Tracking Human Under Occlusion Based on Adaptive Multiple Kernels With Projected Gradients

Chun-Te Chu, Jenq-Neng Hwang, *Fellow, IEEE*, Hung-I Pai, and Kung-Ming Lan

***Abstract*—Kernel based trackers have been proven to be a promising approach for video object tracking. The use of a single kernel often suffers from occlusion since the available visual information is not sufficient for kernel usage. In order to provide more robust tracking performance, multiple inter-related kernels have thus been utilized for tracking in complicated scenarios. This paper presents an innovative method, which uses projected gradient to facilitate multiple kernels, in finding the best match during tracking under predefined constraints. The adaptive weights are applied to the kernels in order to efficiently compensate the adverse effect introduced by occlusion. An effective scheme is also incorporated to deal with the scale change issue during the object tracking. Moreover, we embed the multiple-kernel tracking into a Kalman filtering-based tracking system to enable fully automatic tracking. Several simulation results have been done to show the robustness of the proposed multiple-kernel tracking and also demonstrate that the overall system can successfully track the video objects under occlusion.**

***Index Terms*—Kalman filter, kernel-based tracking, mean shift, projected gradient.**

I. Introduction

RECENTLY, the development of intelligent surveillance systems has attracted much of people's attention. Tracking of video objects is one of the major issues in video surveillance systems. To analyze the behavior of a specific person, the target's position needs to be correctly located in consecutive frames. Several approaches have been proposed to deal with the tracking problems. The following briefly shows the three major categories of tracking methods [1]:

1) Point tracking: The target is expressed as a point in the frame, and the previous target state is utilized to make the association between targets and points. Kalman filter and particle filter are widely used in this category.
2) Kernel tracking: Targets are tracked by computing the motion of the kernels which represent the appearance of the targets. Mean shift tracker is a kind of kernel tracking.
3) Silhouette tracking: Given the target model, the target is tracked by matching the contour region in each frame. This approach is usually adopted when the contour or the shape of the target is required.

Among the above tracking algorithms, kernel-based object tracking has recently gained more popularity for better and more robust tracking performance due to its fast convergence speed and relatively low computation. The basic idea of the kernel-based tracking is to minimize the difference between the target and the candidate appearance models, which are constructed by spatially masking the object with a kernel [2]. However, when the occlusion occurs, the relevant information of the target in the frame may be insufficient to locate the target, thus the tracker tends to lose the target. This paper focuses on dealing with the occlusion problem, which is one of the most challenging issues in tracking. During the occlusion, single kernel tracking is not reliable for locating the object, and hence, multiple-kernel tracking[1] should be employed. We present a projected gradient based multiple-kernel tracking scheme that uses multiple kernels to represent the target. Besides the individual tracking within each kernel, kernels are further controlled by some predefined constraints such as the geometrical relationship among them. Moreover, in order to track multiple objects simultaneously and automatically, we combined the multiple-kernel tracking with a Kalman filter tracking system. The resulting system is fully automatic and does not need any manual intervention.

The main contributions of this paper are:

1) The projected gradient optimization is employed to enable multiple kernels to find the best match of the tracked target under predefined constraints. The mean shift vector is also utilized for the efficiency enhancement.
2) Since not all the kernels are reliable due to occlusion, different weights are assigned to the kernels based on their similarity.
3) We effectively use the gradient of the density estimator with respect to the kernel bandwidth to update the scale of the object.

This paper proceeds by introducing related work in Section II. Then the introduction to our proposed multiple-kernel tracking module is presented in Section III, followed by the experimental results and discussion in Sections IV and V, respectively. Finally, the conclusion is given in Section VI.

Manuscript received February 20, 2012; revised September 17, 2012 and December 11, 2012; accepted January 08, 2013. Date of publication June 06, 2013; date of current version October 11, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Shin'ichi Satoh.

C.-T. Chu is with the Information Processing Lab, Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: ctchu@uw.edu; randyctchu@gmail.com).

J.-N. Hwang is with Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: hwang@uw.edu).

H.-I. Pai and K.-M. Lan are with Triple Domain Vision Co. Ltd., Hsinchu, Taiwan (e-mail: hipai@tdv.com.tw; blue@tdv.com.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2013.2266634

[1]Note that the multiple-kernel in this paper is different from the multiple kernel learning in machine learning area such as [29], [30].

II. Related Work

Since a thorough discussion on the general visual tracking is beyond the scope of this paper, in this section we only review the most relevant ones focusing on the kernel-based tracking.

The mean-shift method, one of the most popular kernel-based tracking methods, was applied to the tracking problems to find the most similar location around the local neighborhood area [2], [3]. Collins [4] used the difference of Gaussian and Lindeberg's theory to track the objects through the scale space. A sample-based similarity measurement combined with a fast Gaussian transform was proposed in [5] to fulfill the mean shift procedure during tracking. In Yilmaz's approach [6], they employed the asymmetric kernels that can adaptively change the scale and orientation to track the target. Other information, such as boundary cues, was also utilized in the kernel-based tracking systems [7], [12]. Although the scale change issue has been considered in some of the above works [2], [4], [6], extra computation, such as using an additional kernel, is required. In contrast, our proposed scale update method in this work saves some computation by exploiting the by-products from the tracking procedure, which is more intuitive and has been demonstrated effective through the experiments.

All the above works only used a single kernel during tracking, and it would fail under occlusion because of the ill-observation of the object. In order to better represent the tracked video object, multiple kernels have also been adopted these years. A different similarity measure was exploited in [8], where they used Newton-style iterations to minimize the sum of squared difference (SSD) measure. In [9], the video object represented by multiple kernels, which denoted various body parts, were tracked by using a two-step approach. The global movement and local movement were alternatively applied to locate the target. Porikli *et al.* [10] used the multiple kernels centered at the high motion areas to enhance the performance when the tracked targets have fast motion. In [17], a fragments-based multiple-kernel tracking scheme was presented. They divided the target into several fragments, and each fragment was tracked by a kernel. However, none of the above utilized the inter-relationship among kernels, which can provide useful information for tracking. In [31], they proposed a feature representation which considered multiple kernel centers. It implicitly embedded the geometrical information into the feature representation, but the relationships between the kernels were not explicitly constrained. Moreover, the adaptive weights for different kernels could not be imposed. Fan *et al.* [11] linked the multiple collaborative kernels by using predefined constraints. In their approach, the tracking was formulated as solving a least square problem. It may work in normal condition in which there is little occlusion or none, but the result tends to deviate from the optimum easily under occlusion since the minimum cost value is no longer a small value. Moreover, the reliability weightings for different kernels were not considered, resulting in an unstable system when kernels are in great conflict. Additionally, their Gauss-Newton style iteration is limited to a least square problem formulation and may not be easily generalized for other kinds of cost functions such as K-L distance. Hence, we propose a projected gradient based multiple-kernel tracking to overcome the above problems [13].

III. Multiple-Kernel Tracking

To achieve the successful tracking, the target needs to be located in consecutive frames. Assume the target model is known, and we can extract the candidate model at each location in a video frame. If we define the similarity measure between these two models, the objective is to find the candidate model that has the highest similarity.

A. Single Kernel Tracking

In conventional kernel based tracking [2], a model is represented as the probability density function in the feature space, i.e., the histogram. During the histogram extraction, the amount of the contribution of a pixel is controlled by the value of a kernel function, and the value is determined based on the distance between the pixel and the kernel center. In [2], the tracking procedure for maximizing the similarity is formulated as finding $\mathbf{x}$ that maximizes the density estimator $f(\mathbf{x})$

$$f(\mathbf{x}) = \frac{\sum_{i=0}^{N_h} \omega_{\mathrm{i}} k\left(\left\|\frac{\mathbf{x}-\mathbf{z_i}}{h}\right\|^2\right)}{\sum_{i=0}^{N_h} k\left(\left\|\frac{\mathbf{x}-\mathbf{z_i}}{h}\right\|^2\right)}, \tag{1}$$

where $\mathbf{x}$ is the center of the kernel, which is normally the centroid of the object; $\mathbf{z_i}$ is the pixel location to be considered and N_h is the number of pixels; ω_{i} is the weight for each pixel; $k(\cdot)$ is the kernel function which is usually a monotonously decreasing function. The mean shift procedure is further adopted to efficiently find the optima.

B. Multiple-Kernel Cost Formulation

Alternatively, we can reformulate the problem from maximizing the similarity to minimizing the cost function, which can be designed to be inversely proportional to the similarity as shown in (2),

$$J(\mathbf{x}) \propto \frac{1}{simi(\mathbf{x})}, \tag{2}$$

where $simi(\mathbf{x})$ is the similarity function at the location $\mathbf{x}$ in the state space domain.

If we use a single kernel to track the object, the mean-shift tracking can be adopted. However, when the target is occluded, the error may occur. This can be avoided by applying multiple kernels as shown in Fig. 1, where a kernel is expressed as an ellipse. If occlusion happens, the tracking performance can be severely affected since the kernel 1 incorporates a large portion of irrelevant information (obstacle) to the target in Fig. 1(a). However, once an additional kernel is added in Fig. 1(b), although kernel 2 is nearly non-observable, the well-observed kernel 1 can still be used to compensate the adverse effect resulted from the occlusion after some constraints which link the two kernels are introduced. Hence, for N multiple kernels we define the total cost function $J(\mathbf{x})$ to be the sum of the N individual cost functions $\{J_i(\mathbf{x})\}$,

$$J(\mathbf{x}) = \sum_{i=1}^{N} J_i(\mathbf{x}). \tag{3}$$

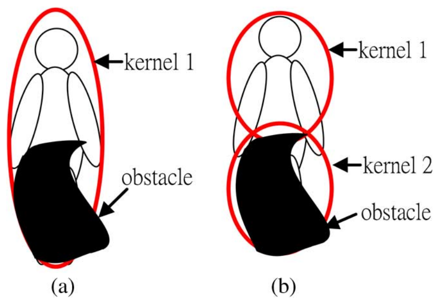


Fig. 1. Red ellipses represent kernels. (a) Single kernel with occlusion. (b) Two kernels with occlusion.

In addition to the cost function, the constraint functions $\mathbf{C}(\mathbf{x}) = \mathbf{0}$ need to be imposed on. The constraint functions confine the kernels based on their inter-relationships. For instance, if the object in Fig. 1 is a rigid body, the kernel 1 will be always on top of the kernel 2 (see Section IV for our design). Hence, the problem would become searching the state $\hat{\mathbf{x}}$, such that

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} J(\mathbf{x}) \qquad \text{subject to } \mathbf{C}(\mathbf{x}) = \mathbf{0}. \tag{4}$$

C. Projected Gradient-Based Multiple-Kernel Tracking

Starting from the state in the previous frame, in order to efficiently decrease the total cost function and keep the constraints satisfied during the state search, the movement vector $\delta\mathbf{x}$ is determined based on the projected gradient method [14], which is used to iteratively solve the constrained optimization problem. The intuitive idea is to project the gradient vector onto the space in which the values of the constraint functions remain unchanged. Moreover, in order to further handle the constraints, the second term, which guarantees the satisfaction of the constraints (i.e., ensuring $\mathbf{C}(\mathbf{x}) = \mathbf{0}$), is introduced (see Appendix A). Hence, $\delta\mathbf{x}$ consists of two components $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$:

$$\begin{aligned}\delta\mathbf{x} &= \alpha\left(-\mathbf{I}+\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^T\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^T\right)\mathbf{J}_{\mathbf{x}}+\left(-\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^T\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}(\mathbf{x})\right) \\ &= \delta\mathbf{x}_A + \delta\mathbf{x}_B,\end{aligned} \tag{5}$$

where $\mathbf{x} \in \mathcal{R}^{\mathrm{n}}$ is the state vector; $\mathbf{C}(\mathbf{x}) = \begin{bmatrix} C_1(\mathbf{x}) \\ \vdots \\ C_m(\mathbf{x}) \end{bmatrix}$ consists of m constraint functions, and $c_i(\mathbf{x}) : \mathcal{R}^{\mathrm{n}} \to \mathcal{R}$ is the $i-th$ constraint function; $\mathbf{C}_{\mathbf{x}} \in \mathcal{R}^{n\times m}$ is the gradient matrix of constraint functions with respect to $\mathbf{x}$; $\mathbf{J}_{\mathbf{x}}$ is the gradient vector of the total cost function with respect to $\mathbf{x}$, and $\alpha > 0$ is the step size.

The iterative projected gradient search formulation has the following characteristics, which are proved in Appendix B:

(i) $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$ are orthogonal to each other.

(ii) Moving along the $\delta\mathbf{x}_A$ decreases the total cost function $J(\mathbf{x})$ and keeps the values of the constraint function vector $\mathbf{C}(\mathbf{x})$ intact.

(iii) Moving along the $\delta\mathbf{x}_B$ can lower the absolute values of the constraint function vector $\mathbf{C}(\mathbf{x})$.

Based on the above characteristics, starting from the initial point in each frame, we evaluate $J(\mathbf{x})$ and $\mathbf{C}(\mathbf{x})$ and then interchangeably apply $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$ to decrease the cost function while satisfying the constraints in each iteration. We stop the iteration until either the cost function and the absolute values of the constraint functions are both below some given thresholds ε_J and $\boldsymbol{\varepsilon}_{\mathbf{C}}$ or the iteration count exceeds T (see Algorithm 1). Fig. 2 is a typical example for how the algorithm proceeds within one frame. Fig. 2(a) shows the initial state, which is inherited from the final state in the previous frame, and Fig. 2(b)–(d) are the subsequent iterations. The two constraint functions specify the geometrical relationship between two kernels, as given in (12) and (13). Based on the values of the cost function and the constraint functions (Fig. 2(e)), $\delta\mathbf{x}_A$ is applied and according to the characteristic (ii), the values of the constraint function remain as 1 and 0 through the iteration until the termination condition is satisfied.

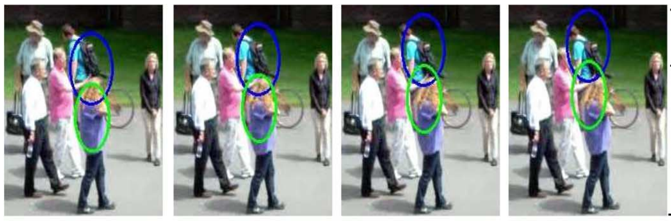
We use the mean shift vector [2] with the opposite direction as our $\mathbf{J}_{\mathbf{x}}$ in the implementation, while maintaining the characteristic (ii) (Appendix C). If there is no constraint (i.e., make both $\mathbf{C}(\mathbf{x})$ and $\mathbf{C}_{\mathbf{x}}$ as $\mathbf{0}$), then the movement in (5) becomes

$$\delta\mathbf{x} = -\alpha\mathbf{J}_{\mathbf{x}}, \tag{6}$$

which is just the formulation of each kernel doing independent mean shift update.

Algorithm 1. Projected gradient iteration in each frame for multiple-kernel tracking. |.| takes entry-wise absolute value.

```
1. $\mathbf{x}$ is the final state from the previous frame; $counter = 0$.
2. **while** $counter < T$
3.   $counter = counter + 1$.
4.   Evaluate $\mathbf{C}(\mathbf{x})$ and$J(\mathbf{x})$.
5.   **if** $|\mathbf{C}(\mathbf{x}| < \boldsymbol{\varepsilon}_{\mathbf{C}}$ and$J(\mathbf{x}) < \varepsilon_J$ **then**
6.     End of the iteration.
7.   **else**
8.     if $|\mathbf{C}(\mathbf{x})| < \boldsymbol{\varepsilon}_{\mathbf{C}}$ and$J(\mathbf{x}) \geq \varepsilon_J$ **then**
9.       Compute $\delta\mathbf{x}_A$.
10.       Apply $\mathbf{x} = \mathbf{x} + \delta\mathbf{x}_A$.
11.     **else**
12.       **if** $|\mathbf{C}(\mathbf{x})| \geq \boldsymbol{\varepsilon}_{\mathbf{C}}$ and $J(\mathbf{x}) < \varepsilon_J$**then**
13.         Compute $\delta\mathbf{x}_B$.
14.         Apply $\mathbf{x} = \mathbf{x} + \delta\mathbf{x}_B$.
15.       **else**
16.         Compute $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$.
17.         Apply $\mathbf{x} = \mathbf{x} + \delta\mathbf{x}_A + \delta\mathbf{x}_B$.
18.       **end if**
19.     **end if**
20.   **end if**
21. **end while**
```



Intermediate Steps	*Iteration 1*	*Iteration 2*	*Iteration 3*	*Iteration 4*
cost function	3.44892	2.25533	1.48434	1.329651
1st constraint function value (Eq. 12)	1	1	1	1
2nd constraint function value (Eq. 13)	0	0	0	0
update vector applied	$\delta\mathbf{x}_A$	$\delta\mathbf{x}_A$	$\delta\mathbf{x}_A$	*NULL*

(e)

Fig. 2. An example of the intermediate steps. The two ellipses represent two kernels. (a) Before the iteration 1 starts in current frame. The initial locations of the two kernels are inherited from the final state in previous frame. (b)(c)(d) Updated results after the iteration 1 to 3. Since the cost function is above the threshold ($\varepsilon_J = 1.4$), $\delta\mathbf{x}_A$ is applied until the cost function is below the threshold. (e) The corresponding values for cost function and the constraint functions.

D. Adaptive Cost Function

When there is occlusion, not all the kernels are reliable. Thus, we associate each kernel with an adaptively changeable weight value w_{i} in the calculation of the total cost function,

$$J(\mathbf{x}) = \sum_{i=1}^{N} w_i J_i(\mathbf{x}). \tag{7}$$

Therefore, the movement vector in (5) is modified to be

$$\delta\mathbf{x} = \alpha\left(-\mathbf{I} + \mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^T\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}_{\mathbf{x}}^T\right)\mathbf{W}\mathbf{J}_{\mathbf{x}} + \left(-\mathbf{C}_{\mathbf{x}}\left(\mathbf{C}_{\mathbf{x}}^T\mathbf{C}_{\mathbf{x}}\right)^{-1}\mathbf{C}(\mathbf{x})\right), \tag{8}$$

where $\mathbf{W} = \begin{bmatrix} W_1\mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & W_N\mathbf{I} \end{bmatrix}$ and $W_i \propto simi_i(\mathbf{x})$; $\mathbf{I}$ is an $n/N \times n/N$ identity matrix with n being equal to the dimension of the state space, and N is the number of kernels. The value w_i, which corresponds to the $i-th$ kernel, is adaptively updated based on the individual similarity $simi_i$ and is normalized to make the sum equal to N. The similarity represents the degree of match between the color feature of the candidate and the target model in the single kernel [2]. The higher similarity will give us the higher weight value, which corresponds to more confidence on this kernel. This would enable the system to produce a better movement toward the optimum when the kernels have conflicts.

E. Scale Change Issue

The scale (size) of the video object will probably change if the object is moving toward or away from the camera. Unlike the other scale update methods [2], [4], [6], which need additional processes for dealing with scale variable, we propose a simple while effective solution to overcome this issue, as evidenced by the experimental results in Section IV.

The extracted histogram used for similarity measure of the object is highly dependent on the kernel bandwidth h. If the object size is changing, it is better to adaptively change the kernel bandwidth in order to obtain the histogram that matches the original one. If all the corresponding pixels have the similar weight, the histograms would be similar. For instance, Fig. 3 shows the same object with different sizes. The two corresponding pixels (red dots) have the same weights for the histogram calculation if $d/h = d'/h'$. Hence, the object size and the kernel bandwidth are positively correlated. Thus, the change of the kernel bandwidth h is utilized to infer the

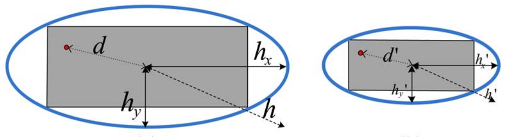


Fig. 3. (a) Original size. (b) Smaller size. The grey rectangle is the object. The blue ellipse represents the kernel with bandwidth h and h' which is equal to $\sqrt{h_x^2 + h_y^2}$ and $\sqrt{h_x'^2 + h_y'^2}$, respectively. The red dots, whose distances to the center are d and d', are the corresponding locations on the objects with two different sizes.

change of the object scale. We take the derivative of the density estimator in (1) with respect to h,

$$\nabla f(h) = \frac{\partial f(h)}{\partial(h)} = f(h)\left[\frac{-2\sum_i (g(v_i)v_i)}{h\sum_i k(v_i)} + \frac{2\sum_i (\omega_{\mathrm{i}} g(v_i)v_i)}{h\sum_i \omega_{\mathrm{i}} k(v_i)}\right], \tag{9}$$

where $g(v_i) = -k'(v_i)$ and $v_i = \|\frac{\mathrm{x}-\mathrm{z_i}}{h}\|^2$.

We can see that all the terms in (9), including ω_{i}, v_i, $g(v_i)$, $k(v_i)$, and h, can be directly inherited from the tracking steps such as the mean shift vector and cost function computing. Thus, based on these by-products, the scale change factor can be efficiently estimated as,

$$\Delta s = \beta\frac{\nabla f(h)}{hf(h)} = \beta r(h), \tag{10}$$

where β is the scalar factor which takes into account both the sensitivity of the scale change and the smoothing factor. Δs is proportional to the gradient $\nabla f(h)$ normalized by the current kernel bandwidth h and density estimation value $f(h)$ and enables the system to control the step size adaptively. The scale s_t at time t can be updated according to (11),

$$s_t = s_{t-1}(1 + \Delta s). \tag{11}$$

If the average of all kernels' similarity values is above a certain threshold, the scale update takes place after the iteration in Algorithm 1 finishes. The scale change method is developed based on the assumption that the localization is done, i.e., the center of the kernel in the current frame and the center of the kernel in the previous frame are at the same point or two close points on the target, therefore the scale update should take place after the tracking iteration is completed. In this manner, the scale can be adaptively updated to reflect the appropriate size.

IV. Experimental Results

In this section, we will start with introducing our implementation details. Next, the experiments are divided into three parts: first, the simulation scenarios are mainly in tracking a particular person in the crowd. We compared our proposed multiple-kernel tracking with two kernel-based tracking methods: multiple collaborative kernels [11] and single kernel mean shift [2], and two state-of-the-art trackers: on-line boosting [15] and superpixel tracking [16], by using several databases. Second, we will show the effectiveness of our approach for dealing with the scale changes of the tracked objects. Finally, robust tracking of multiple people simultaneously under occlusion is demonstrated.

A. Implementation Setup

In the multiple-kernel tracking module, each object is represented by multiple inter-related kernels, and the overall state vector $\mathbf{x}$ is composed of the centroids of all the kernels $[x_1 y_1 \ldots x_N y_N]^T$, where N is the number of kernels, and (x_i, y_i) is the centroid of the $i-th$ kernel. Theoretically, more kernels will give better results since it can handle occlusion from various directions, especially in a crowded scene environment. However, due to the different sizes of the targets, the number of kernels should be limited. For example, if the size of the target is small, each kernel may cover only a small portion of the body. The extracted histograms may not be informative enough due to insufficient pixels. In this paper, based on the symmetry and characteristics of human body, we use two kernels or four kernels to represent an object, and the layouts are shown in Fig. 4. These kernels are generated systematically and automatically once the multiple-kernel tracking is activated. The layout and the constraints can be designed to reflect the target's physical characteristics. For instance, in the two-kernel setting, no matter the viewing angle is from front or side, the system automatically generates the first kernel to represent the upper 55% while the other one covers the lower 55% of the object, so there is some overlapping between them. Since the upper and lower body parts of human normally have rigid geometrical relationship, it is reasonable to formulate the constraints as (12) and (13):

$$c_1(x_i, x_j) = \frac{\left((x_i - x_j)^2 - L_{x,ij}^2\right)}{\left(L_{x,ij}^2 + 1\right)}, \tag{12}$$

$$c_2(y_i, y_j) = \frac{\left((y_i - y_j)^2 - L_{y,ij}^2\right)}{\left(L_{y,ij}^2 + 1\right)}, \tag{13}$$

where $L_{x,ij}$ and $L_{y,ij}$ are the distances which remain constant after the initialization, unless the object size changes. At the first frame, each kernel initializes its own target model based on its covering area, and the $L_{x,ij}$ and $L_{y,ij}$ values are automatically determined by setting (12) and (13) to zero; that is, the constraint functions are satisfied in the beginning: $c_1(x_i^{initial}, x_j^{initial}) = 0$ and $c_2(y_i^{initial}, y_j^{initial}) = 0$. The $L_{x,ij}$ and $L_{y,ij}$ values are adjusted in proportional to the scale change only when the size of the object changes. As shown in Fig. 4, the extremes of the

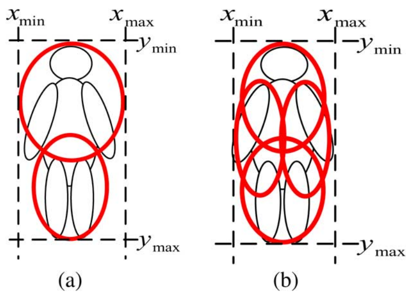


Fig. 4. Layout for the multiple kernels. (a) 2 kernels. (b) 4 kernels. Kernels are represented as red ellipses. Note that the object is not necessarily viewed from the front. The figure here is just for the demonstration purpose.

kernels, x_{min}, x_{max}, y_{min}, and y_{max}, form the bounding box. The centroid of the tracking result is obtained based on (14),

$$x_c = \left(\frac{x_{min} + x_{max}}{2}\right), \quad y_c = \left(\frac{y_{min} + y_{max}}{2}\right). \tag{14}$$

The α in (8) is set to 1 since the mean shift vector already provides adaptive step size. In Algorithm 1, the values of $\boldsymbol{\varepsilon_C}$ actually depends on the size of the object, and we assign the values in the range of 0.05 to 9 to each entry of it; ε_J is given by the value of $0.7 \times N$, where N is the number of the kernels. The max number of iteration (T) is equal to 5. β in (10) is set as 9×10^3 empirically to reflect appropriate amount of the scale change. The target's model and the scale will only be updated if the average of all kernels' similarity values is above 0.3. We use K-L distance [18] for all the similarity-related and cost computations through the implementation. To construct the histogram of the object, the HSV color space and roof kernel are employed.

B. Experiments on Multiple-Kernel Tracking

First of all, in order to emphasize the robustness of our proposed multiple-kernel tracking method, we focus on tracking a specific person who is occluded, with potential scale change, in the crowd within the video. The comparisons with two other kernel-based methods, as presented in [2] and [11], and two state-of-the-art trackers, on-line boosting [15] and superpixel tracking [16], show the robustness and improvement of our proposed method. In this experiment, all the targets of interest and kernels are selected manually in the beginning.

Fig. 5 shows some of our test results. We have evaluated our approach using CAVIAR Database [19], with the frame size being 384×288. We select some clips that have at least 5 people in the scene and pick the target that is continuously occluded during the movement. The target is marked by the bounding box for readers to see clearly. Fig. 5(a) shows 5 representative frames out of the tracking results based on using our proposed method with two kernels and the competing methods, and the corresponding trackers are shown in different colors. In frames #28 and #132, the target is occluded. Our method can effectively track the target (marked as red bounding box), while it results in larger errors by applying the methods in [11], [15] or even losing the tracking of the target by using the methods in [2], [16].

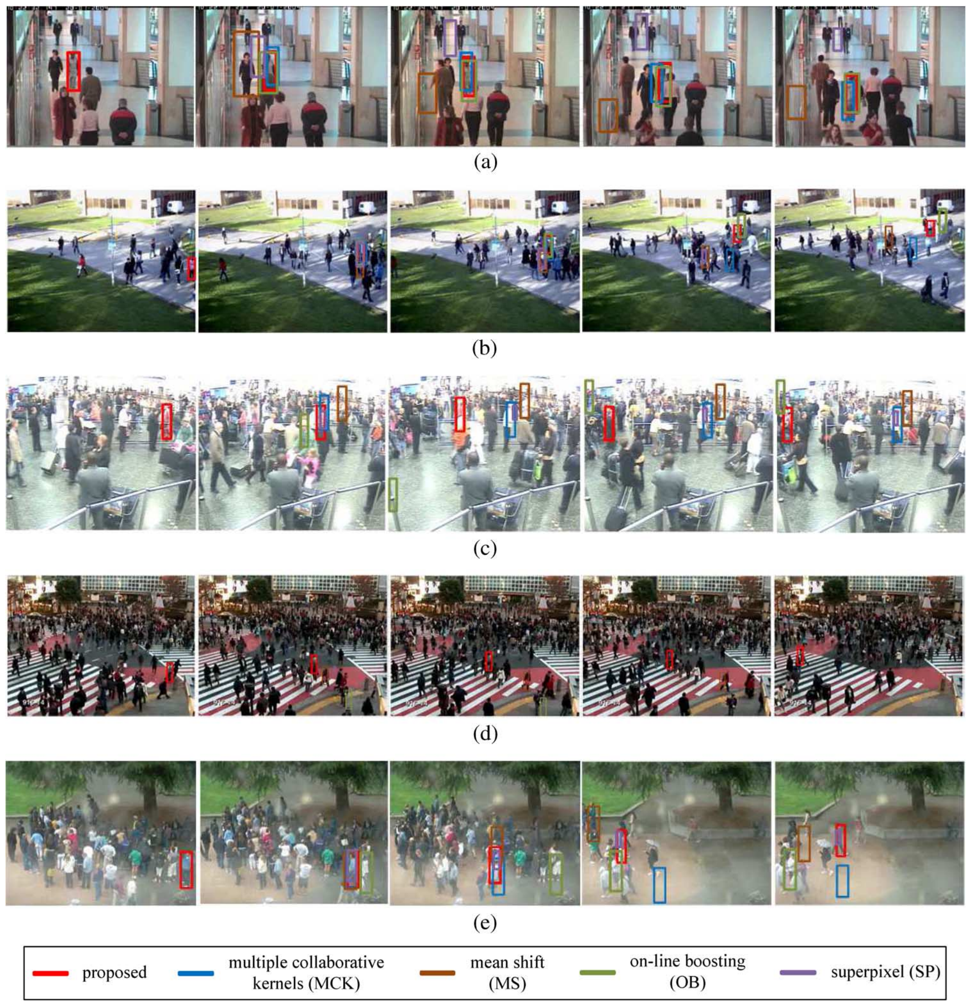


Fig. 5. Track a person under occlusion. (a) CAVIAR Database: *OneStopMoveEnter1cor* sequence. Frame #1, #28, #73, #132, #164. (b) *PETS2010* Database: *S2.L2.View_001* sequence. Frame #344, #368, #381, #406, #429. (c) *i-LIDS* Database: *MCTTE0102* sequence. Frame #1, #71, #189, #322, #361. (d) Crowd Analysis Dataset. Frame #1, #61, #82, #93, #181. (e) Our own recorded video. Frame #1, #39, # 102, #444, #485.

In addition to CAVIAR database, we use the video from PETS 2010 Benchmark Database [20]. One of the objectives of creating this database is to track the individuals within the crowd, so it is perfect for our usage to prove the effectiveness of our approach. The frame size is 768 $\times$ 576, and it is taken under 7 frames per second. Fig. 5(b) shows that our method with two kernels produces the promising tracking and scale updating results. Moreover, the iLIDS database [21] is utilized to evaluate the tracking accuracy. It captures the scene in a busy airport. The frame size is 720x576. Our algorithm using four kernels clearly outperforms the others (Fig. 5(c)). Some video clips for the crowd analysis [22], with the frame size of 480 $\times$ 360, are also used for the experiments in this paper. In Fig. 5(d), our tracker with two kernels tracks the target moving across the central area. The compared trackers lose the target, and some of them even drift out of the visible area.

We also use our own captured videos to do the experiments. These videos, with the frame size of 640 $\times$ 480, contain the tracked target changing the direction while walking in the crowd scene. In Fig. 5(e), frames #39, #102, and #444 show the target is continuously occluded in the crowd. The proposed method using two kernels steadily tracks the specific person effectively.

To further evaluate the performance quantitatively, the absolute errors for each frame in the above scenarios are computed and shown in Fig. 6. The error is defined as the pixel distance between the target's centroid of the simulation result and the ground truth as provided by the CAVIAR database [19] for the scenario in Fig. 5(a). For PETS database, iLIDS database, crowd analysis dataset, and our own captured video, the ground truth is produced by using the Video Performance Evaluation Resource (ViPER) tool [23]. The deviations remain in the lowest values by using our method among all the scenarios, while the other trackers lose the targets during the occlusion, resulting in large deviations in some or all of the cases. Note that sometimes small error does not necessarily mean an accurate tracking. For instance, in Fig. 5(c), the on-line boosting tracker drifts away after the occlusion in the beginning and is trapped at the upper left corner. The target happens to move from right to left, resulting in the decrease of the error of the on-line boosting tracker (Fig. 6(c)), but in fact the tracker still loses the target. In Fig. 6(d), three compared trackers drift away and are out of the visible area, so their corresponding error curves terminate in the early stage. Table I further shows the average error among all the frames of the video clips. The quantitative error measure-

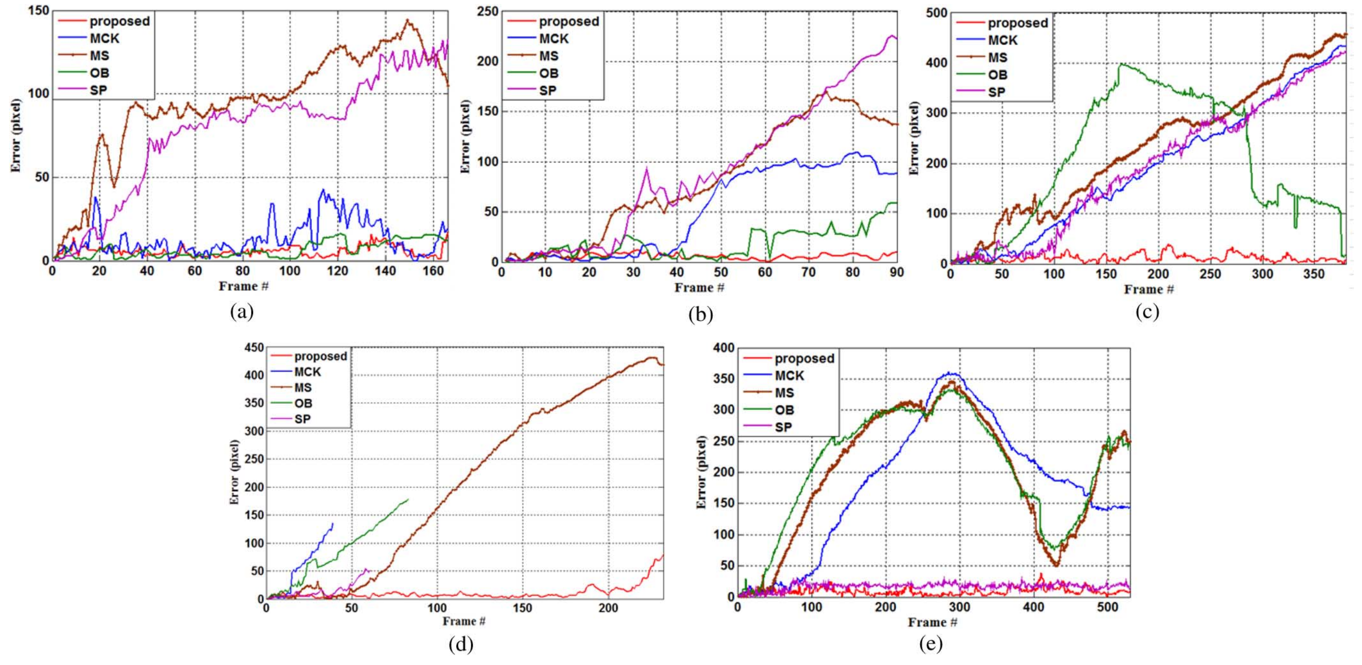


Fig. 6. Errors (pixels) of the object center against the ground truth in Fig. 5(a)–(e) are the error plots for Fig. 5(a)–(e), respectively. Our proposed method (red) outperforms other methods.

TABLE I
AVERAGE ERROR

Method	Avg. Error (pixels)				
	CAVIAR	*PETS2010*	*i-LIDS*	*Crowd Analysis*	*Our own video*
proposed method	**5.62**	**5.73**	**11.43**	**10.81**	**8.75**
multiple collaborative kernels (MCK) [11]	13.27	50.51	192.19	55.60	177.21
mean shift (MS) [2]	94.95	81.40	231.59	205.55	199.18
on-line boosting (OB) [15]	6.53	17.86	200.74	84.17	208.34
superpixel (SP) [16]	77.62	90.55	192.13	15.02	16.70

ment in the table demonstrates the superior performance of our proposed method in dealing with occlusion.

C. Experiments on Scale Change

In these experiments, we focus on tracking the targets that have obvious scale changes in the videos from different databases [19]–[21]. Similarly, all the targets are selected manually in the beginning and are tracked by our multiple- kernel tracking with the scale update. Fig. 7 exhibits the robustness of our proposed method. Since some of the other methods under comparison did not explicitly consider the scale change in their systems, we do not provide the comparison in the paper. By efficiently using the by-product from the tracking procedure, i.e., the required terms for scale change factor computation in (9) can be inherited from the tracking steps in the associated optimization iterations, we not only perform the successful tracking but also effectively updates the scale of targets accordingly.

D. Experiments on the Automatic Tracking System

In order to achieve the fully automatic tracking, we further integrate the multiple-kernel tracking scheme into a Kalman filter based tracking system [24], where we use the Kalman prediction as the initial location for the multiple-kernel tracking module and treat the result from the multiple-kernel tracking as the measurement for Kalman update. Our goal is to develop

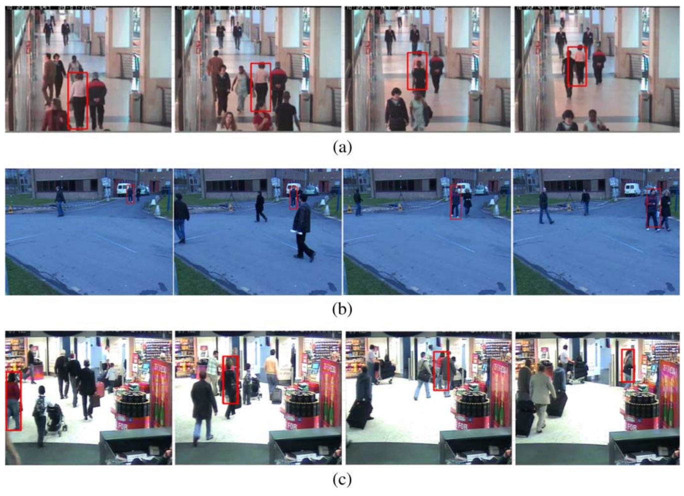


Fig. 7. Tracking a person under occlusion with obvious scale change by using proposed method. (a) *CAVIAR* Database: *OneStopMoveEnter1cor* sequence. (b) *PETS2010* Database: *S2.L1.View_006* sequence. (c) *i-LIDS* Database: *MCTTE0201* sequence.

a fully automatic tracking system that can track multiple objects simultaneously. Several video clips are used for the performance evaluation. In order to make people differentiable, we tag them with numbers and different colors bounding boxes as

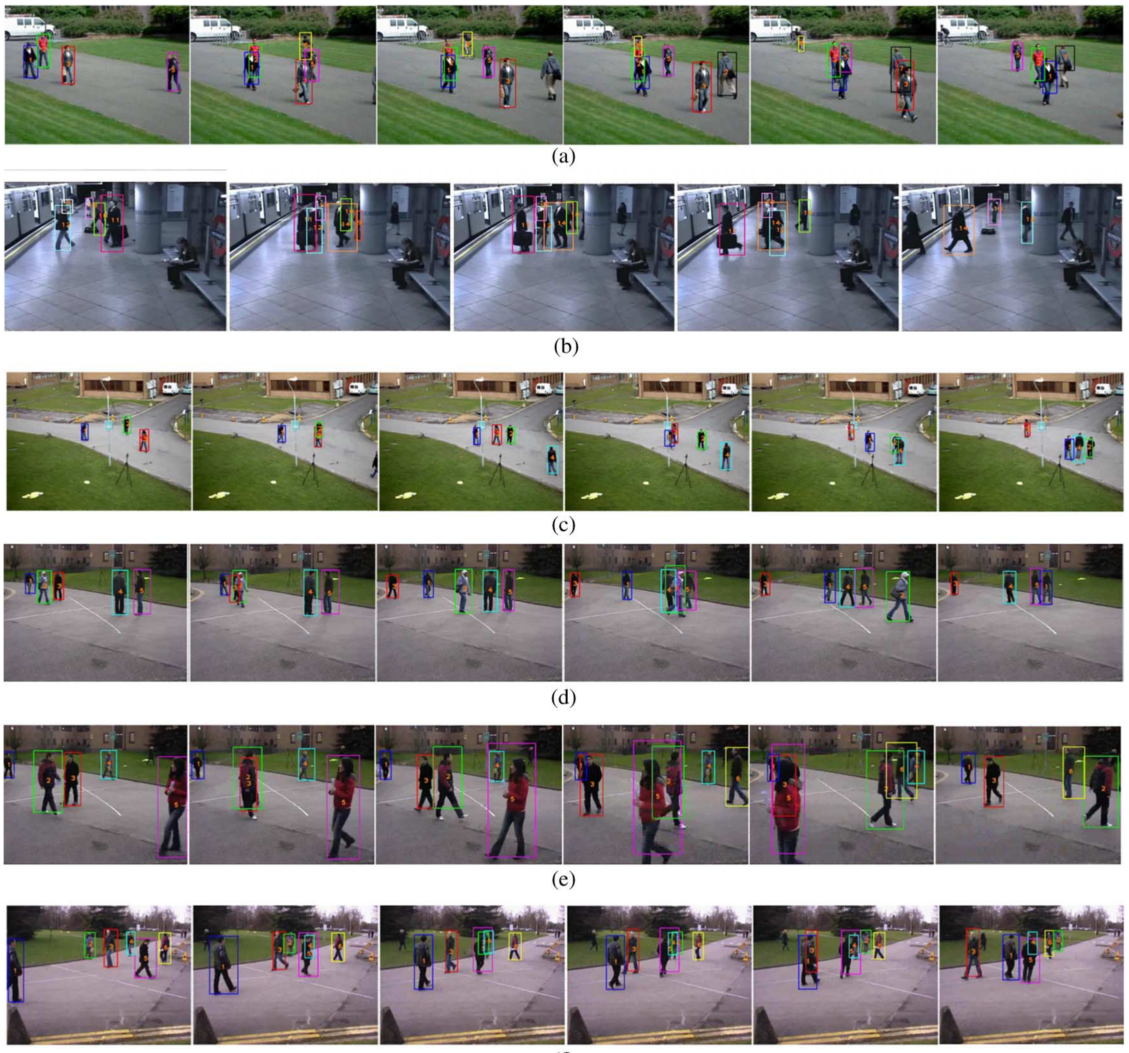
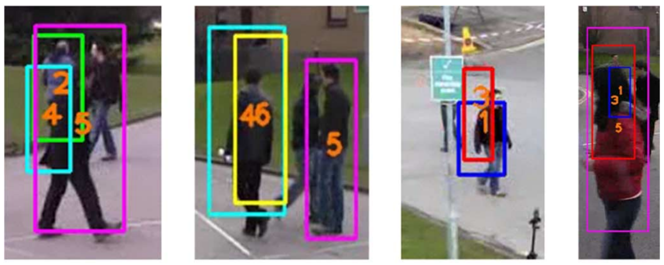

(a)
(b)
(c)
(d)
(e)
(f)

Fig. 8. Tracking all individuals in the video simultaneously. Different people have been labeled with numbers and also with different colors bounding boxes. (a) Our own captured video. (b) *AVSS_AB_EVAL*. (c) *PETS 2010_S2.L1.View_001*. (d) *PETS 2010_S2.L1.View_008*. (e) *PETS 2010_S2.L1.View_008*. (f) *PETS 2010_S2.L1.View_007*.

well. Fig. 8 shows some of the results. In addition to our own captured videos, we use the video from AVSS 2007 Database [25] and PETS 2010 Benchmark Database [20]. From the background subtraction to the end of tracking, everything is fully automatic, and there is no need for any manual intervention. Note that the initialization of the multiple kernels is automatic as stated in Section IV-A.

We further compare the performance, in terms of the average error, with two base-line trackers, mean shift tracking [2] and particle filtering [26], [27], by using the same test video clips. The average error is obtained based on all the people in all the frames, and the error is defined as the pixel distance between the targets' centroids of the simulation results and the ground truths which are produced by using ViPER tool [23]. Originally, the mean shift method and particle filtering lose the targets occasionally during occlusion, and thus the trackers tend to drift away easily, resulting in larger errors. Some of their failure cases are shown in Fig. 9. In order to show the robustness and stability of our method, here we adopt another way to compute the average errors. For each person, we only consider



(a) (b) (c) (d)

Fig. 9. Some tracking failure examples from mean shift or particle filter. In (c), the bounding box 3 lost the person at left who has been occluded by the sign and the person at right. Since the box does not cover the area of the person at left, we remove this error value from the calculation of average error in Fig. 10.

the frames in which the corresponding bounding boxes of all the three trackers overlap with that person in the error computation. For instance, the frame in Fig. 9(c) is excluded from the average error computation for the person labeled as 3 since the bounding box 3 totally leaves the corresponding person. Fig. 10

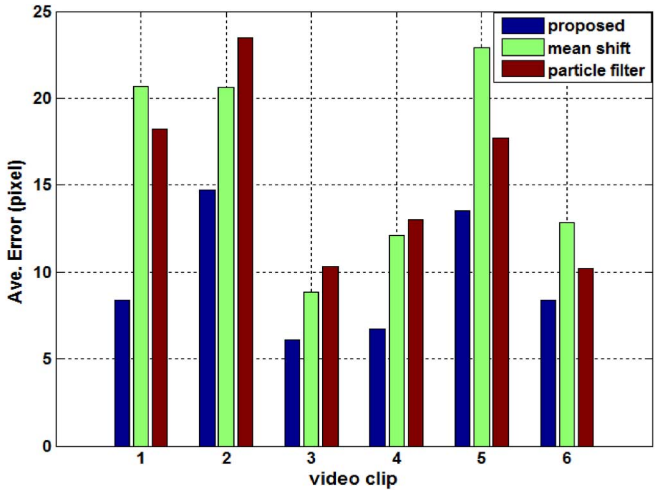


Fig. 10. Comparison between our method (blue), single kernel mean shift (green), and particle filter (brown). The video clips 1–6 correspond to the results in Fig. 8(a)–(f), respectively.

shows the quantitative comparison based on the above rule. We can see that even after discarding those frames having huge error values, our proposed method still attains better performance and higher stability since the occlusion results in larger deviation when using the other two methods even if they do not lose the target completely.

V. Discussion

A. Projected Gradient

The decomposition of the update vector $\delta\mathbf{x}$ ensures the state vector reaching the optimum efficiently according to the characteristics proved in Appendix B. Although some linear approximation has been exploited, the promising simulation results demonstrate that the characteristics still hold in practice. Fig. 11 shows a typical case that how the values of the cost function and one of the constraint functions change. Since the cost function exceeds the threshold ε_j in Algorithm 1 and the constraint function is satisfied, the update takes place by moving along $\delta\mathbf{x}_A$ where one can see that the cost function drops to the lower level while the constraint function remains the same as the iteration goes. In this example, the process stops at the 4th iteration since both the cost function and constraint function are below the thresholds.

The projected gradient in (8) plays an important role in both the accuracy and computation efficiency. The tracking accuracy comparison between the methods with and without projected gradient term is shown in Table II. In the simulation without projected gradient, we discard the term $\mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T$ in (8) while computing $\delta\mathbf{x}_A$, and other elements are left unchanged. Since the projected term is removed, the characteristic (ii) does not exist, and it is likely that the values of constraint functions change while moving along the $\delta\mathbf{x}_A$. After that, applying $\delta\mathbf{x}_B$ may increase the cost function. Thus, the oscillation tends to happen and makes the tracking accuracy worse. The snapshots of the kernels' movements and the function values are shown in Fig. 12, which is a typical example of how the projected

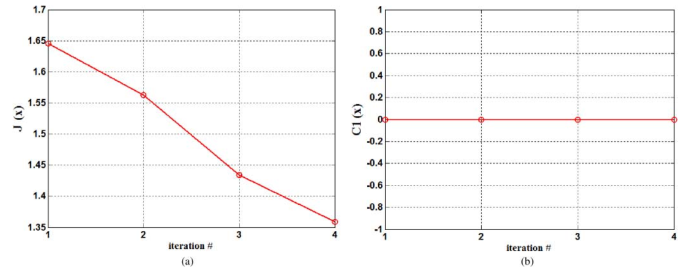


Fig. 11. An example that shows the values of (a) cost function and (b) constrain function change as the iteration goes. Note that only one constraint function value is shown here. In this example, we can see it takes 4 iterations to make $J(\mathbf{x}) < \varepsilon_J(\varepsilon_J = 1.4)$.

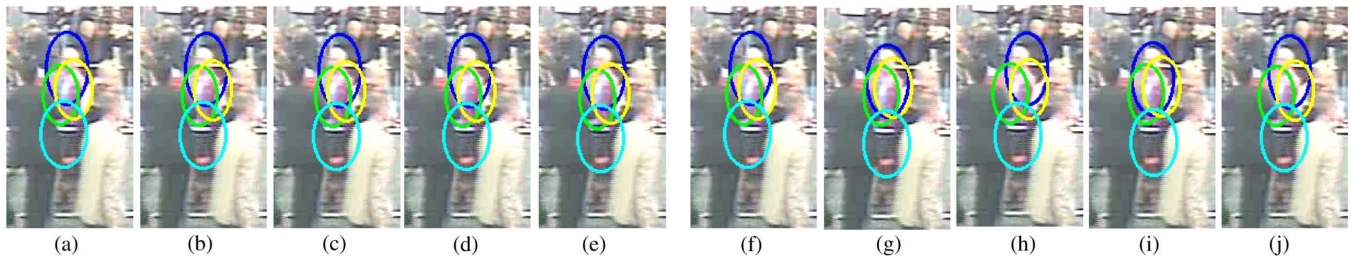
gradient leads to the convergence through the iterations while the one without projected gradient does not approach the steady point before the predefined maximum number of the iterations is reached $(T = 5)$. One can see clearly the oscillation of either the kernels' movements or the function values when the projected gradient is not utilized. Projected gradient enables the decrease of the cost function while maintaining the satisfaction of the constraints, so in practice our proposed method usually does not need the computation of $\delta\mathbf{x}_B$.

In order to decrease the probability of oscillation which occurs while not using projected gradient, one possible way is to reduce the step size (α) when applying the gradient vector. However, it takes more iterations for tracking, and it still fails in some cases. One can compare the computations of two methods, i.e., with and without projected gradient, by looking into (8). In each iteration, the method with projected gradient needs additional computation of $\mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}\mathbf{C_x}^T$, and the one without projected gradient usually needs to compute $\delta\mathbf{x}_B$, which includes $\mathbf{C_x}(\mathbf{C_x}^T\mathbf{C_x})^{-1}$. Since the matrix inversion is normally the dominant computation and both of them have it in the computation, the computation cost in each iteration is similar in both cases. However, the required number of the iteration is higher for the one without projected gradient in order to get the similar tracking performance. For instance, Table III shows the efficiency comparison between these two schemes using the video clips in PETS database. We tried to adjust the parameters in order to obtain the similar tracking accuracy for both cases, and the numbers of maximum allowed iteration (T) in Algorithm 1 are set as 3 and 6 for with and without projected gradient, respectively. Hence, the one with projected gradient attains higher computation efficiency in terms of processed frames per second.

Compared to the Newton's method, gradient based methods usually need more iterations to reach the optimum point. However, it has been proved in [28] that "*the mean shift procedure with a piecewise constant profile g is equivalent to the Newton's method applied to a density estimate using the shadow of g*." If we choose a piecewise linear profile k, such as the roof kernel, as we did in the implementation, the mean shift procedure will have a piecewise constant profile $\mathrm{g}(\mathrm{g}(\cdot) = -\mathrm{k}'(\cdot))$; that is, doing mean shift update can have the same iterations just as Newton's method (i.e., the mean shift vector equals to the Newton step). Therefore, by combining the mean shift vector into our gradient-based approach (8) enables our method to attain both robust performance and efficiency.

TABLE II
AVERAGE ERROR

Method	Avg. Error (pixels)				
	CAVIAR	*PETS2010*	*i-LIDS*	*Crowd Analysis*	*Our own video*
proposed method	**5.62**	**5.73**	**11.43**	**10.81**	**8.75**
without projected gradient	6.88	10.85	87.52	90.08	43.86



Intermediate Steps	*Iter. 1*	*Iter. 2*	*Iter. 3*	*Iter. 4*	*Iter. 5*
cost function	9.52	7.61	7.42	7.37	7.37
1st constraint function	0	0	0	0	0
3rd constraint function	0	0	0	0	0

Intermediate Steps	*Iter. 1*	*Iter. 2*	*Iter. 3*	*Iter. 4*	*Iter. 5*
cost function	9.52	7.86	9.02	7.80	9.16
1st constraint function	0	-0.681	0.178	-0.637	0.04
3rd constraint function	0	-0.883	-0.17	-0.77	-0.22

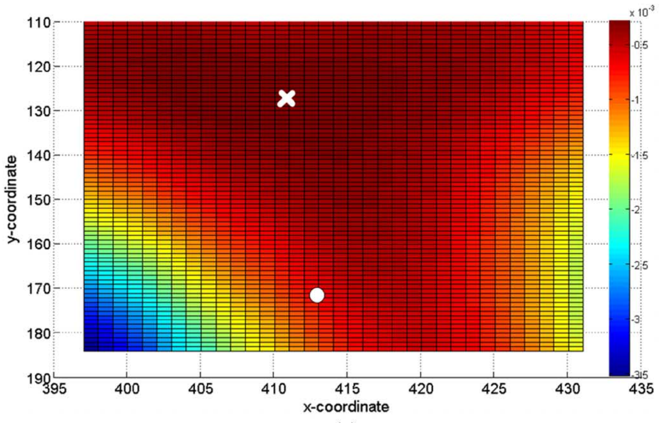
Fig. 12. Two examples of the intermediate steps with and without the projected gradient. The four ellipses represent four kernels. Left and right columns show the snapshots and the function values of the methods with and without the projected gradient, respectively. The one with projected gradient gradually move to the target while the one w/o projected gradient swings from different states and does not converge. Note that there are 12 constraints in the four-kernel setting, and we only show two constraint functions here for illustration purposes.

TABLE III
EFFICIENCY COMPARISON

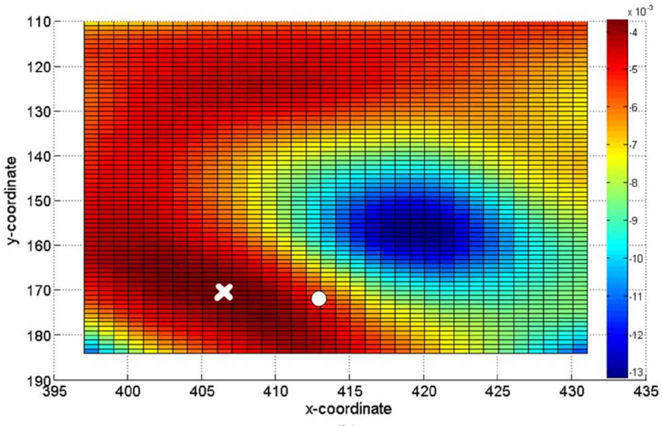
Computation	*w/ projected gradient*	*w/o projected gradient*
ave. error (pixel)	7.3575	7.4008
T in algorithm 1	3	6
frames per second	10.19	5.869

B. Similarity Map

The proposed method considers multiple kernels with constraints to make the solution feasible during the optimization procedure. Take the same frame in Fig. 12 as an example, Fig. 13(a) is the similarity map constructed under single kernel based on the negative value of the K-L divergence around the neighborhood area of the target. One can see that there is a large flat region with the similar values (dark red) which makes it hard to locate the local maximum of the map. The local optimum, marked as a white cross, is far away from the ground truth, marked as a white dot, which causes the tracking error. In Fig. 13(b), the same map is built based on the multiple kernels with constraints (7). The local optimum locates in the smaller region at the lower left corner, and the variation of the map values are more obvious which makes it suitable for gradient-based optimization. During the optimization iteration, it prevents the divergence of the tracking. It is not surprising to see that the estimated optimum (white cross) does not perfectly coincide with the ground truth. In practice, due to the appearance variation of the target, presence of the noise, and especially in the video with a lot of occlusion, we do not expect the experimental results can match the ground truth perfectly. However, it shows they are close enough to perform the successful tracking. In short, although our proposed method utilizes simple elements, e.g., HSV histogram and K-L divergence, the multiple constrained kernels effectively build a similarity map that enables us to do the successful tracking.



(a)



(b)

Fig. 13. Similarity map of the neighborhood area of Fig. 12. White dot is the ground truth of the target location. White cross is the local optimum of the map. (a) Compute the map based on single kernel. (b) Compute the map based on the multiple kernels with constraints.

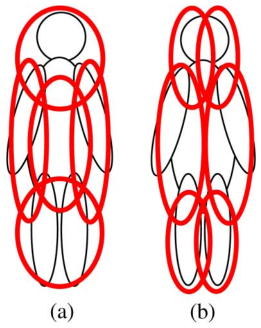


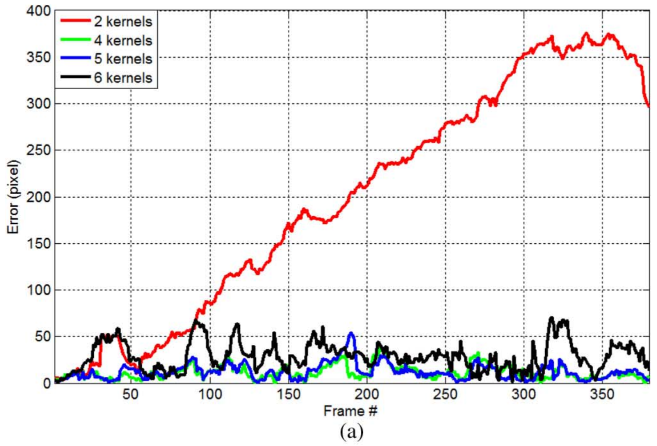
Fig. 14. Layout for the multiple kernels. (a) 5 kernels. (b) 6 kernels.

C. Kernel Design

When we design the configuration of the kernels, the overlapping between a pair of the kernels is allowed. However, the overlapping area between any pair of the kernels should be limited; otherwise, if the occlusion happens, it would be easily for both kernels to suffer from the ill-observable condition. For human tracking, it is intuitive to design two kernels, one for the upper part and one for the lower part, since (i) It is reasonable to assume that the human remains upright when they are moving, so the geometrical relationship of upper and lower body parts rarely changes. (ii) We have observed that for many cases of the occlusion between humans, the occlusion happens from the bottom of the human body. When the lower part is occluded, the upper part may still be well-observed which helps track the target correctly. For more complex scenarios (e.g. Fig. 5(c)) that the occlusion happens from different directions, four kernels are adopted for handling the occlusion from various directions. Theoretically, with more kernels representing the target, more information is available to enable the better tracking, especially in the crowded scene environment. We compare the results from using different numbers of kernels. The layouts of the five and six kernels are shown in Fig. 14. Fig. 15(a) shows the tracking errors of $i-LIDS$ video (Fig. 5(c)) under different numbers of kernels. In this scenario, the scene is more cluttered and the target is usually occluded from the various directions. The one with two kernels loses the target while the ones with four, five and six kernels successfully locate the target. However, due to the different sizes of the targets, the number of kernels should be bounded. If we assign more kernels to the target, each kernel will cover only a small portion of the target (assume there are only limited overlapping areas between kernels). The extracted histograms may not be informative enough due to insufficient pixels, especially when the target is small, like the scenario in Fig. 5(d). Fig. 15(b) shows the tracking errors of *Crowd Analysis*video (Fig. 5(d)). One can see the error of four-kernel is slightly better than the error of two-kernel, but when we add more kernels, the performance is worse due to the information deficiency for some kernels. In this case, instead of using statistical-based feature, other features may be included with more kernels setting.

D. Limitation

Our method can be applied to solve the occlusion problem, but it may not perform well if the target is totally occluded for a long time. For example, Fig. 16(a), (b) shows a person walking into a crowd and can be barely seen for a long time. Our tracking scheme stays around the same position in the beginning but starts to drift away if the target keeps unobservable. When it reappears afterward, the tracker cannot be recovered. To deal with this long-term totally occlusion, the classifier/detector-based approach may be adopted. Once the target reappears, it can search all over the images and detect the target.



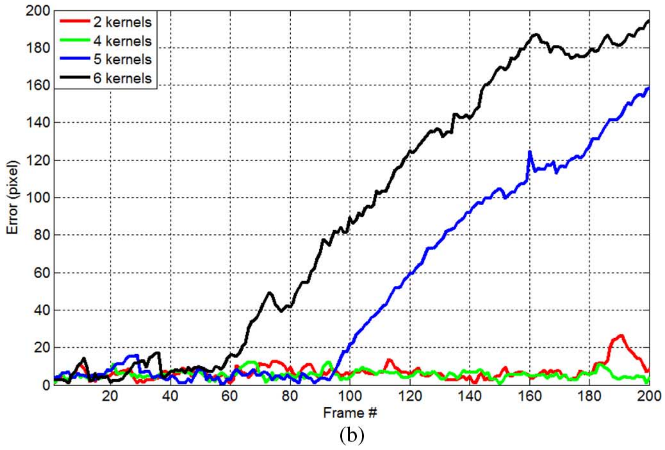


Fig. 15. Errors (pixels) of the object center against the ground truth. Results of using different number of kernels are shown in different colors. Red: two kernels; Green: four kernels; Blue: five kernels; Black: six kernels. (a) *i-LIDS* Database. (b) Crowd Analysis Dataset.

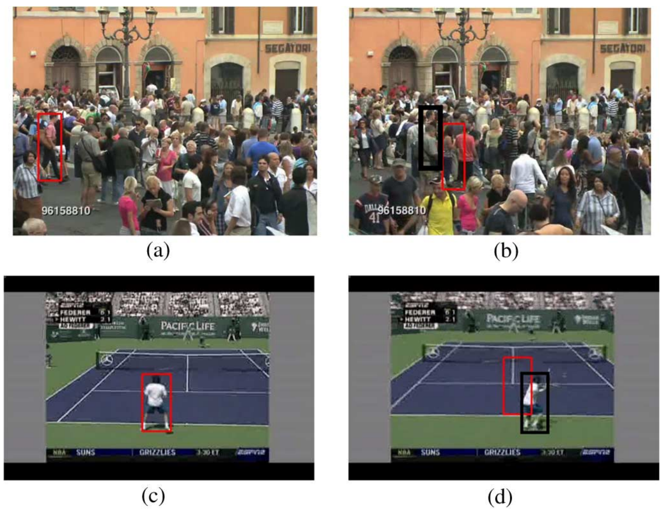


Fig. 16. Some failure cases. Red: proposed tracker; Black: ground truth. (a) The case for the long-term total occlusion. (b) The case for fast and abrupt motion.

Our method relies on the gradient based solution which is a local optimum searching scheme, so if the target is not in the neighborhood area between two consecutive frames, the tracker is expected to lose the target. Like Fig. 16(c), (d), if the target has fast and abrupt motion relative to the frame rate, the tracker is unlikely to follow the target. To overcome this issue, in addition to the detection method, a good dynamic model can be combined with our method, like particle filter and advanced MCMC sampling [32].

VI. Conclusion

We propose an innovative method that uses projected gradient to facilitate multiple-kernel tracking in finding the best match under predefined constraints. Since some of the kernels are not observable, the adaptive weights are employed to the kernels to lower the importance of the ones being occluded while enhance the ones which are well-observable. Moreover, a simple yet effective approach is presented to deal with the scale change issue. Finally, the multiple-kernel tracking is combined with Kalman filter to form a complete automatic tracking system. Based on the experimental results, the multiple-kernel tracking can successfully track the specific target under severe occlusion, and the overall system also demonstrates the promising results for tracking every individual simultaneously.

Appendix A

The projection matrix $\mathbf{P}$ which projects the vector onto the space in which the values of the constraint functions remain intact is expressed as $(\mathbf{I}-\mathbf{C}_\mathbf{x}(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x})^{-1}\mathbf{C}_\mathbf{x}^T)$ [14]. After applying $\mathbf{P}$ to $-\mathbf{J}_\mathbf{x}$, the gradient descent direction, the first term $\delta\mathbf{x}_A$ is $(-\mathbf{I}+\mathbf{C}_\mathbf{x}(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x})^{-1}\mathbf{C}_\mathbf{x}^T)\mathbf{J}_\mathbf{x}$. In addition to this, the constraint functions $\mathbf{C}(\mathbf{x})=\mathbf{0}$ need to be satisfied; hence, another moving vector $\delta\mathbf{x}_B$ enabling the constraints to hold is introduced; that is, to make the constraint functions approach zero when moving along $\delta\mathbf{x}_B$,

$$\mathbf{0}=\mathbf{C}(\mathbf{x}+\delta\mathbf{x}_B)\approx\mathbf{C}(\mathbf{x})+\mathbf{C}_\mathbf{x}^T\delta\mathbf{x}_B. \tag{A.1}$$

Thus, $\delta\mathbf{x}_B=-\mathbf{C}_\mathbf{x}(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x})^{-1}\mathbf{C}(\mathbf{x})$.

Appendix B

The characteristics of the projected gradient vector can be proved in the following:

i)

$$\begin{aligned}(\delta\mathbf{x}_A)^T\delta\mathbf{x}_B
&=\left(\alpha\left(-\mathbf{I}+\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}_\mathbf{x}^T\right)\mathbf{J}_\mathbf{x}\right)^T\left(-\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}(\mathbf{x})\right)\\
&=\alpha\left(-\mathbf{J}_\mathbf{x}^T+\mathbf{J}_\mathbf{x}^T+\mathbf{C}_\mathbf{x}\left(\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\right)^T\mathbf{C}_\mathbf{x}^T\right)\left(-\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}(\mathbf{x})\right)\\
&=\alpha\Bigg(\mathbf{J}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}(\mathbf{x})-\mathbf{J}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\left(\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\right)^T\\
&\qquad\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}(\mathbf{x})\Bigg)\\
&=\alpha\left(\mathbf{J}_\mathbf{x}^T+\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}(\mathbf{x})-\mathbf{J}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\left(\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\right)^T\mathbf{C}(\mathbf{x})\right)\\
&=0.\end{aligned}\tag{B.1}$$

Hence, $\delta\mathbf{x}_A$ and $\delta\mathbf{x}_B$ are orthogonal to each other.

ii) Let δJ_A and $\delta\mathbf{C}_A$ denote the changes of functions $J(\mathbf{x})$ and $\mathbf{C}(\mathbf{x})$ by moving along $\delta\mathbf{x}_A$, respectively. Assume the local linear approximation:

$$\begin{aligned}\delta J_A\approx\mathbf{J}_\mathbf{x}^T\delta\mathbf{x}_A&=\mathbf{J}_\mathbf{x}^T\alpha\left(-\mathbf{I}+\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}_\mathbf{x}^T\right)\mathbf{J}_\mathbf{x}\\
&=-\alpha\mathbf{J}_\mathbf{x}^T\left(\mathbf{I}-\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}_\mathbf{x}^T\right)\mathbf{J}_\mathbf{x}.\end{aligned}\tag{B.2}$$

Let matrix $\mathbf{M}=\begin{bmatrix}\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x} & \mathbf{C}_\mathbf{x}^T\\ \mathbf{C}_\mathbf{x} & \mathbf{I}\end{bmatrix}=\begin{bmatrix}\mathbf{C}_\mathbf{x}^T\\ \mathbf{I}^T\end{bmatrix}[\mathbf{C}_\mathbf{x}\ \mathbf{I}]=\mathbf{K}^T\mathbf{K}$, which is always positive semi-definite. Moreover, since $\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}$ is also positive semi-definite, the generalized Schur complement $(\mathbf{I}-\mathbf{C}_\mathbf{x}(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x})^{-1}\mathbf{C}_\mathbf{x}^T)$ is also positive semi-definite. Thus, $\mathbf{J}_\mathbf{x}^T(\mathbf{I}-\mathbf{C}_\mathbf{x}(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x})^{-1}\mathbf{C}_\mathbf{x}^T)\mathbf{J}_\mathbf{x}\geq 0$ and $\delta J_A\leq 0$. The cost function always decreases if moving along the direction $\delta\mathbf{x}_A$. Moreover,

$$\begin{aligned}\delta\mathbf{C}_A\approx\mathbf{C}_\mathbf{x}^T\delta\mathbf{x}_A&=\mathbf{C}_\mathbf{x}^T\alpha\left(-\mathbf{I}+\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}_\mathbf{x}^T\right)\mathbf{J}_\mathbf{x}\\
&=\alpha\left(-\mathbf{C}_\mathbf{x}^T+\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}_\mathbf{x}^T\right)\mathbf{J}_\mathbf{x}\\
&=\alpha\left(-\mathbf{C}_\mathbf{x}^T+\mathbf{C}_\mathbf{x}^T\right)\mathbf{J}_\mathbf{x}=\mathbf{0}.\end{aligned}\tag{B.3}$$

Thus, it will not change the values of the constraint functions by moving along the direction $\delta\mathbf{x}_A$.

iii) Let $\delta\mathbf{C}_B$ denote the changes of functions $\mathbf{C}(\mathbf{x})$ by moving along $\delta\mathbf{x}_B$. Assume the local linear approximation:

$$\delta\mathbf{C}_B\approx\mathbf{C}_\mathbf{x}^T\delta\mathbf{x}_B=\mathbf{C}_\mathbf{x}^T\left(-\mathbf{C}_\mathbf{x}\left(\mathbf{C}_\mathbf{x}^T\mathbf{C}_\mathbf{x}\right)^{-1}\mathbf{C}(\mathbf{x})\right)=-\mathbf{C}(\mathbf{x}).\tag{B.4}$$

Thus, no matter what the current values of constraint functions are, they will always go toward the value zero, i.e., make our problem constraints hold $\mathbf{C}(\mathbf{x})=\mathbf{0}$.

Appendix C

To minimize the cost function $J(\mathbf{x})$ is equivalent to maximizing $-J(\mathbf{x})$. Similar to the derivation in [2], we take the linear approximation of $-J(\mathbf{x})$ around $\mathbf{x}=\mathbf{x}_0$ and obtain the following equation after some manipulations,

$$\begin{aligned}-J(\mathbf{x})&=h(\mathbf{x}_0)+c_1\sum_i\omega_\mathrm{i}k\left(\left\|\frac{\mathbf{x}-\mathbf{z_i}}{h}\right\|^2\right)\\
&=h(\mathbf{x}_0)+f_K(\mathbf{x}),\end{aligned}\tag{C.1}$$

where $J(\mathbf{x})=\sum_{i=1}^m q_\mathrm{i}log(q_\mathrm{i}/p_\mathrm{i}(\mathbf{x}))$ is K-L distance between two histograms: $\mathbf{p}(\mathbf{x})=[p_1(\mathbf{x})\dots p_\mathrm{m}(\mathbf{x})]$ is $m-bin$ candidate model and $\mathbf{q}=[q_1\dots q_\mathrm{m}]$ is the target model. $h(\mathbf{x}_0)$ and c_1 can be seen as constants (independent to $\mathbf{x}$); $\omega_\mathrm{i}=\sum_{u=1}^m\frac{q_\mathrm{u}}{p_\mathrm{u}(\mathbf{x}_0)}\delta[b(\mathbf{z_i})-u]$ and $\mathbf{z_i}$ is the pixel location. Thus, we can get

$$\mathbf{J}_\mathbf{x}=\frac{\partial J}{\partial\mathbf{x}}=-\nabla f_K(\mathbf{x}).\tag{C.2}$$

According to [3], the mean shift vector corresponding to maximizing the function $f_K(\mathbf{x})$ can be expressed as:

$$\mathbf{m}(\mathbf{x})=c_2\times\frac{\nabla f_K(\mathbf{x})}{f_G(\mathbf{x})},\ c_2>0,\tag{C.3}$$

where $f_G(\mathbf{x}) = \sum_i \omega_i g(\|\frac{\mathbf{x}-\mathbf{z_i}}{h}\|^2)$ and $g(\cdot) = -k'(\cdot)$. Due to the positivity of ω_i and the monotonous decreasing of the kernel profile $k(\cdot)$ (i.e., $k'(\cdot)$ is negative), $f_G(\mathbf{x})$ is always positive. Hence, from (C.2) and (C.3) we can get

$$\mathbf{J_x} = c_3 \times (-\mathbf{m}(\mathbf{x}))\,,\ c_3 > 0; \tag{C.4}$$

that is, $\mathbf{J_x}$ and $(-\mathbf{m}(\mathbf{x}))$ are aligned. Therefore, for each kernel, it is reasonable to use the mean shift vector with the opposite direction as our $\mathbf{J_x}$.

In the characteristic (ii) in Appendix B, if we use $-\mathbf{m}(\mathbf{x})$ in calculating $\delta\mathbf{x}_A$,

$$\begin{aligned}\delta J_A &\approx \mathbf{J_x}^T\delta\mathbf{x}_A = \mathbf{J_x}^T\alpha\left(-\mathbf{I}+\mathbf{C_x}\left(\mathbf{C_x}^T\mathbf{C_x}\right)^{-1}\mathbf{C_x}^T\right)(-\mathbf{m}(\mathbf{x}))\\ &= -\frac{\alpha}{c_3}\mathbf{J_x}^T\left(\mathbf{I}-\mathbf{C_x}\left(\mathbf{C_x}^T\mathbf{C_x}\right)^{-1}\mathbf{C_x}^T\right)\mathbf{J_x} \le 0, \end{aligned}\tag{C.5}$$

which will not affect the non-positivity of δJ_A. Similarly, the values of the constraint functions will not change.

$$\begin{aligned}\delta\mathbf{C}_A &\approx \mathbf{C_x}^T\delta\mathbf{x}_A = \mathbf{C_x}^T\alpha\left(-\mathbf{I}+\mathbf{C_x}\left(\mathbf{C_x}^T\mathbf{C_x}\right)^{-1}\mathbf{C_x}^T\right)(-\mathbf{m}(\mathbf{x}))\\ &= \alpha\left(-\mathbf{C_x}^T+\mathbf{C_x}^T\right)(-\mathbf{m}(\mathbf{x})) = \mathbf{0}.\end{aligned}\tag{C.6}$$

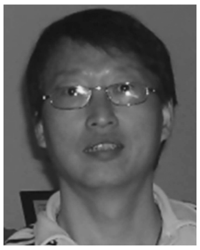
Acknowledgment

The authors would like to thank Dr. Mohamed El-Sharkawi of the University of Washington for the valuable discussions and materials about the projected gradient method.

References

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surveys*, vol. 38, no. 4, 2006.
[2] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003.
[3] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.
[4] R. T. Collins, "Mean-shift blob tracking through scale space," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003, vol. 2, pp. 234–240.
[5] V. Yang, R. Duraiswami, and L. Davis, "Efficient mean-shift tracking via a new similarity measure," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005, vol. 1, pp. 176–183.
[6] A. Yilmaz, "Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007, pp. 1–6.
[7] I. Leichter, M. Lindenbaum, and E. Rivlin, "Tracking by affine kernel transformations using color and boundary cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 1, pp. 164–171, Jan. 2009.
[8] G. D. Hager, M. Dewan, and C. V. Stewart, "Multiple kernel tracking with SSD," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004, vol. 1, pp. 790–797.
[9] B. Martinez, L. Ferraz, X. Binefa, and J. Diaz-Caro, "Multiple kernel two-step tracking," in *Proc. IEEE Int. Conf. Image Processing*, 2006, pp. 2785–2788.
[10] F. Porikli and O. Tuzel, "Multi-kernel object tracking," in *Proc. IEEE Int. Conf. Multimedia and Expo.*, 2005, pp. 1234–1237.
[11] Z. Fan, Y. Wu, and M. Yang, "Multiple collaborative kernel tracking," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005, vol. 2, pp. 502–509.
[12] H. Zhang, W. Huang, Z. Huang, and L. Li, "Affine object tracking with kernel-based spatial-color representation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005, vol. 1, pp. 293–300.
[13] C. Chu, J. Hwang, H. Pai, and K. Lan, "Robust video object tracking based on multiple kernels with projected gradients," in *Proc. IEEE Conf. Acoustics, Speech and Signal Processing*, May 2011, pp. 1421–1424.
[14] J. A. Snyman, *Practical Mathematical Optimization*. New York, NY, USA: Springer Science + Business Media, 2005, ch. 3.
[15] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006, vol. 1, pp. 260–267.
[16] S. Wang, H. Lu, F. Yang, and M. Yang, "Superpixel tracking," in *Proc. IEEE Int. Conf. Computer Vision*, 2011.
[17] J. Fang, J. Yang, and H. Liu, "Efficient and robust fragments-based multiple kernels tracking," *Int. J. Electron. Commun.*, vol. 65, pp. 915–923, 2011.
[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[19] CAVIAR: Context Aware Vision using Image-based Active Recognition, EC founded CAVIAR project/IST 2001 37540. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.
[20] in *PETS 2010: 13th IEEE Int. Workshop Performance Evaluation of Tracking and Surveillance*, 2010. [Online]. Available: http://pets2010.net/.
[21] UK Home Office, The Image Library for Intelligent Detection Systems (i-LIDS): Multiple Camera Tracking (MCT), 2008. [Online]. Available: http://www.homeoffice.gov.uk/science-research/hosdb/i-lids/.
[22] M. Rodriguez, J. Sivic, I. Laptev, and J. Audibert, "Data-driven crowd analysis in videos," in *Proc. IEEE Int. Conf. Computer Vision*, 2011, pp. 1235–1242.
[23] D. Doermann and D. Mihalcik, "Tools and techniques for video performance evaluation," in *Proc. IEEE Int. Conf. Pattern Recognition*, 2000, pp. 167–170. [Online]. Available: http://viper-toolkit.sourceforge.net/.
[24] C. Chu, J. Hwang, S. Wang, and Y. Chen, "Human tracking by adaptive Kalman filtering and multiple kernels tracking with projected gradients," in *Proc. ACM/IEEE Int. Conf. Distributed Smart Cameras*, 2011.
[25] i-Lids dataset for AVSS 2007. [Online]. Available: http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html.
[26] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "An adaptive color-based particle filter," *Image Vision Comput.*, pp. 99–110, 2003.
[27] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.
[28] M. Fashing and C. Tomasi, "Mean shift is a bound optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 471–474, Mar. 2005.
[29] A. Kembhavi, B. Siddiquie, R. Miezianko, S. McCloskey, and L. Davis, "Incremental multiple kernel learning for object recognition," in *Proc. IEEE Int. Conf. Computer Vision*, 2009, pp. 638–645.
[30] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "More efficiency in multiple kernel learning," in *Proc. ICML*, 2007.
[31] V. Parameswaran, V. Ramesh, and I. Zoghlami, "Tunable kernels for tracking," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006, vol. 2, pp. 2179–2186.
[32] X. Zhou, Y. Lu, J. Lu, and J. Zhou, "Abrupt motion tracking via intensively adaptive Markov-chain Monte Carlo sampling," *IEEE Trans. Image Process.*, vol. 21, no. 2, pp. 789–801, Feb. 2012.

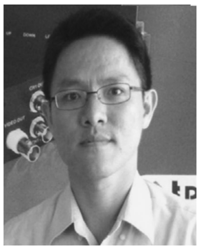
Chun-Te Chu received the B.S. degree in the Department of Electrical Engineering from National Taiwan University in 2006, and the M.S. degree in the Department of Electrical Engineering from University of Washington in 2010. He is currently pursuing the Ph.D. degree in Information Processing Lab at the Department of Electrical Engineering of University of Washington. His current research interests are in computer vision, machine learning, and video/image processing.



Jenq-Neng Hwang (F'01) received the B.S. and M.S. degrees, both in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 1981 and 1983 respectively. He then received his Ph.D. degree from the University of Southern California. In the summer of 1989, Dr. Hwang joined the Department of Electrical Engineering of the University of Washington in Seattle, where he has been promoted to Full Professor since 1999. He is currently the Associate Chair for Research in the Electrical Engineering Department. He has written more than 280 journal, conference papers and book chapters in the areas of multimedia signal processing, and multimedia system integration and networking, including a recent textbook on "Multimedia Networking: from Theory to Practice," published by Cambridge University Press. He has close working relationship with the industry on multimedia signal processing and multimedia networking.

Dr. Hwang received the 1995 IEEE Signal Processing Society's Best Journal Paper Award. He is a founding member of Multimedia Signal Processing Technical Committee of IEEE Signal Processing Society and was the Society's representative to IEEE Neural Network Council from 1996 to 2000. He is currently a member of Multimedia Technical Committee (MMTC) of IEEE Communication Society and also a member of Multimedia Systems and Applications Technical Committee (MSATC) of the IEEE Circuits and Systems Society. He served as an associate editor for IEEE T-SP, T-NN and T-CSVT, and is now an Associate Editor for IEEE T-IP and IEEE SIGNAL PROCESSING MAGAZINE, as well as an Editor for JISE, ETRI and IJDMB. He was the Program Co-Chair of ICASSP 1998 and ISCAS 2009.

Hung-I Pai received the B.S. and M.S. degrees in the Department of Physics from the National Chung Hsing University of Taiwan in 2001 and 2003 respectively, and the Ph.D. degree in the Department of Physics from the National Central University of Taiwan in 2008. He was with the Industrial Technology Research Institute, Taiwan, from 2008 to 2011. He is currently an Engineer of the Triple Domain Vision Company. His current research interests are in computer vision, video/image processing and statistical analyses.



Kung-Ming Lan received the M.S. degree in the Department of Industrial Engineering from the National Taipei University of Technology University of Taiwan in 2001 and 2003 respectively, and the Ph.D. degree in the Department of Industrial Engineering and Management from the National Tsing Hua University of Taiwan in 2008. He was with the Industrial Technology Research Institute, Taiwan, from 2008 to 2011. He is currently a Manager of the Triple Domain Vision Company. His current research interests are in computer vision, video/image processing and statistical analyses.