Online Multi-Person Tracking by Tracker Hierarchy

Jianming Zhang, Liliana Lo Presti, Stan Sclaroff
Department of Computer Science, Boston University
111 Cummington Street, Boston MA 02215
{jmzhang,loprest,sclaroff}@bu.edu

Abstract

Tracking-by-detection is a widely used paradigm for multi-person tracking but is affected by variations in crowd density, obstacles in the scene, varying illumination, human pose variation, scale changes, etc. We propose an improved tracking-by-detection framework for multi-person tracking where the appearance model is formulated as a template ensemble updated online given detections provided by a pedestrian detector. We employ a hierarchy of trackers to select the most effective tracking strategy and an algorithm to adapt the conditions for trackers' initialization and termination. Our formulation is online and does not require calibration information. In experiments with four pedestrian tracking benchmark datasets, our formulation attains accuracy that is comparable to, or better than, the state-of-the-art pedestrian trackers that must exploit calibration information and operate offline.

1. Introduction

Tracking-by-detection is a commonly used paradigm for multi-person tracking [4, 8, 1]. In tracking-by-detection, a classifier is used to detect candidate instances of pedestrians in the current video frame; the resulting detections are linked together, frame-to-frame to reconstruct the trajectories of pedestrians across time. The underlying classifier can be trained offline [8, 6], or adapted via an online retraining mechanism [4, 19].

In practice, tracking-by-detection performance in pedestrian tracking is adversely affected by variations in crowd density, obstacles in the scene, varying illumination, human pose variation, scale changes, etc.. Some approaches employ occlusion reasoning to fill gaps in pedestrian tracks [1, 15]. However, in practice, tracking-by-detection is still vulnerable to missing and false detections.

In tracking-by-detection, the tradeoff between missing detections and false detections seems inescapable. If detector's parameters are adjusted to reduce false detections, then more "true positive" detections are missed, resulting in gaps in tracks, lost tracks, or missed tracks. If detector's parameters are adjusted to reduce missing detections, then more false detections occur, leading to false tracks, noisy tracks, or drifting tracks.

To cope with these challenges, tracking methods often incorporate dynamical models and appearance models. Dynamical models not only help with predicting the new position of a pedestrian during a "detection gap," they also can help with smoothing out noisy trajectories and improving detection association. Appearance models can also help improve data association, particularly when the models are adapted online to account for gradual changes in appearance and pose. However, dynamical models and appearance models also come with significant shortcomings. Dynamical models essentially impose priors on motion and tend to hallucinate incorrect tracks when pedestrian motion changes abruptly, e.g., turning a corner, stopping, etc. Appearance models, when updated online, inevitably diverge from some targets due to false detections, errors in data association, and abrupt changes in pose or appearance.

We propose an improved tracking-by-detection strategy for multi-person tracking whose main contributions are:

1. A new appearance model formulated as a template ensemble, which is updated online given detections provided by a pedestrian detector. An online pruning strategy maintains each ensemble and ensures that weaker templates are discarded.

2. A hierarchy of trackers to select the most effective tracking strategy. An *expert tracker* has a sufficient set of reliable templates it uses to localize the target; a *novice tracker* has fewer templates and relies more on new detections than on its template ensemble. Through time, trackers can be promoted to experts or demoted to novices based on demonstrated level of reliability.

3. An online algorithm to adapt the conditions for trackers' initialization and termination based on the consensus of the trackers that are active at a given frame.

978-0-7695-4797-8/12 $26.00 © 2012 IEEE
DOI 10.1109/AVSS.2012.51

In experiments with four pedestrian tracking benchmark datasets, our formulation attains accuracy that is comparable to, or better than, state-of-the-art pedestrian trackers. But unlike these other methods, our formulation is online and does not require calibration information.

2. Related Work

In recent years, several works combined discriminative and generative methods for tracking, *e.g.* [3, 8, 16]. In these works, offline trained detectors and standard tracking techniques are combined, and the detectors are used in a tracker's appearance/observation model.

Some works integrate the detector's response as well as motion and appearance models into a global optimization framework [2, 13, 6]. In others, a data association problem is formulated to link detections through time [7, 5]. These techniques are based on offline approaches, and require calibration information. In contrast to these approaches, we present a fast online method to track multiple pedestrians with an uncalibrated camera.

Some tracking-by-detection methods employ the mean-shift tracker [10], which searches for the detection whose appearance best matches the target via a gradient ascent procedure. In [9], mean-shift tracking is coupled with online adaptive feature selection. Features are selected to separate the target from the background; the most discriminative features are used to compute weight images. Then the mean-shift tracker is applied on each of the weight images to localize the target. Based on the new target detection, foreground/background samples are selected to compute a new set of discriminative features.

In [4], at each frame weak classifiers are trained to discriminate between pixels on the target vs. the background. Online AdaBoost is used to train a strong classifier. Mean-shift is applied on the classifier's confidence map to localize the target. In [19] several classifiers are co-trained to detect the object and reach a consensus on the predicted location.

In all these methods, the appearance model is fully dependent on the tracker's detection. Moreover, in [4, 19], the classifiers are dependent one each other and, through time, they may correlate. In our approach, we adopt the mean-shift tracker, but the appearance model consists of a pool of templates found by the pedestrian detector. The template ensemble is updated over time adding or removing templates; however, as the templates are found by the pedestrian detector and not by the tracker itself, the templates are independent of each other. Each template's appearance is not updated through time with the new detections, limiting the noise introduced in case of false detections.

Some works [1, 20, 15] integrate occlusion modeling within the tracking framework. These techniques define repulsive forces between the occluding observations, or

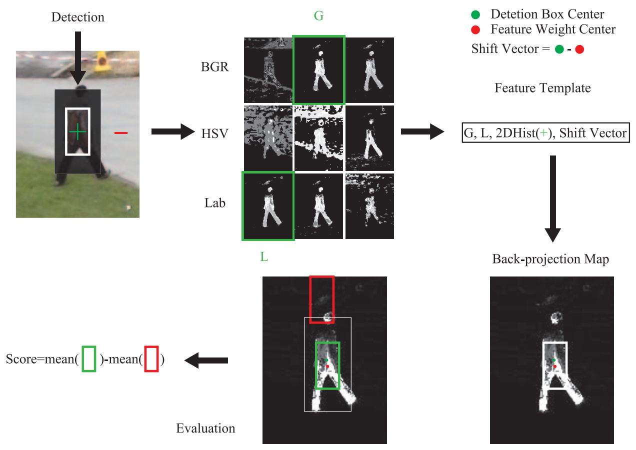

Figure 1. Feature selection and back-projection image. "+" marks the inner window b_{inner}.

define grouping behaviors. In contrast to these works, we consider that during an occlusion the appearance model may provide little information about the object position and the pedestrian behavior may be unpredictable during the gap. Therefore, we take a different approach by adaptively growing the tracker's search area to enable the target's reacquisition when the occlusion ends.

3. Tracking by Template Ensemble

At each frame I^t, a detector outputs a set of N_d detections $D^t = \{d_i^t\}_{i=1}^{N_d}$, in which d_i^t is a bounding box in frame I^t. Although a threshold for the detector's response can be manually set to filter out false detections in some specific scene, our system does not rely on this threshold to perform robust tracking.

At time t, the state of the tracker T_i is defined as $\{F_{T_i}^t, b_{T_i}^t, kf_{T_i}^t\}$, where the template ensemble $F_{T_i}^t = \{f_i^j\}_{j=1}^{N_i^t}$ is the set of N_i^t templates collected across time, $b_{T_i}^t = [xc_i^t, yc_i^t, w_i^t, h_i^t]$ defines the current estimate of the target's bounding box centered at (xc_i^t, yc_i^t) and of size (w_i^t, h_i^t), and kf_i is the Kalman filter used to model the tracker's dynamics. The Kalman filter's state is defined as the tracker's 2D center point position and the 2D velocity $(x, y, \dot{x}, \dot{y})$. In our implementation, the Kalman filter's process and measurement noise covariance matrices are set to $(\frac{w_i^t}{r_f})^2 \cdot diag(0.025, 0.025, 0.25, 0.25)$ and ${w_i^t}^2 \cdot diag(1, 1)$ respectively, where r_f is the frame rate of the sequence.

3.1. Template Ensemble

Every time a detection is associated to a target, a template is added to the corresponding template ensemble. A template is defined as $f := \{channel[2], H, v_{shift}\}$, where $channel[2]$ represents two color channels selected to represent the template, H is the 2D histogram for the two

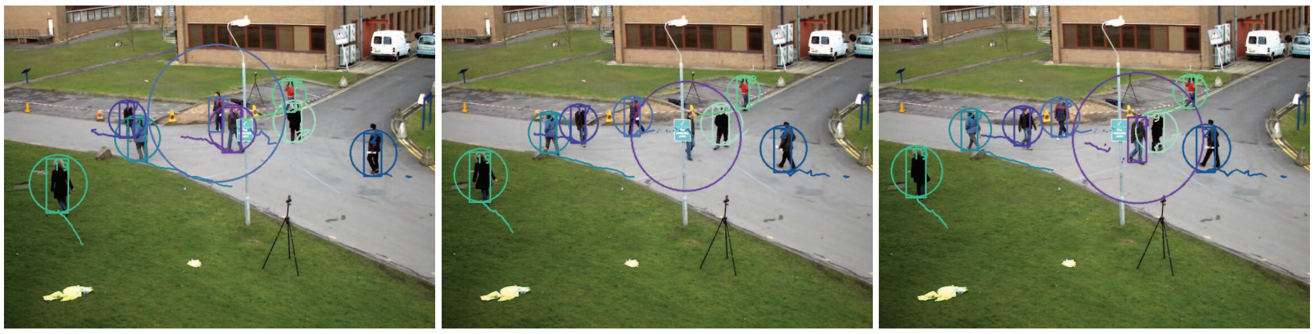

Figure 2. Short-term Occlusion Handling. Circles represent the search area for each tracker. When the Expert tracker with the purple box crosses the pole, it loses templates due to the occlusion. Then it degrades to Novice and expands its search area (middle frame). When a near detection is matched with it again, it recovers the tracking correctly (final frame).

selected channels and v_{shift} is a 2D shifting vector. Once the template is created, it will not be updated afterwards.

The two color channels are selected – among the 9 from RGB, HSV and Lab color spaces – to yield separation of pixels within foreground and background. To select these channels, we follow [9], with several changes made to better suit the pedestrian tracking problem. As shown in Fig. 1, a central patch (inner window) inside the detection window (outer window) is used as the foreground area, while the ring region surrounding the detection window is taken as background. The best two channels are selected based on the variance ratio between the background and foreground's feature weight distributions (see [9] for details).

The histogram H of the selected color channels is calculated inside the inner window and used to compute the back-projection image BP for the region of interest. As illustrated in Fig. 1, the distribution of pixel values in the back-projection image may not be evenly distributed on the person's body. When we apply mean-shift tracking on the back-projection image, the tracking window shifts to the densest area, causing a bias in position with respect to the object's center. The vector v_{shift} accounts for such bias so that, during mean-shift tracking, the feature template is shifted back to the center of the body.

For maintaining the template ensemble, a score is computed for each template based on its back-projection image given the updated target's position from the tracker. The tracker maintains at most N_{max} templates by discarding all the templates with negative scores, and the templates with lowest scores. The score is computed as follows:

$$Score(f) = \underset{inner}{\text{mean}}\, BP(f) - \max_{r \subset R}\{\underset{r}{\text{mean}}\, BP(f)\}, \quad (1)$$

where $\text{mean}_{inner}(\cdot)$ is the mean pixel value inside the inner window b_{inner} on $BP(f)$; the second term takes the mean pixel value in the patch r whose size is the same of b_{inner} and whose mean value is the highest within the ring area R. $Score(\cdot)$ measures the difference between foreground and distracting areas in the background for the given feature template's back-projection.

3.2. Tracking

Tracking is performed by alternating between mean-shift tracking (to take into account the target's appearance) and Kalman filtering (to take into account the target's motion dynamics). Alg. 1 summarizes the main steps needed to perform tracking. At each frame, each template of tracker T_i is used to compute a back-projection image BP. The set of computed BP images is then averaged to provide a *voting map*. A mean-shift tracker is applied to the voting map to estimate the new target's position. This position is then used as the measurement input to the Kalman filter kf_i, which will predict the position for T_i to start the mean-shift tracker at the next frame. Trackers adopt different update strategies based on the template ensemble characteristics as described in Sec. 4.

Alg. 1 $T_i^t = Tracking(T_i^{t-1}, I^t, O^t)$

Input: Tracker T_i^{t-1}; Current frame I^t; Occupancy map for current frame O^t
Output: Updated tracker T_i^t

```
1:  Calculate the back-projection map for each feature template of the tracker
2:  Compute the voting map by summing up the back-projection maps
3:  Look up in occupancy map $O^t$ and set occupied pixels to 0 in the voting map to
    avoid coalescence (see Sec. 6)
4:  Predict the tracker's starting position for mean-shift using Kalman filter
5:  Update the tracker's position by mean-shift tracking on the voting map
6:  Register the pixels it occupies on occupancy map $O^t$ for subsequent trackers
7:  if $T_i^{t-1}$ is a Novice then
8:      Replace $kf_i$'s posterior state with current position and previous velocity
9:  else
10:     Correct $kf_i$ using current position by Kalman gain
11: Calculate the scores for tracker'feature templates by Eq. 1 and discard weak
    templates
12: $T_i^t \leftarrow T_i^{t-1}$
```

4. Tracker Hierarchy

We assume that a tracker's confidence may be quantified as the number of templates it possesses. Trackers are divided into two groups:

1. *Experts* are trackers having more than K templates. These trackers have high confidence and may use their template ensemble to perform tracking;

2. *Novices* are trackers having fewer than K templates. These trackers have low confidence and during tracking rely more on the newly assigned detections than on their template ensemble.

We also consider a group of *candidate trackers*, which are trackers waiting for enough matched detections before being accepted as novice. Once a novice is initialized, it tries to accumulate templates that may robustly track the target. After K templates have been accumulated, a novice is promoted to expert. Conversely, an expert that loses templates is demoted to novice; this usually occurs when the target undergoes occlusions or appearance changes. A novice will retain its last template if all the templates' scores are negative. In our implementation (N_{max}, K) are set to $(10, 5)$.

4.1. Detection Association and Tracking Strategies

Novices and experts apply different tracking strategies, i.e. across time they update the dynamical model and the search area in different ways.

Given the set of detections D^t, the assignment of the detections to the trackers is formulated as an assignment problem and it is solved by finding the maximum matching in a bipartite graph. In a way similar to that described in [12], we used the Hungarian Algorithm [18] with the cost matrix

$$c_{ij} = \begin{cases} dist_{ij}, & dist_{ij} < \mathcal{R}_i \\ \infty, & otherwise \end{cases} \tag{2}$$

where $dist_{ij}$ is the distance between the current estimated target position and the center of the detection d_j^t, while $\mathcal{R}_i = \alpha_i w_i$ depends on the width w_i of the target's bounding box. α_i controls the extension of the search area for new detections. In our implementation, $\alpha_\mathbf{i}$ is calculated as $\alpha_i = \min(\phi_1 \frac{\sigma_{kf_i}}{w_i} + \phi_2, \alpha_{max})$, where σ_{kf_i} is the square root of the scale of the posteriori covariance of kf_i, w_i is the width of the bounding box, (ϕ_1, ϕ_2) are constants and α_{max} is an upper bound. We set $(\phi_1, \phi_2, \alpha_{max})$ to be $(0.5, 1.5, 4)$.

While an expert corrects its Kalman filter at each step, a novice only updates the posterior state of its Kalman filter without shrinking the posterior covariance (see Alg. 1). This mechanism is similar in spirit to the adaptive Kalman filter [17]. Our method can be interpreted as switching between two modes of measurement noise covariance in Kalman filter by thresholding the number of templates a tracker has. In this way, a novice has a growing search area and, therefore, it has more chances to be associated with a detection even though it may lose its track for a short time. When a new detection $(x_{det}, y_{det}, w_{det}, h_{det})$ is associated to a novice T_i, the posterior state of kf_{T_i}, $(x, y, \dot{x}, \dot{y})$, is replaced by $(x_{det}, y_{det}, \dot{x}, \dot{y})$. In this way, the novice jumps to the position of the newly matched detection. This allows a novice to recover its tracking efficiently after a short-term occlusion or change of appearance (see Fig.2).

The detection association process has three steps. First, detections are matched to the experts. Then, remaining detections are matched to the novices. Finally, detections that are not associated to experts or novices are matched with candidates in the waiting list (in this case α_i is set to 1.5). If any detections remain unassigned after these steps, then these are added to the waiting list. Thus, to initiate a novice a minimum number of matching observations must be found. The condition to initiate a novice tracker is learned adaptively as described in Sec. 5.

5. Trackers' Birth and Death

The initialization or termination of a tracker is determined based on the tracker's matching rate τ_i and average matching rate $\bar{\tau}$ among all the established trackers. A

Alg. 2 $(\mathbf{T}^t, W_L^t) = Multi\_track(\mathbf{T}^{t-1}, \mathbf{D}^t, I^t, W_L^{t-1})$

Input: Tracker group $\mathbf{T}^{t-1} = \{T_i^{t-1}\}_{i=1}^n$; Detections $\mathbf{D}^t = \{d_j^t\}_{j=1}^m$; Current frame I^t; Waiting list W_L^{t-1}
Output: Updated tracker group $\mathbf{T}^t$; Updated waiting list W_L^t

```
1. Tracking:
Create a new occupancy map $O^t$ and set it to 0 (see Sec. 6)
for $T_i^{t-1}$ in $\mathbf{T}^{t-1}$
    $T_i^t = Tracking(T_i^{t-1}, I^t, O^t)$ by Alg. 1
$\mathbf{T}^t \leftarrow \mathbf{T}^{t-1}$

2. Detection Association:
Associate $\mathbf{D}^t$ with experts and add feature templates accordingly
Associate rest of $\mathbf{D}^t$ with novices and add feature templates accordingly
Associate rest of the detections with candidates in $W_L^{t-1}$
Use the remaining detections to activate new candidates and insert in $W_L^{t-1}$

3. Initialization
for candidate $C_i \in W_L^{t-1}$
    if $C_i$ has detection rate more than $\theta_{init}$
        Initialize $C_i$, and move it to $\mathbf{T}^{t-1}$
$W_L^t \leftarrow W_L^{t-1}$

4. Termination
for $T_i^t \in \mathbf{T}^t$
    if $T_i^t$ has detection rate less than $\theta_{term}$
        remove $T_i^t$ from $\mathbf{T}^t$

5. Promotion and Demotion:
Promote novices if their numbers of templates are greater than or equal to $K$
Demote experts if their numbers of templates are less than $K$

6. Matching Rate Update:
Update $\bar{\tau}$ and $\tau_i$ by Eqs. 4 and 3
```

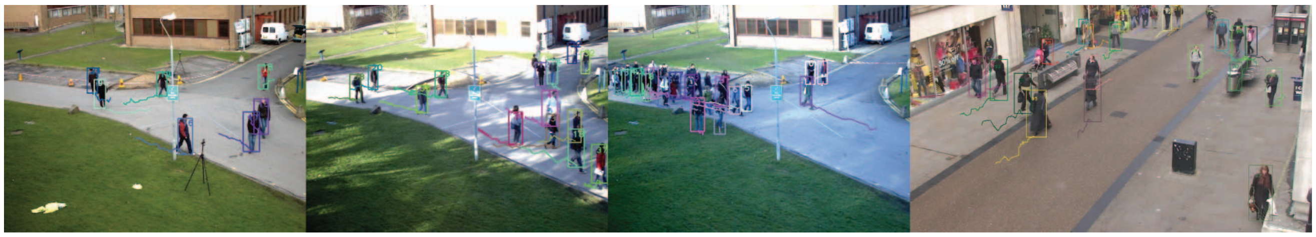
Figure 3. Example tracking results for PETS09 and Town Center datasets.

tracker's matching rate is defined as

$$\tau_i = \frac{\Delta N_i^{matched}}{\Delta t}, \quad (3)$$

where $\Delta N_i^{matched}$ is the number of detections matched with T_i in a temporal sliding window of length Δt. Considering that the detections have a Poisson distribution, the average matching rate among all the established trackers is estimated by

$$\bar{\tau} = \frac{\alpha + \sum_{i=0}^{N_{trackers}} \Delta N_i^{matched}}{\beta + \Delta t \cdot N_{trackers}} \quad (4)$$

where (α, β) are the parameters the Poisson distribution (in our implementation (α, β) are set to $(30, 5)$).

Once a tracker candidate is activated, it will stay in the waiting list until its detection τ_i is above the threshold:

$$\theta_{init} = \bar{\tau} - \gamma_1 \sqrt{\bar{\tau}}, \quad (5)$$

where γ_1 is a scale factor controlling the confidence interval. The waiting list has a maximum length and when this length is reached the oldest candidates are removed.

A tracker will be terminated when its detection rate is less than

$$\theta_{term} = \bar{\tau} - \gamma_2 \sqrt{\bar{\tau}}, \quad (6)$$

where γ_2 is a scale factor similar to γ_1. In this way, the conditions for tracker's termination and initialization adapt to different recall rates of the given pedestrian detector. In our implementation, (γ_1, γ_2) are set to $(1, 2)$. Alg. 2 summarizes our formulation.

6. Implementation Details

Occupancy Map: To prevent multiple trackers from sticking to the same object, a simple 2D occupancy map is employed. For a single frame, the areas occupied by trackers are recorded one by one in the occupancy map and each tracker sets to zero the pixels in the voting map that are already occupied by other targets.

Detection Filtering: Before being associated, the detections are filtered out based on their height. In our system, a body height projection map is learned online. We divide the frame into (8×8) cells and estimate online the mean μ_{ij} and standard deviation σ_{ij} of the heights of expert trackers' bounding boxes in each cell c_{ij}. Then we calculate

$$[\mu_{ij} - \eta_{ij}, \mu_{ij} + \eta_{ij}] \quad \text{with} \quad \eta_{ij} = k_1 \frac{\sigma_{ij}}{N_{samples}} + k_2 \sigma_{ij},$$

as the confidence interval for the height of the detections' bounding box. (k_1, k_2) are parameters controlling the confidence interval; in particular, k_1 determines how fast the confidence interval shrinks with the increase of the sample number. (k_1, k_2) are set to $(10, 3)$ in our implementation. In this way, many false detections are filtered out without the need for camera calibration information.

Scale adaptation: When a tracker is matched with a detection by the method described in Sec. 4.1, it updates the size of its bounding box (w_i^t, h_i^t) with a learning rate λ_s, *i.e.* $(w_i^t, h_i^t) = \lambda_s * (w_{det}, h_{det}) + (1 - \lambda_s) * (w_i^{t-1}, h_i^{t-1})$, where (w_{det}, h_{det}) is the window size of the detection. λ_s is set to 0.4 in our implementation.

7. Experimental Evaluation

We implemented the system using C++ and the OpenCV library and tested it on a computer with a dual core 3.2GHzX2, 12GB memory. A software implementation of our tracker is available for public download[1].

We tested our method on four publicly available sequences: three from the PETS 2009 dataset S2[2] and one called "Town Center"[3]. For the PETS09 data set, we use "View 1" of sequences S2.L1, S2.L2 and S2.L3. These sequences have different levels of difficulty based on the density of the moving crowd, and the ground-truth used for evaluation is publicly available[4]. The Town Center sequence is much longer than the PETS sequences. The crowd density changes across time from sparse to dense. The ground-truth for evaluation on this sequence is publicly available[2] (see Table 1). The runtime performance is about 7 fps for PETS09 data set (768×576) and 0.8 fps for the Town Center sequence(1920×1080) given the detections.

[1] http://www.cs.bu.edu/groups/ivc/software/TrackerHierarchy/
[2] http://www.cvg.rdg.ac.uk/PETS2009
[3] http://www.robots.ox.ac.uk/ActiveVision/Research/Projects/2009bbenfold_headpose/project.html
[4] http://www.gris.informatik.tu-darmstadt.de/~aandriye/data.html

Table 1. Dataset description

	Frame Rate	N. of frames	N. of Id.	HOG OpenCV[11] Precision	HOG OpenCV[11] Recall	From [1] or [6] Precision	From [1] or [6] Recall
`PETS09 S2.L1`	7	795	19	94.75%	82.22%	75.14%	87.81%
`PETS09 S2.L2`	7	436	43	90.48%	51.62%	72.55%	55.91%
`PETS09 S2.L3`	7	240	44	88.56%	39.08%	90.48%	51.62%
`TownCenter`	25	4500	230	89.08%	62.46%	84.45%	74.16%

Table 2. Methods Comparison

	calibration	batch	multi-camera	appearance
Our	NO	NO	NO	YES
[1]	YES	YES	NO	NO
[6]	YES	NO	NO	NO
[5]	YES	YES	YES	YES
[15]	YES	YES	NO	NO

To measure performance we employed the CLEAR MOT metrics described in [14]. The Multiple Object Tracking Accuracy (MOTA) considers the number of missed detections (MS), the number of false positives (FP), and the switches of identities (ID Sw.). The Multiple Object Tracking Precision (MOTP) considers the precision of the detections and it is computed using the intersection over union (IOU) metric for 2D bounding boxes. Note that higher values of these metrics indicate better performance.

As summarized in Table 3, we compare our method with [1] and [6] on the PETS09 and Town Center data sets respectively. Tracking results for [1] on the PETS sequences were provided to us by the authors, and tracking results of [6] on the Town Center sequence are available on the same web page of the Town Center data set. For fair comparison, we adopted the same evaluation protocol and detector's output: for the PETS09 data set, we use the same detections used in [1]; in the Town Center data set we use the same detections used in [6]. Note that the results of [1, 6] are slightly different from the reported results in the original papers, because we have re-evaluated them using the same 2D matching protocol. We also report the tracking results based on a generic HOG pedestrian detector [11] from the OpenCV Library. We also show the results reported in [5] and [15] when available.

Table 1 compares the performance of the pedestrian detectors used for testing. The detector provided by [1, 6] gives a higher recall rate but lower precision rate than the OpenCV HOG detector in most sequences, which means there are more false positives in the provided detections. Without using calibration information, our system successfully suppresses the false detections and gives similar MOTA scores (see Table 3). Thus, for these data sets, the choice of pedestrian detector did not greatly affect the performance of our system.

As shown in Table 2, all the methods we compare with, except [6], require calibration information and perform batch processing of the data, while our method is fully online. However, our method achieves higher MOTA scores than most of them. On the PETS09 data set, our system achieves higher MOTA scores in S2L1 and S2L2, and slightly lower but comparable MOTA score in S2L3 compared with [1] based on the same detections. Note that in [1], tracking is performed only in a central cropped area, whereas we track all the pedestrians in the scene; this potentially increases the risk of more missing detections and false detections. We believe that the slightly lower MOTP scores are due to the fact we neither use camera calibration information nor employ off-line smoothing or pruning of trajectories (as in [1]). On the Town Center sequence, our system produces fewer false positives than [6] (about 40% less) using the same detections. Thus, our method outperforms the online method [6] in terms of MOTA score on this dataset.

Table 3. Results and Comparison

		MOTA	MOTP	MS	FP	ID Sw.
PETS S2L1	Our+det.[1]	93.27%	68.17%	162	132	19
	Our+det.(HOG)	90.75%	68.64%	360	56	14
	[1](cropped)	91.71%	74.47%	285	32	11
	[5]	81%	-	-	-	-
	[15]	67%	-	-	-	-
PETS S2L2	Our+det.[1]	66.72%	58.21%	2963	247	215
	Our+det.(HOG)	64.12%	58.66%	3394	106	193
	[1](cropped)	60.17%	63.02%	3169	54	104
PETS S2L3	Our+det.[1]	40.38%	56.41%	2507	22	80
	Our+det.(HOG)	39.67%	57.75%	2477	96	67
	[1](cropped)	43.37%	59.77%	1814	9	23
Town Center	Our+det.[6]	73.61%	68.75%	12689	5746	421
	Our+det.(HOG)	72.30%	66.35%	16430	3176	188
	[6]	69.73%	69.68%	11886	9313	402
	[15]	67.30%	71.50%	-	-	86

We examined the ratio of expert trackers for each experimental sequence. The ratio was around 90% for the PETS09S2L1 and Town Center sequences, whereas the ratio was lower for the other two sequences: 56% for PETS09S2L2 and 60% for PETS09S2L3. As expected, the ratio of experts decreases when there are more occlusions and/or greater densities of people within the observed scene.

Finally, experiments were conducted to evaluate the formulation's sensitivity to the parameter settings. Each parameter is varied within the range of $\pm 40\%$ from the baseline parameter setting while keeping the other parameters fixed. The corresponding MOTA and MOTP scores are shown in the graphs of Fig. 4. The changes of parameters have varied levels of influence on different sequences. This is because these sequences have different characteristics like crowd density, pedestrians' dynamics, *etc*. The performance of the system varies within a reasonable range for all four data sets, which indicates that our system is relatively robust to the setting of parameters that need to be set a priori.

8. Conclusion and Future Work

We propose a new appearance model that is formulated as template ensemble. In experimental evaluation, the proposed ensemble formulation tends to be robust to noise

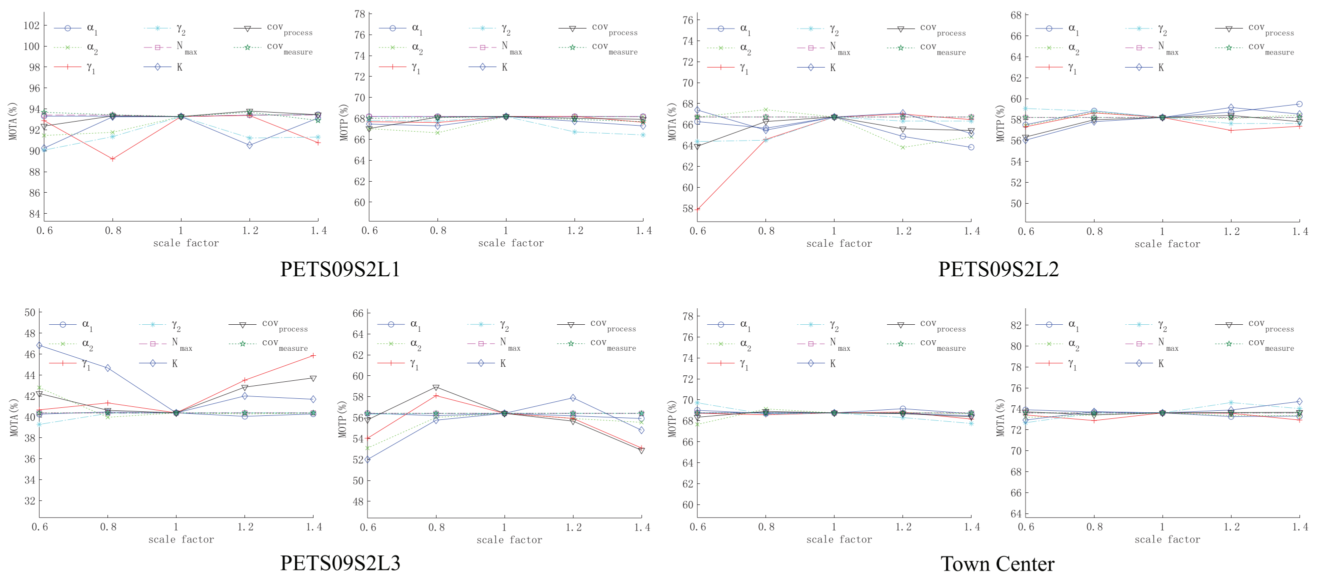

Figure 4. Influence of parameters on system performance on different data sets. Each parameter is varied within the range of $\pm 40\%$ from the baseline parameter setting while keeping the other parameters fixed. The parameters $cov_{process}$ and $cov_{measure}$ are scale factors for the process and measurement noise of the Kalman filter respectively.

and false detections. We formulate multi-person tracking using a tracker hierarchy. Trackers are classified as experts or novices based on the number of templates they have, and apply different strategies for adapting their dynamical models and search areas for matching new detections. False detections languish in the waiting list and expire if they do not gain sufficient support to be promoted to novice. In our experiments, we have found that this tracker hierarchy provides robustness to missing and false detections.

In future work, a model of individuals' interactions can also be incorporated in our formulation to help account for longer-term occlusions. We also expect that our approach can be extended for use in multi-camera systems.

9. Acknowledgments

This work was supported in part through grants from the US National Science Foundation #1029430, #0910908, and #0855065.

References

[1] A. Andriyenko, S. Roth, and K. Schindler. An analytical formulation of global occlusion reasoning for multi-target tracking. *ICCV Workshop*, 2011.

[2] A. Andriyenko and K. Schindler. Multi-target tracking by continuous energy minimization. *CVPR*, 2011.

[3] S. Avidan. Support vector tracking. *PAMI*, 26(8):1064–1072, 2004.

[4] S. Avidan. Ensemble tracking. *PAMI*, pages 261–271, 2007.

[5] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua. Tracking multiple people under global appearance constraints. *ICCV*, 2011.

[6] B. Benfold and I. Reid. Stable multi-target tracking in real-time surveillance video. *CVPR*, 2011.

[7] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *PAMI*, 33(9), 2011.

[8] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *PAMI*, 33(9):1820 –1833, 2011.

[9] R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, pages 1631–1643, 2005.

[10] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. *CVPR*, 2, 2000.

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 1, 2005.

[12] T. Huang and S. Russell. Object identification in a bayesian context. *IJCAI*, 15:1276–1283, 1997.

[13] H. Jiang, S. Fels, and J. Little. A linear programming approach for multiple object tracking. *CVPR*, 2007.

[14] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *PAMI*, 31(2):319–336, 2009.

[15] L. Leal-Taixe, G. Pons-Moll, and B. Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. *ICCV Workshop*, 2011.

[16] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: a cascade particle filter with discriminative observers of different life spans. *PAMI*, pages 1728–1740, 2008.

[17] R. Mehra. On the identification of variances and adaptive kalman filtering. *Automatic Control, IEEE Transactions on*, 15(2):175–184, 1970.

[18] J. Munkres. Algorithms for the assignment and transportation problems. *J. of the Society of Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[19] F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. *ICCV*, 2007.

[20] X. Zhang, W. Hu, W. Qu, and S. Maybank. Multiple object tracking via species-based particle swarm optimization. *IEEE TCSVT*, 20(11):1590–1602, 2010.