29, 15], and predicting crowd behaviours [35]. Different than many conventional surveillance approaches which focus on tracking individuals and analysing their behaviours, crowd surveillance treats the whole crowd as a union at the macroscopic level. It does not require extracted motions exactly corresponding to individual objects, as long as they reflect the motion patterns of the whole crowd. On the other hand, the learned motion patterns can assist tracking a particular target of interest through the crowd [2, 13, 21, 23]. Existing works learn crowd motion patterns through computing optical flows [2, 13, 21, 23], tracking keypoints [34–36] or pedestrians [1, 22, 26].

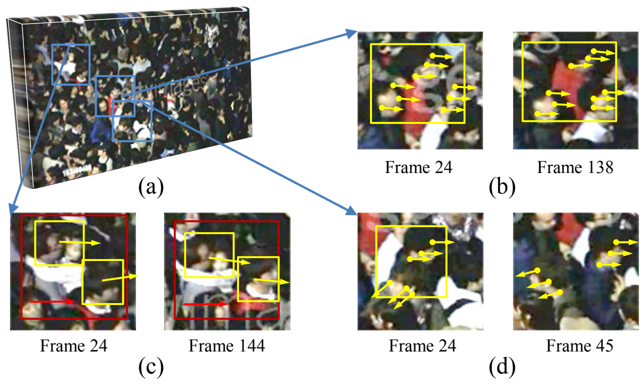


Fig. 1. (a) Crowd video. (b) A good mid-level patch for tracking with coherent internal motions. Since the motions of keypoints and the patch are consistent, tracking the patch well reflects local motions of the crowd. The patch covers three persons whose relative positions are stable over time. They form unique visual pattern for tracking. (c) Multi-scale patches. The red patch covers two small yellow patches. They have coherent motions. The red patch can assist tracking the yellow patches by adding spatial constraint at a larger scale. (d) A patch with incoherent internal motions. It is not suitable for tracking as it cannot keep stable visual pattern. Since its motion differs from those of keypoints inside, it cannot accurately reflect local motion of the crowd.

We can treat crowd tracking as generating trajectories of different length from a set of particles (i.e., pixels, keypoints, patches or pedestrians) at different scale levels. The major challenges of crowd tracking lie in three aspects: (1) partial or full occlusions caused by frequent interactions among objects; (2) a large number of individuals with similar appearance; and (3) significant appearance variation due to the perspective distortion of camera views. Different types of crowd tracking provide different amount of information for further motion analysis, and they also need to balance the risk of tracking errors.

Optical flows are computed at all pixels but their tracking only lasts for one frame. Keypoint tracking only selects good feature points to track. Those keypoints can be considered as the smallest patches. If they are well tracked over multiple frames, keypoint tracking can provide accurate information for crowd motion analysis. However, keypoint tracking is very sensitive to even small occlusions, since a feature point does not contain appearance information from a large area. Therefore, only short tracklets can be obtained. Tracking with pedestrian detection is very difficult because of heavy occlusions especially in very dense crowd and large changes of viewpoints. So far there is no pedestrian detector working robustly in all kinds of crowd scenes.

The above observations motivate us to find good mid-level patches to track, which cover larger areas than keypoints and are more robust to occlusions. Patch tracking can provide us longer trajectories which are useful for crowd motion analysis. We can even track patches with different sizes (Fig. 1 (c)). Large patches are more robust and take less risk of drifting. However, motions captured by them are less accurate, since they are not sensitive to local movements. Then the key question is what are good patches for tracking. Different from other tracking problems, one patch may be placed on multiple objects in crowd as shown in Fig. 1 (b). Such a patch may not be bad for tracking, as long as the two objects move coherently with stable relative positions. Moreover, study [18] has shown that neighbouring pedestrians may form unique visual patterns which make the patch distinctive for tracking. Since keypoint tracking provides accurate motions within short periods, it can help to find patches with stable internal structures and distinctive visual patterns [6, 20, 27]. It is important to detect patches with coherent internal motions, because such patches keep stable appearance and can accurately reflect the motions of the crowd as shown in Fig. 1 (b) and (d).

Scientific studies [14, 16] have shown that when a person is placed in crowd, he or she tends to form collective behaviours with others instead of moving freely. Thus crowd tends to form groups with coherent motions [34]. The relative positions of individuals within a group are more stable. Moreover, the collectiveness of crowd increases as it becomes denser [33]. Therefore, patches within the same coherent group should be tracked together by modeling their spatial structures. It will significantly improve the robustness when tracking through occlusions.

We target on automatically detecting distinctive and stable mid-level patches and jointly tracking them with dynamically evolved group structures. Keypoint tracking, patch tracking and group evolution are integrated at three different levels. It is motivated by our insights on the strength and weakness of the three aspects in crowd tracking. While patch tracking is more robust to partial occlusion and appearance change, keypoint tracking provides more accurate motion information in short periods. Keypoint tracking can help to detect patches with stable internal structures, and organize patches into groups with coherent motions and stable structures. Patches are tracked with a new dynamic hierarchical tree structure. It models the spatial relationships between patches at different scales and the evolution of group structures. Since crowd motions change dynamically, group structures are updated through merge and split over time.

2 Related Work

Some works [2, 13, 21, 23] have been done on tracking targets through crowd by learning models of scene structures and long-term motion patterns from optical flows or trajectories of keypoints. Ali and Shah [2] proposed multiple floor fields to assist tracking targets through crowd. These floor fields characterize forces from dominant paths, preferred exit regions, and boundaries of scene structures. Rodriguez *et al.* [21] employed the Correlated Topic Model [5] to learn a mixture of motion patterns for a specific scene and used it as prior to guide tracking. In [23], they extended this approach such that the learned priors of crowd behaviors can be transferred across scenes. Kratz and Nishino [13] captured the crowd motion at each spatio-temporal location with 3D Gaussian distributions and HMM, and used it as prior to guide tracking. All these models of scene structures and crowd motion patterns are learned from optical flows with off-line training.

None of them jointly track multiple targets together. Their focus is on *tracking a particular particle through the crowd* instead of *tracking the whole crowd with a set of particles* as we do. Our online-tracking approach does not require a training process to obtain priors of scene structures or motion patterns.

Recently, the social force model [9] has been used to track multiple pedestrians by modeling their interactions, the influence of destinations and scene structures with a physical model [19, 24]. This approach requires a lot of prior knowledge on scene structures and only suitable to top-down views. The parameters of the physical models need to be manually set or trained for each scene specifically. The initialization of tracking must rely on a pedestrian detector on “all” the individuals rather than keypoint or patch detectors (since the social force model is based on the psychological and physiological interactions of individuals) which generally does not work well in crowded scene. Manual initialization was used in [19, 24]. It is not suitable for general crowd tracking.

Many model-free trackers [13, 3, 8, 30] are proposed to track general objects including patches. They track each target separately without modeling the spatial constraints among targets. Idrees, Warner and Shah [10] tracked the crowd using neighbourhood motion concurrence. The work most relevant to ours is the structure preserving multi-object tracking proposed by Zhang and Maaten [31]. It jointly tracks multiple objects by modeling their spatial constraints. This work has several major differences with ours. Its patches are manually initialized and considered as one group during the whole tracking process (i.e. its group structure and the number of edges connecting patches are fixed), while we automatically detect patches and dynamic update group structures through merge and split operations. In [31], all the patches are placed on coherently moving objects with stable relative positions. However, in crowd tracking, patches may be on groups moving in different directions, and adding spatial constraints on them may bias the tracking. In [31], the spatial constraints of patches are modeled at a single scale, while a hierarchical tree structure at multiple scales is used by us.

3 Our Method

Our crowd tracking framework is shown in Fig. 2. It integrates low-level keypoint tracking, mid-level patch detection and tracking, and high-level group evolution. Keypoints are tracked with the KLT tracker [28]. Whenever ambiguity arises due to occlusions or other factors, keypoint tracking stops. Therefore, keypoint tracking can provide accurate information on crowd motions within short periods. Keypoint tracking results are used to detect mid-level patches suitable for tracking (Section 3.1), and update group structures over time (Section 3.2). Patches are tracked together with the spatial constraints added by the dynamic hierarchical tree structures (Section 3.3).

3.1 Patch Detection

Patches good for tracking should satisfy two requirements: (1) the appearance is distinctive; and (2) points inside the patch have coherent motions. In [25],

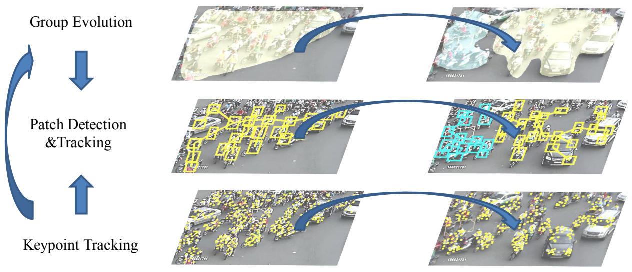


Fig. 2. Our crowd tracking framework integrates low-level keypoint tracking, mid-level patch detection and tracking, and high-level group evolution. Both group evolution and patch detection are guided by keypoint tracking. Group structures, which are used to assist patch tracking, are updated during high-level group evolution.

feature points good for tracking are selected as "Shi-Tomasi Corners" which are distinctive in local areas. We assume that patches with high density of such corner points are easy to track as they contain more distinctive textures. In order to find such patch candidates, we adopt the clustering method proposed in [32] to find dense clusters of keypoints. In [32], a K-NN graph is built from keypoints and graph indegrees well reflect the boundaries of keypoint distributions with different density levels. Outliers with low indegrees are removed and dense clusters with high indegrees are detected through agglomerative clustering. These dense clusters help to generate patch candidates.

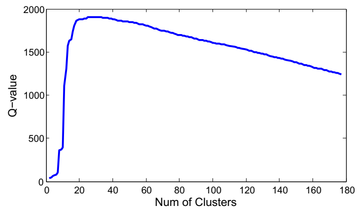
Instead of clustering keypoints directly on K-NN graph, we first measure motion coherence [34] between neighbouring keypoints. If a keypoint moves coherently with others, its neighbour set should keep invariant over time and its motion correlation with neighbours should be high. To achieve this, starting at frame t, the invariant neighbours of keypoint i in the successive $d+1$ frames are found as $\mathcal{M}^i_{t\to d} = \bigcap_{\tau=t}^{t+d} \mathcal{N}^i_\tau$, where $\mathcal{N}^i_\tau$ is the K-NN set of keypoint i in frame τ. The motion correlation between i and its invariant neighbour j is measured by

$$C_{i,j} = \begin{cases} \dfrac{1}{d+1} \sum_{\tau=t}^{t+d} \dfrac{\mathbf{v}^i_\tau \cdot \mathbf{v}^j_\tau}{\|\mathbf{v}^i_\tau\| \cdot \|\mathbf{v}^j_\tau\|}, & \text{if } j \in \mathcal{M}^i_{t\to d}, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

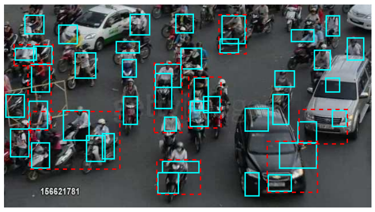
where $\mathbf{v}^i_\tau$ is the velocity of i in frame τ and $C_{i,j}$ is the (i,j) entry of motion correlation matrix $\mathbf{C}$. Given motion coherence, a graph is built among keypoints and it is represented with matrix $\mathbf{G}$ which is derived from $\mathbf{C}$ with entries

$$G_{i,j} = \begin{cases} 1, & \text{if } C_{i,j} > C_h \ \& \ C_{j,i} > C_h, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

C_h is a predefined threshold set as 0.8 in all our experiments. $G_{i,j} = 1$ stands for an edge between i and j. Edges are only assigned to pairs which are both in the invariant neighbour set of each other and have high motion correlation.



(a)



(b)

Fig. 3. (a) A typical curve of Q-value against cluster numbers. (b) Examples of detected patches, red boxes are layer-2 patches mentioned in Section 3.3.

We cluster these keypoints based on a weighted version $\mathbf{G}^w$ of graph $\mathbf{G}$ by integrating accurate spatial information

$$G^w_{i,j} = \begin{cases} \exp(-\frac{dist(i,j)^2}{\sigma^2}), & \text{if } G_{i,j} = 1, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $dist(i,j)$ is the Euclidean distance between i and j, and $\sigma^2 = \sum_{i,j} (dist(i,j)^2 \cdot G_{i,j}) / \sum_{i,j} G_{i,j}$. We apply the graph-based bottom-up clustering algorithm in [32] to find dense keypoint clusters and determine the number of clusters using "Q-value" [17]. A typical curve of Q-value against cluster numbers is in Fig. 3(a). We choose the cluster number with the maximum Q-value.

Patches are estimated from the detected keypoint clusters as follows: $x = \frac{1}{N}\sum_{i=1}^{N} x_i$, $y = \frac{1}{N}\sum_{i=1}^{N} y_i$, $w = 2\sigma_x$, $h = 2\sigma_y$, where (x,y) is the patch center, w and h are width and height respectively, (x_i, y_i) is the coordinate of the i^{th} keypoint, σ_x and σ_y are the standard deviation of x_i and y_i, and N is the number of keypoints in the cluster.

Some examples of patches detected in videos are shown in Fig. 3(b). Static keypoints on the background are filtered out by motion correlation in Eq.(1), so patches locate on moving targets. Since the graph is built with K-NN, the sizes of clusters/patches increase with K. In the extreme case, a fully connected graph becomes one cluster. In Section 3.3, patches are generated with two different scales by choosing $K = 10$ and $K = 20$.

3.2 Group Evolution

Pedestrians in crowd interact with each other and form groups with coherent motions. The relative positions of individuals in the same group are more stable. So it is profitable to identify groups and track targets in the same group jointly. However, crowd group structures are also changeable and affected by pedestrian destinations and scene structures, and therefore need to be updated dynamically.

We derive groups of patches from the results of keypoint tracking, since they provide more accurate motion information. Within a short period Δ, the whole crowd is segmented into several coherent motion patterns with the Collective

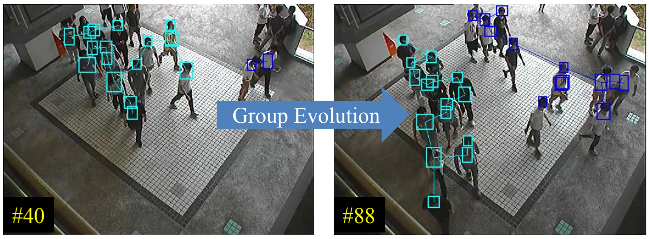


Fig. 4. Group evolution and dynamic structures, both group merge and split occurs

Merging algorithm [33]. It results in several groups of keypoints, such that each group exhibits one collective motion pattern. Grouping information from the keypoint-level guides the organization of mid-level patches. We assign keypoint k a unique label y_k indicating which collective motion it belongs to and determine the label for patch i by majority vote of the keypoints covered by i.

Final groups for patches are generated by temporal smoothing. Let l_t^i be the label of patch i at frame t, then connectivity matrix $\mathbf{L}_t$ is defined as

$$L_t(i,j) = \begin{cases} 1, & \text{if } l_t^i = l_t^j, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where $L_t(i,j)$ is the (i,j) entry of $\mathbf{L}_t$. We sum up connectivity matrices over time by $\mathbf{L}_{sum} = \sum_{k=0}^{k_c-1} \mathbf{L}_{t-k\Delta}$, where k_c is the length of temporal buffer ($k_c = 10$ in all our experiments), and Δ is a short time period for calculating each $\mathbf{L}_t$ ($\Delta = 3$ in all our experiments). Then entry (i,j) of $\mathbf{L}_{sum}$ is set to 1 if $L_{sum}(i,j) \geq \kappa k_c$ and otherwise 0. The connected components in $\mathbf{L}_{sum}$ are extracted to be final groups at frame t. κ controls the value of threshold and is set as $\kappa = 0.7$ in all of our experiments. The buffer window makes the group structures change smoothly and robust to errors of detected collective motions.

By regularly detecting collective motions and grouping patches, we dynamically adjust group structures through merge and split events with the evolution of crowd (Fig. 4).

3.3 Dynamic Hierarchical Tree Structure

Because of frequent occlusions and neighbours with similar appearance, trackers that only utilize appearance features of targets are likely to drift in crowd scenes. We propose a dynamic hierarchical tree structure which imposes spatial constraints on patches in a hierarchical manner and update structures dynamically according to the evolution of groups.

As illustrated in Fig. 5, each group of patches constitute a hierarchical structure which could model more spatial relationships and appearance features than a single tree structure. Since patches selected in Section 3.1 suggest coherent internal motions, we generate patches for the second layer using the same method but with a larger K than the first layer. Then, cross-layer constraints are added

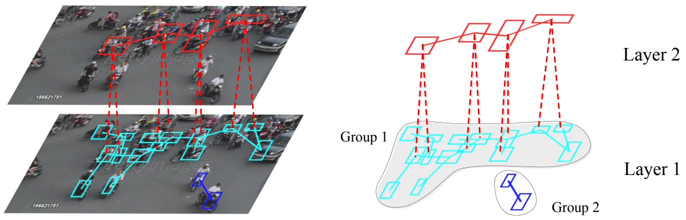


Fig. 5. Dynamic hierarchical tree structure. Patches of different color in the first layer indicates different groups which are generated dynamically. Lines connecting patches stand for spatial relationships.

straightforwardly between layer-2 patches and overlapping layer-1 ones. For each group, the structure at each layer are initialized as a minimum spanning tree with respect to spatial locations, so constraints tend to be added between neighbouring patches. When merge or split occurs, the structures will be merged or split accordingly as shown in Fig. 4. We also re-initialize structures in each group regularly to make sure that constrains are always imposed on adjacent patches.

For each group, we represent the set of patches at layer m by $P^m = \{\, p_i^m \mid p_i^m = (\mathbf{x}_i^m, w_i^m, h_i^m)\,, i = 1, \dots N_m\}$, where $(\mathbf{x}_i^m, w_i^m, h_i^m)$ defines a bounding box for patch p_i^m with location $\mathbf{x}_i^m$, width w_i^m and height h_i^m. Edges within each layer are denoted as E^m, and E_c stand for cross-layer edges. In coming new frames, each group of patches are tracked by maximizing

$$S(\mathcal{C}; \mathbf{I}) = \sum_{m} \sum_{P^m} D(\mathbf{I}, p_i^m) - \sum_{m} \sum_{E^m} L(p_i^m, p_j^m) - \sum_{E_c} L_c(p_i^1, p_j^2), \tag{5}$$

where $\mathcal{C}$ is the locations of all patches to be optimized given the new frame $\mathbf{I}$. The first term $D(\mathbf{I}, p_i^m)$ is a unary term that measures appearance similarity between p_i^m and the corresponding appearance model. $L(p_i^m, p_j^m)$ and $L_c(p_i^1, p_j^2)$ are pairwise terms that encode spatial constraints between patches. $L(p_i^m, p_j^m)$ stands for intra-layer constraints and $L_c(p_i^1, p_j^2)$ are inter-layer ones.

The appearance score is measured as

$$D(\mathbf{I}, p_i^m) = \mathcal{L}(\mathbf{w}_i^m \cdot f(\mathbf{I}, p_i^m)), \tag{6}$$

where $f(\mathbf{I}, p_i^m)$ is the HOG feature vector extracted in p_i^m, $\mathbf{w}_i^m$ is the linear weights on HOG features trained for patch p_i^m using linear SVM. $\mathcal{L}$ stand for logistic function that regularize filter responses to $[0, 1]$. $\mathbf{w}_i^m$ is updated using a passive-aggressive algorithm similarly to [31].

Spatial constraint is defined as

$$L(p_i^m, p_j^m) = \lambda_{ij}^m \|(\mathbf{x}_i^m - \mathbf{x}_j^m) - \mathbf{e}_{ij}^m\|^2, \tag{7}$$

where $\mathbf{e}_{ij}^m$ is the expected relative position between target i and j, parameter λ_{ij}^m ($\lambda_{ij}^m = 0.001$ in all of our experiments) controls the deformation cost in the final score $S(\mathcal{C}; \mathbf{I})$. Spatial constraint $\mathbf{e}_{ij}$ between patch i and j is initialized as

$\mathbf{e}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, and updated by $\mathbf{e}_{ij}^{new} = 0.5(\mathbf{x}_i^{new} - \mathbf{x}_j^{new}) + 0.5\mathbf{e}_{ij}^{old}$. Inter-layer constraints $L_c(p_i^1, p_j^2)$ are formulated similarly to $L(p_i^m, p_j^m)$.

Though it is hard to perform exact inference on loopy graphs, Eq.(5) can be maximized efficiently with an iterative approach. First, the optimal configuration of patches in the first layer regardless of inter-layer constraints is found by

$$\hat{P}^1 = \max_{P^1} \sum_{P^1} D(\mathbf{I}, p_i^1) - \sum_{E^1} L(p_i^1, p_j^1), \tag{8}$$

where $\hat{P}^1 = \{\hat{p}_i^1 \ | i = 1, \ldots, N_1\}$ represents the optimal solution. Since the first layer is tree-structured, exact inference can be performed in Eq.(8) via dynamic programming. Then, patches in the second layer are located by integrating $\hat{P}^1$ as known

$$\hat{P}^2 = \max_{P^2} \sum_{P^2} D(\mathbf{I}, p_i^2) - \sum_{E^2} L(p_i^2, p_j^2) - \sum_{E_c} L_c(\hat{p}_i^1, p_j^2), \tag{9}$$

where the first two terms are similar to Eq.(8), and $L_c(\hat{p}_i^1, p_j^2)$ turns into a unary term since $\hat{p}_i^1$ is fixed. So Eq.(9) can also be solved efficiently by dynamic programming. Then, $\hat{P}^1$ can be refined by adding inter-layer term $L_c(p_i^1, \hat{p}_j^2)$ with fixed $\hat{p}_j^2$ to Eq.(8) and optimize it again. After several iterations, the overall score $S(\mathcal{C}; \mathbf{I})$ will converge to a stable value, and we stop iterations at the condition that $|S_{new}(\mathcal{C}_{new}; \mathbf{I}) - S_{old}(\mathcal{C}_{old}; \mathbf{I})| < \epsilon$, where ϵ is a small constant which we set as $\epsilon = 10^{-4}$ in our work. It usually takes 2$\sim$4 iterations before convergence.

4 Experiments

We conduct two sets of experiments for evaluation and comparison. In both experiments, our tracker automatically detects mid-level patches regularly and tracks them together with the selected target (which is manually initialized, since all the other trackers in comparison require manual initialization) by adding spatial constraints. Automatically detected patches act as "assistant patches". Targets and assistant patches are tracked together as described in Section 3.2 and 3.3. Assistant patches in layer 1 are treated equally with targets, while ones in layer-2 are discarded if their appearance scores defined in Eq.(6) are lower than a threshold (0.3 in our experiments). Newly detected patches are assigned to current groups and those leaving the field of view are discarded during tracking.

The task of the first experiment is to track a single target through crowd videos. In comparison, several state-of-the-art model-free trackers are also used to track the target. We will show that tracking the crowd as a whole can effectively improve the tracking of any single target in the crowd. In the second experiment, we compare with the structure preserving multi-object tracking (SPOT) approach [31] by jointly tracking multiple targets manually initialized. SPOT also models the spatial constraints among targets during tracking. We will show the advantage of our dynamic hierarchical tree structures during multi-object tracking compared with keeping static structures unchanged in SPOT.

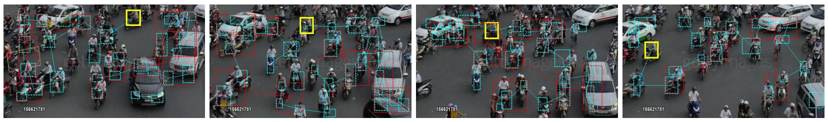


Fig. 6. Tracking single target together with assistant patches. Bounding boxes stand for tracked areas (yellow: the selected target; cyan: layer-1 assistant patches; red: layer-2 assistant patches) and lines connecting these boxes are spatial constraints (cyan: constraints in layer 1; red : constraints in layer 2 and cross-layer constrains).

The dataset[1] used in our experiments contains six videos of crowd scenes as shown in Fig. 7. Three of them (i.e., "Traffic", "Crowds" and "Marathon") are downloaded from the Web. The other three (i.e., "Split", "Merge", "Cross") are captured by us and they exhibit three types of group evolution. The resolutions of these videos are in the range of 480×360 to 768×568.

Due to frequent occlusions, KLT tracker is very unstable and produces highly fragmented tracklets. The average length of its tracks is 10.03 pixels on this dataset, while ours is 94.33. We did not compare with crowd tracking methods relying on priors of "repeated" motion patterns and scene structures [2, 13, 21, 23], since they need offline training to obtain priors. Our videos such as "Split", "Merge" and "Cross" do not have such motion priors since events only happen once. We only compare with model-free trackers as our tracker does not rely on priors of scene structures or repeated motion patterns. Social force models [19, 24] are not compared, since they require human detection on "all" the individuals as initialization. It is impractical in crowd tracking. See details in Sec. 2.

We set $K = 10$ and 20 for detecting patches in layer 1 and 2 respectively. 4×4 cells are used for HOG feature extraction. Other parameter settings are explained in Section 3. Experiments on all the videos share the same settings of parameters. The tracking speed depends on video resolutions and scene crowdness. With an unoptimized matlab implementation, our tracking speed is in the range of 7 fps to 14 fps on this dataset by using Intel Core 2 Duo of CPU 3.0GHz.

4.1 Experiment I: Single-Object Tracking

In this experiment, we manually annotate the tracks of 10 targets through each video (60 targets in total) to evaluate the tracking performance. Only one target is tracked in each time with manual initialization. Experiments run for 10 times on each video. Performance of trackers is evaluated by two metrics: (1) **Error**: average distance between the center of hypothesis returned by trackers and that of the ground truth; and (2) **Recall**: percentage of successfully tracked frames (in which the overlap of hypothesis and ground truth is larger than 50%).

Four state-of-the-art model-free trackers are compared: OAB [8], MIL [3], TLD [12] and CXT [7]. The first 3 trackers concentrate on modeling appearance of tracked targets while the CXT tracker explores context (supporters and distracters) from background to help tracking.

[1] Available from `http://home.ustc.edu.cn/~zhufengx/crowdTracking/`

Table 1. Error and Recall of Experiment I. Bold font indicates best performance.

	Ours		**TLD** [12]		**MIL** [3]		**OAB** [8]		**CXT** [7]	
	Error	Recall	Error	Recall	Error	Recall	Error	Recall	Error	Recall
Traffic	**3.29**	**97%**	85.85	46%	49.86	50%	46.1	67%	47.5	73%
Crowds	**4.37**	**96%**	16.52	67%	10.69	71%	9.87	76%	70.4	55%
Marathon	**5.57**	**87%**	7.35	41%	26.85	31%	15.75	53%	129.5	20%
Split	**3.93**	**91%**	23.91	57%	9.06	64%	21.48	58%	46.8	67%
Merge	**5.67**	**89%**	34.36	47%	14.98	58%	15.31	65%	94.9	40%
Cross	**4.10**	**90%**	22.33	54%	30.09	37%	37.05	50%	97.6	47%

Fig. 6 shows the selected target, assistant patches and spatial constrains in our approach. Tracking result frames shown in Fig. 7 and quantitative comparisons reported in Table 1 indicate that our tracker significantly outperforms others in comparison with large margins. It benefits from the assistance of auxiliary patches used in crowd tracking and also the fact that our approach can successfully detect and track these patches. In "Traffic" and "Crowds" videos, occlusions between objects occur frequently. OAB, MIL and TLD do not work well on such videos as they only model the appearance of targets being tracked without using the motion information from nearby objects. Therefore, these trackers tend to drift when the target is occluded. By tracking the target jointly with automatically detected mid-level patches, selected targets can be tracked well through occlusions. In the "Marathon" video, the targets for tracking are small and less distinctive as the video is captured from bird-view and humans in the scene wear similar clothes. By tracking targets jointly with spatial constraints from neighbouring individuals, targets are less likely to drift among similar objects. In "Split", "Merge" and "Cross" videos, pedestrian heads are selected for tracking. Targets in these videos have large scale variations due to perspective distortion and all heads are very similar in appearance. Our tracker has drawn promising results in such videos. Though CXT tracker has also utilized context information around the target, it can not handle frequent occlusions and too many similar objects in crowd scenes. In our experiment, CXT tracker tend to drift when occlusion happens or jump among analogous neighbours, which lead to the worst performance in all compared trackers.

4.2 Experiment II: Multi-Object Tracking

In this experiment, we compare with the SPOT tracker [31] which has also utilized spatial constraints for multi-target tracking. While in SPOT, tree structures at a single scale are initialized in the first frame and remain unchanged (though relative positions between nodes can be updated) during tracking. We also compare with the NMC [10] tracker which utilizes neighbourhood motion concurrence to predict positions of targets in Experiment II.

More targets are annotated to evaluate multi-object tracking performance, such that the manually initialized and annotated targets can cover the whole crowd. The number of annotated targets for the "Traffic", "Crowds", "Marathon",

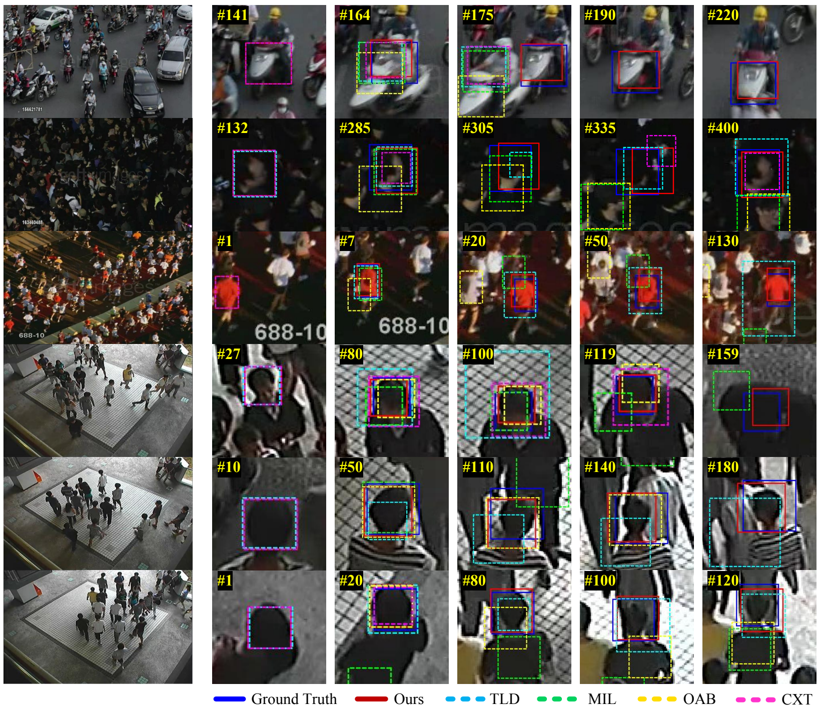


Fig. 7. Snapshots of test videos and some tracking results in Experiment I. From top to bottom: "Traffic", "Crowds", "Marathon", "Split", "Merge", "Cross".

"Split", "Merge", and "Cross" videos are 31, 18, 17, 19, 22, and 19. We also compare with ourselves by removing some functional parts of our tracker. The trackers for evaluation are summarized as (1) DHT+AP: manually selected targets and multi-scale assistant patches (AP) are jointly tracked with our dynamic hierarchical tree structure (DHT); (2) DHT: targets are tracked with only layer-2 assistant patches compared to DHT+AP; (3) TREE: single tree structure connecting all targets which are initialized in the first frame and stays unchanged during tracking (our implementation of SPOT); (4) SPOT; (5) NMC.

Results are presented in Table 2 and 3. Table 2 shows Errors and Recalls which are defined as before, and CLEAR MOT metrics [4] for multi-object tracking are reported in Table 3. In CLEAR MOT metrics, we make correspondence between object and hypothesis if their overlap is larger than 0, so for both MotP (multi-object tracking precision) and MotA (multi-object tracking accuracy), larger value indicates better performance. DHT+AP and DHT outperform other trackers in both Error/Recall and CLEAR MOT metrics. The NMC tracker predicts positions of targets using motions of neighbouring individuals without identifying different groups in crowd, which makes tracking performance drop significantly in scenes with complex motions (e.g. Traffic, Cross). TREE and SPOT are similar in algorithm, but differ in some implementation details (e.g., we perform exact

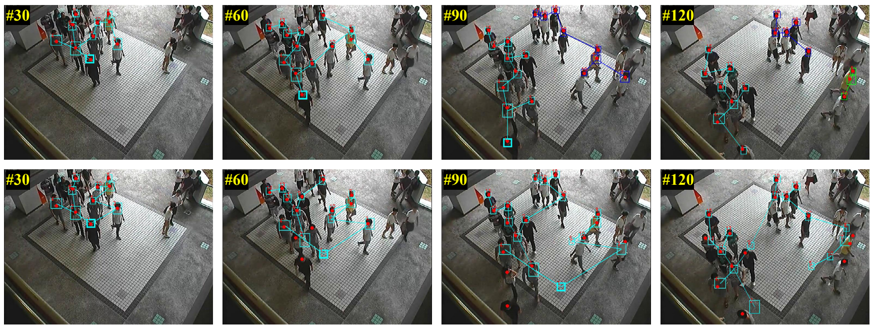
Table 2. Error and Recall of Experiment II. Bold font indicates best performance.

	DHT+AP		**DHT**		**TREE**		**SPOT** [31]		**NMC** [10]	
	Error	Recall	Error	Recall	Error	Recall	Error	Recall	Error	Recall
Traffic	**3.03**	**99%**	3.27	97%	8.93	92%	17.37	81%	35.89	74%
Crowds	8.17	**90%**	8.94	83%	19.78	71%	15.10	64%	**6.64**	81%
Marathon	4.57	88%	**4.26**	**92%**	11.44	77%	112.68	2%	5.97	70%
Split	5.02	**86%**	**4.50**	**86%**	24.09	59%	68.47	25%	9.77	70%
Merge	**8.27**	82%	8.41	**85%**	27.23	62%	48.87	57%	11.57	77%
Cross	5.82	**89%**	**4.61**	**89%**	48.00	51%	90.96	27%	20.62	69%

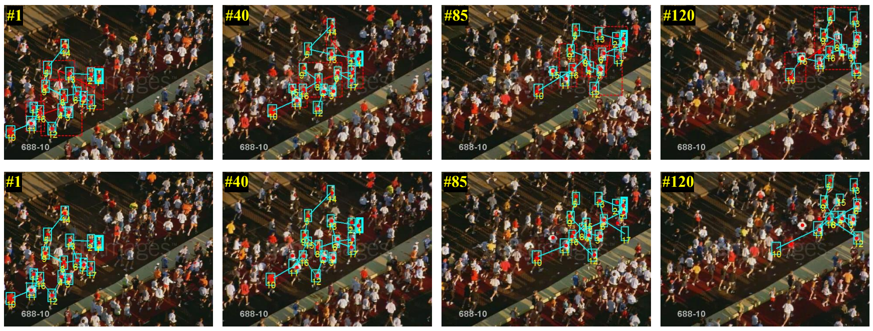
Table 3. CLEAR MOT metrics of Experiment II. Bold font indicates best performance.

	DHT+AP		**DHT**		**TREE**		**SPOT** [31]		**NMC** [10]	
	MotP	MotA	MotP	MotA	MotP	MotA	MotP	MotA	MotP	MotA
Traffic	0.81	**99%**	**0.82**	98%	0.81	84%	0.68	52%	0.72	59%
Crowds	**0.72**	90%	0.70	87%	0.68	72%	0.61	77%	0.66	**99%**
Marathon	0.70	91%	**0.72**	91%	0.69	72%	0.22	-74%	0.62	**94%**
Split	**0.68**	86%	**0.68**	**92%**	0.60	49%	0.45	-6%	0.65	65%
Merge	0.68	**84%**	**0.71**	81%	0.64	50%	0.62	44%	0.69	69%
Cross	**0.72**	90%	0.71	**93%**	0.63	18%	0.49	-10%	0.65	74%

inference on tree structures instead of transforming them into star-structured ones first as in the released code of SPOT). Although performance is improved by better implementation compared to SPOT, static structures used in TREE still do not work well in crowd scenes. Spatial constraints could indeed help track targets which have stable relative positions in crowd, but they may bias tracking if constraints are imposed between targets moving in different directions. Comparisons between DHT and TREE show that our dynamic structure is more suitable for tracking in crowd scenes than the static structure. There seems to be no significant difference in performance between DHT+AP and DHT because we have manually selected sufficient targets for tracking, so finding more mid-level patches does not help more. In "Split", "Merge" and "Cross" videos, group motions change drastically, so that a "good" spatial constraint may turn into a "bad" one as the group evolves. Our approach can identify these changes, and adjust group structures through merge and split (Fig. 8(a)). The proposed DHT tracker even outperforms TREE in videos with stable group structures, e.g. the "Marathon" sequence shown in Fig. 8(b). Layer-2 patches and hierarchical tree structure used in DHT model more appearance features and spatial constrains than the single tree structure, which can help prevent targets from drifting (in Fig. 8(b), target 15 in the second column, target 17 in the third column, target 10 and 15 in the fourth column).



(a) “Split”



(b) “Marathon”

Fig. 8. Some tracking results in Experiment II. Red dots: centers of ground truth; cyan boxes: manually selected targets for tracking; dashed red boxes: automatically detected layer-2 patches in DHT tracker. Row 1 and 3: tracking results of DHT tracker (layer-2 patches are hidden in row 1 for better visualization); Row 2 and 4: tracking results of TREE tracker. (a): dynamic structure outperforms static structure in crowd scenes; (b): hierarchical structure could help prevent targets from drifting.

5 Conclusions

We have proposed to detect distinctive and stable patches, and jointly track them using dynamic hierarchical tree structures in order to capture crowd motions. We integrate low-level keypoint tracking, mid-level patch tracking and high-level group evolution into one united framework, in which keypoint tracking provide accurate motion information in short periods, patch tracking is more robust to partial occlusion or appearance changes, and group evolution guide the update of group structures. Experimental results show that our tracker can track targets more accurately than traditional trackers in crowd videos. The proposed dynamic hierarchical tree structure outperforms static single-tree structure.

Acknowledgement. This work is supported by the National Science Foundation of China (No.61371192, No.61103134), the General Research Fund sponsored by the Research Grants Council of Hong Kong (Project Nos. 417110, 417011, 419412), and Shenzhen Basic Research Program (JCYJ20130402113127496).

References

1. Ali, I., Dailey, M.N.: Multiple human tracking in high-density crowds. Image and Vision Computing 30(12), 966–977 (2012)
2. Ali, S., Shah, M.: Floor fields for tracking in high density crowd scenes. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 1–14. Springer, Heidelberg (2008)
3. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. PAMI 33(8), 1619–1632 (2011)
4. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. EURASIP Journal on Image and Video Processing 2008 (2008)
5. Blei, D.M., Lafferty, J.D.: A correlated topic model of science. Annals of Applied Statistics 1(1), 17–35 (2007)
6. Brostow, G.J., Cipolla, R.: Unsupervised Bayesian detection of independent motion in crowds. In: CVPR (2006)
7. Dinh, T.B., Vo, N., Medioni, G.: Context tracker: Exploring supporters and distracters in unconstrained environments. In: CVPR (2011)
8. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via on-line boosting. In: BMVC (2006)
9. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Physics Review 51(5), 4282–4286 (1995)
10. Idrees, H., Warner, N., Shah, M.: Tracking in dense crowds using prominence and neighborhood motion concurrence. Image and Vision Computing 32(1), 14–26 (2014)
11. Jing, S., Loy, C.C., Wang, X.: Scene-independent group profiling in crowd. In: CVPR (2014)
12. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. PAMI 34(7), 1409–1422 (2012)
13. Kratz, L., Nishino, K.: Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. PAMI 34(5), 987–1002 (2012)
14. Le Bon, G.: The crowd: A study of the popular mind. The Macmillan Co. New York (1897)
15. Mahadevan, V., Li, W., Bhalodia, V., Vasconcelos, N.: Anomaly detection in crowded scenes. In: CVPR (2010)
16. Moussaid, M., Garnier, S., Theraulaz, G., Helbing, D.: Collective information processing and pattern formation in swarms, flocks, and crowds. Topics in Cognitive Science 1(3), 469–497 (2009)
17. Newman, M.E.: Finding community structure in networks using the eigenvectors of matrices. Physical Review E 74(3), 036104 (2006)
18. Ouyang, W., Wang, X.: Single-pedestrian detection aided by multi-pedestrian detection. In: CVPR (2013)
19. Pellegrini, S., Ess, A., Schindler, K., van Gool, L.: You'll never walk alone: Modeling social behavior for multi-target tracking. In: ICCV (2009)

20. Rabaud, V., Belongie, S.: Counting crowded moving objects. In: CVPR (2006)
21. Rodriguez, M., Ali, S., Kanade, T.: Tracking in unstructured crowded scenes. In: ICCV (2009)
22. Rodriguez, M., Laptev, I., Sivic, J., Audibert, J.Y.: Density-aware person detection and tracking in crowds. In: ICCV (2011)
23. Rodriguez, M., Sivic, J., Laptev, I., Audibert, J.Y.: Data-driven crowd analysis in videos. In: ICCV (2011)
24. Scovanner, P., Tappen, M.F.: Learning pedestrian dynamics from the real world. In: ICCV (2009)
25. Shi, J., Tomasi, C.: Good features to track. In: CVPR (1994)
26. Shu, G., Dehghan, A., Oreifej, O., Hand, E., Shah, M.: Part-based multiple-person tracking with partial occlusion handling. In: CVPR (2012)
27. Sugimura, D., Kitani, K.M., Okabe, T., Sato, Y., Sugimoto, A.: Using individuality to track individuals: clustering individual trajectories in crowds using local appearance and frequency trait. In: CVPR (2009)
28. Tomasi, C., Kanade, T.: Detection and tracking of point features, Technical report, CMU-CS-91-132 (1991)
29. Wang, X., Ma, X., Grimson, E.: Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. PAMI 31(3), 539–555 (2009)
30. Yao, R., Shi, Q., Shen, C., Zhang, Y., van den Hengel, A.: Part-based visual tracking with online latent structural learning. In: CVPR (2013)
31. Zhang, L., van der Maaten, L.: Structure preserving object tracking. In: CVPR (2013)
32. Zhang, W., Wang, X., Zhao, D., Tang, X.: Graph degree linkage: Agglomerative clustering on a directed graph. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part I. LNCS, vol. 7572, pp. 428–441. Springer, Heidelberg (2012)
33. Zhou, B., Tang, X., Wang, X.: Measuring crowd collectiveness. In: CVPR (2013)
34. Zhou, B., Tang, X., Wang, X.: Coherent filtering: Detecting coherent motions from crowd clutters. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part II. LNCS, vol. 7573, pp. 857–871. Springer, Heidelberg (2012)
35. Zhou, B., Tang, X., Wang, X.: Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In: CVPR (2012)
36. Zhou, B., Wang, X., Tang, X.: Random field topic model for semantic region analysis in crowded scenes from tracklets. In: CVPR (2011)