Generalized Kernel-Based Visual Tracking

Chunhua Shen, Junae Kim, and Hanzi Wang, *Member, IEEE*

Abstract—Kernel-based mean shift (MS) trackers have proven to be a promising alternative to stochastic particle filtering trackers. Despite its popularity, MS trackers have two fundamental drawbacks: 1) the template model can only be built from a single image, and 2) it is difficult to adaptively update the template model. In this paper, we generalize the plain MS trackers and attempt to overcome these two limitations. It is well known that modeling and maintaining a representation of a target object is an important component of a successful visual tracker. However, little work has been done on building a robust template model for kernel-based MS tracking. In contrast to building a template from a single frame, we train a robust object representation model from a large amount of data. Tracking is viewed as a binary classification problem, and a discriminative classification rule is learned to distinguish between the object and background. We adopt a support vector machine for training. The tracker is then implemented by maximizing the classification score. An iterative optimization scheme very similar to MS is derived for this purpose. Compared with the plain MS tracker, it is now much easier to incorporate online template adaptation to cope with inherent changes during the course of tracking. To this end, a sophisticated online support vector machine is used. We demonstrate successful localization and tracking on various data sets.

Index Terms—Global mode seeking, kernel-based tracking, mean shift, particle filter, support vector machine.

I. INTRODUCTION

VISUAL localization/tracking plays a central role for many applications like intelligent video surveillance, smart transportation monitoring systems, etc. Localization and tracking algorithms aim to find the most similar region to the target in an image. Recently, kernel-based tracking algorithms [1]–[3] have attracted much attention as an alternative to particle filtering trackers [4]–[6]. One of the most crucial difficulties in robust tracking is the construction of representation models (likelihood models in Bayesian filtering trackers) that can accommodate illumination variations, deformable appear-

ance changes, partial occlusions, etc. Most current tracking algorithms use a single static template image to construct a target representation based on density models. For both kernel-based trackers and particle filtering trackers, a popular method is to exploit color distributions in simple regions (region-wise density models). Generally, semi-parametric kernel density estimation techniques are adopted. However, it is difficult to update this target model [1], [2], [4], [7], and the target representation's fragility usually breaks these trackers over a long image sequence.

Considerable effort has been expended to ease these difficulties. We believe that the key to finding a solution is to find the right representation. In order to accommodate appearance changes, the representation model should be learned from as many training examples as possible. Fundamentally two methods, namely online and offline learning, can be used for the training procedure. Online learning means constantly updating the representation model during the course of tracking. Lim *et al.* [8] proposed an incremental eigenvector update strategy to adapt the target representation model. A linear probabilistic principal component analysis model is used. The main disadvantage of the eigen-model is that it is not generic and is usually only suitable for characterizing texture-rich objects. In [9], a wavelet model is updated using the expectation maximization algorithm. A classification function is progressively learned using AdaBoost for visual detection and tracking in [10] and [11], respectively. Han and Davis [12] adopted pixel-wise Gaussian mixture models to represent the target model and sequentially updated them. To date, however, less work has been reported on how to elegantly update *region-wise density* models in tracking.

In contrast, classification[1] is a powerful bottom-up procedure: it is trained offline and works online. Due to the training being typically built on very large amounts of training data, its performance is fairly promising even without online updating of the classifier/detector. Inspired by image classification tasks with color density features and real-time detection, we learn offline a density representation model from multiple training data. By considering tracking as a binary classification problem, a discriminative classification rule is learned to distinguish between the tracked object and background patterns. In this way, a robust object representation model is obtained. This proposal provides a basis for considering the design of enhanced kernel-based trackers using robust kernel object representations. A by-product of the training is the classification function, with which the

C. Shen is with NICTA, Canberra Research Laboratory, Canberra, ACT 2601, Australia (e-mail: chunhua.shen@nicta.com.au).

J. Kim is with the Research School of Information Science and Engineering, Australian National University, Canberra, ACT 0200, Australia, and also with NICTA, Canberra Research Laboratory, Canberra, ACT 2601, Australia (e-mail: jkim@rsise.anu.edu.au).

H. Wang is with the School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia (e-mail: hanzi.wang@ieee.org).

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

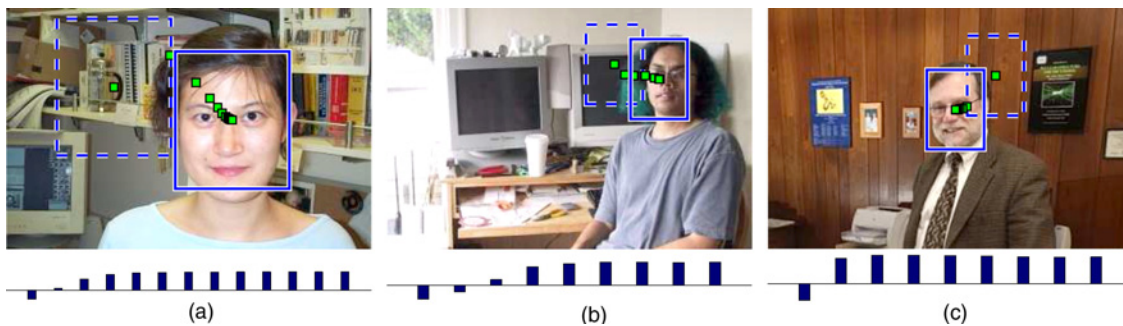[1]Object detection is typically a classification problem.

Fig. 1.   Examples of the initial position (dashed line) and the final convergence position (solid line). Squared dots show the optimization convergence trajectory. The image size in all tests is 320×211. The object size is 60×50 for the first example and 35×25 for the other two. The bars under every test image indicate the SVM score at each gradient-ascent iteration. The SVM score change is: (a) initial: -1.41, final: 2.77; (b) initial: -0.86, final: 1.41; and (c) initial: -1.04, final: 1.62.

tracking problem is cast into a binary classification problem. An object detector directly using the classification function is then available. Combining a detector into the tracker makes the tracker more robust and provides the capabilities of automatic initialization and recovery from momentary tracking failures.

In theory, many classifiers can be used to achieve our goal. In this paper, we show that the popular kernel-based nonlinear support vector machine (SVM) well fits the kernel-based tracking framework. Within this framework the traditional kernel object trackers proposed in [1] and [13] can be expressed as special cases. Because we use probabilistic density features, the learning process is closely related to probabilistic kernels-based SVMs [14]–[17]. It is imperative to minimize computational costs for real-time applications, such as tracking. A desirable property of the proposed algorithm is that the computational complexity is independent of the number of support vectors. Furthermore, we empirically demonstrate that our algorithm requires fewer iterations to achieve convergence.

Our approach differs from [18] although both use the SVM classification score as the cost function. In [18], Avidan builds a tracker along the line of standard optical flow tracking. Only the homogeneous quadratic polynomial kernel (or kernels with a similar quadratic structure) can be used in order to derive a closed-form solution. This restriction prevents one from using a more appropriate kernel obtained by model selection. An advantage of [18] is that it can be used consistently with the optical flow tracking, albeit only gray pixel information can be used. Moreover, the optimization procedure of our approach is inspired by the kernel-based object tracking paradigm [1]. Hence extended work such as [2] is also applicable here, which enables us to find the global optimum. If joint spatial-feature density is used to train an SVM, a fixed-point optimization method may also be derived that is similar to [13]. The classification function of the SVM trained for vehicle recognition is not smooth mis-registration (see Fig. 1 in [19]). We employ a spatial kernel to smooth the cost function when computing the histogram feature. In this way, gradient-based optimization methods can be used. Using statistical learning theory, we devise an object tracker that is consistent with mean shift (MS) tracking. The MS tracker is initially derived from kernel density estimation (KDE). Our work sheds some light on the connection between SVM and KDE.[2]

Another important part of our tracker is its online retraining in parallel with tracking. Continuous updating of the representation model can capture changes of the target appearance/backgrounds. Previous work such as [8], [9], [11], and [12] has demonstrated the importance of this online update during the course of tracking. The incremental SVM technique meets this end [22]–[25], which efficiently updates a trained SVM function whenever a sample is added to or removed from the training set. For our proposed tracking framework, the target model can be learned in either batch SVM training or online SVM learning. We adopt a sophisticated online SVM learning proposed in [24] for its efficiency and simplicity. We address the crucial problem of adaptation, i.e., the online learning of discriminant appearance model while avoiding drift.

The main contributions of our work are to solve MS trackers' two drawbacks: the template model can only be built from a single image; and it is difficult to update the model. The solution is to extend the use of statistical learning algorithms for object localization and tracking. SVM has been used for tracking by means of spatial perturbation of the SVM [18]. We exploit SVM for tracking in a novel way (along the line of MS tracking). The key ingredients of our approach are as follows.

1) Probabilistic kernel-based SVMs are trained and incorporated into the framework of MS tracking. By carefully selecting the kernel, we show that no extra computation is required compared with the conventional single-view MS tracking.
2) An online SVM can be used to adaptively update the target model. We demonstrate the benefit of online target model update.
3) We show that the annealed MS algorithm proposed in [2] can be viewed as a special case of the continuation method under an appropriate interpretation. With the new interpretation, annealed MS can be extended to

---

[2]It is believed that statistical learning theory (SVM and many other kernel learning methods) can be interpreted in the framework of information theoretic learning [20], [21].

more general cases. Extension and new discovers are discussed. An efficient localizer is built with global mode seeking techniques.

4) Again, by exploiting the SVM binary classifier, it is able to determine the scale of the target. An improved annealed MS-like algorithm with a cascade architecture is developed. It enables a more systematic and easier design of the annealing schedule, in contrast with ad hoc methods in previous work [2].

The remainder of this paper is organized as follows. In Section II, the general theory of MS tracking and SVM is reviewed for completeness. Our proposed tracker is presented in Section III. Finally, experimental results are reported in Section IV. We conclude this paper in Section V.

II. PRELIMINARIES

For self-completeness, we review mean shift tracking, support vector machine, and its online learning version in this section.

A. Mean Shift (MS) Tracking

Mean shift tracking was first presented in [1]. In MS tracking, the object is represented by a square region which is cropped and normalized into a unit circle. By denoting $\mathbf{q}$ as the color histogram of the target model, and $\mathbf{p}(\mathbf{c})$ as the target candidate color histogram with the center at $\mathbf{c}$, the similarity function between $\mathbf{q}$ and $\mathbf{p}(\mathbf{c})$ is (when Bhattacharyya divergence [1] is used)

$$\text{dist}(\mathbf{q}, \mathbf{p}(\mathbf{c})) = \sqrt{1 - \varrho(\mathbf{q}, \mathbf{p})}.$$

Here $\varrho(\mathbf{q}, \mathbf{p}) = \sqrt{\mathbf{q}}^T \sqrt{\mathbf{p}}$ is the dissimilarity measurement. Let $\{\mathbf{I}_\ell\}_{\ell=1}^n$ be a region's pixel positions in image $\mathbf{I}$ with the center at $\mathbf{c}$. In order to make the cost function smooth—otherwise gradient-based MS optimization cannot be applied—a kernel with profile $k(\cdot)$ is employed to assign smaller weights to those pixels farther from the center, considering the fact that the peripheral pixels are less reliable. An m-bin color histogram is built for an image patch located at $\mathbf{c}$, $\mathbf{q}(\mathbf{c}) = \{q_u(\mathbf{c})\}_{u=1}^m$, where

$$q_u = \lambda \sum_{\ell=1}^n k\left(\left\|\frac{\mathbf{c} - \mathbf{I}_\ell}{h}\right\|^2\right)\delta(\vartheta(\mathbf{I}_\ell) - u). \tag{1}$$

Here $k(\cdot)$ is the homogeneous spatial weighting kernel profile and h is its bandwidth. $\delta(\cdot)$ is the delta function and λ normalizes $\mathbf{q}$. The function $\vartheta(\mathbf{I}_\ell)$ maps a feature of $\mathbf{I}_\ell$ into a histogram bin u. $\mathbf{c}$ is the kernel center; and for the target model usually $\mathbf{c} = 0$. The representation of candidate $\mathbf{p}$ takes the same form.

Given an initial position c_0, the problem of localization/tracking is to estimate a best displacement $\Delta \mathbf{c}$ such that the measurement $\mathbf{p}(c_0 + \Delta \mathbf{c})$ at the new location best matches the target $\mathbf{q}$, i.e.,

$$\Delta \mathbf{c}^\star = \text{argmin}_{\Delta \mathbf{c}}\text{dist}(\mathbf{q}, \mathbf{p}(c_0 + \Delta \mathbf{c})).$$

By Taylor expanding $\text{dist}(\mathbf{q}, \mathbf{p}(\mathbf{c}))$ at the start position c_0 and keeping only the linear item (first-order Taylor approximation),

the above optimization problem can be resolved by an iterative procedure

$$\mathbf{c}^{[\tau+1]} = \frac{\sum_{\ell=1}^n \mathbf{I}_\ell \widetilde{w}_\ell g(\|\frac{\mathbf{c}^{[\tau]} - \mathbf{I}_\ell}{h}\|^2)}{\sum_{\ell=1}^n \widetilde{w}_\ell g(\|\frac{\mathbf{c}^{[\tau]} - \mathbf{I}_\ell}{h}\|^2)} \tag{2}$$

where $g(\cdot) = -k'(\cdot)$ and the superscript $\tau = 0, 1, 2\ldots$, indexes the iteration step. The weights $\widetilde{w}_\ell$ are calculated as $\widetilde{w}_\ell = \sum_{u=1}^m \sqrt{\frac{q_u}{p_u(c_0)}}\delta(\vartheta(\mathbf{I}_\ell) - u)$. See [1] for details.

B. Support Vector Machines

We limit our explanation of the support vector machine classifiers algorithm to an overview.

Large margin classifiers have demonstrated their advantages in many vision tasks. SVM is one of the popular large margin classifiers [26] which has a very promising generalization capacity.

The linear SVM is the best understood and simplest to apply. However, linear separability is a rather strict condition. Kernels are combined into margins for relaxing this restriction. SVM is extended to deal with linearly nonseparable problems by mapping the training data from the input space into a high-dimensional, possibly infinite-dimensional, feature space, i.e., $\Phi(\cdot) : \chi \to \mathcal{F}$. Using the kernel trick, the map $\Phi(\cdot)$ is not necessarily known explicitly. Like other kernel methods, SVM constructs a symmetric and positive definite kernel matrix (Gram matrix) which represents the similarities between all training datum points. Given N training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, the kernel matrix is written as: $K_{ij} \equiv K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$, $i, j = 1, \ldots, N$. When K_{ij} is large, the labels of $\mathbf{x}_i$ and $\mathbf{x}_j$, y_i and y_j are expected to be the same. Here, $y_i, y_j \in \{+1, -1\}$. The decision rule is given by sign $(f(\mathbf{x}))$ with

$$f(\mathbf{x}) = \sum_{i=1}^{N_S} \beta_i K(\hat{\mathbf{x}}_i, \mathbf{x}) + b \tag{3}$$

where $\hat{\mathbf{x}}_i \in \chi, i = 1, \ldots, N_S$ are support vectors, N_S is the number of support vectors, β_i is the weight associated with $\hat{\mathbf{x}}_i$, and b is the bias.

The training process of SVM then determines the parameters $\{\hat{\mathbf{x}}_i, \beta_i, b, N_S\}$ by solving the optimization problem

$$\begin{aligned} \text{minimize } & \frac{1}{2}\|\boldsymbol{\omega}\|_r^r + C\sum_{i=1}^N \xi_i \\ \text{subject to } & y_i(\boldsymbol{\omega}^T\Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \ \forall i \\ & \xi_i \geq 0 \ \forall i \end{aligned} \tag{4}$$

where $\boldsymbol{\xi} = \{\xi_i\}_{i=1}^N$ is the slack variable set and the regularization parameter C determines the trade-off between SVM's generalization capability and training error. $r = 1, 2$ corresponds to 1-norm and 2-norm SVM, respectively. The solution takes the form $\boldsymbol{\omega} = \sum_{i=1}^N y_i\alpha_i\Phi(\mathbf{x}_i)$. Here, $\alpha_i \geq 0$ and most of them are 0, yielding sparseness. The optimization (4) can be efficiently solved by linear programming (1-norm SVM) or quadratic programming (2-norm SVM) in its dual. Refer to [26] for details.

C. Online Learning with Kernels

A simple online kernel-based algorithm, termed NORMA, has been proposed for a variety of standard machine learning tasks in [24]. The algorithm is computationally cheap at each update step. We have implemented NORMA here for online SVM learning. See Fig. 1 in [24] for the backbone of the algorithm. We omit the details due to space constraint.

As mentioned, visual tracking is naturally a time-varying problem. An online learning method allows updating the model during the course of tracking.

III. GENERALIZED KERNEL-BASED TRACKING

The standard kernel-based MS tracker is generalized by maximizing a sophisticated cost function defined by SVM.

A. Probability Product Kernels

Measuring the similarity between images and image patches is of central importance in computer vision. In SVMs, the kernel $K(\cdot, \cdot)$ plays this role. Most commonly used kernels such as Gaussian and polynomial kernels are not defined on the space of probability distributions. Recently various probabilistic kernels have been introduced, including the Fisher kernel [14], TOP [15], Kullback–Leibler kernel [16], and probability product kernels (PPK) [17] to combine generative models into discriminative classifiers. A probabilistic kernel is defined by first fitting a probabilistic model $\mathbf{p}(\mathbf{x}_i)$ to each training vector $\mathbf{x}_i$. The kernel is then a measure of similarity between probability distributions. PPK is an example [17], with kernel given by

$$K_\rho^\star(\mathbf{q}(\mathbf{x}), \mathbf{p}(\mathbf{x})) = \int_\chi \mathbf{q}(\mathbf{x})^\rho \mathbf{p}(\mathbf{x})^\rho d\mathbf{x} \qquad (5)$$

where ρ is a constant. When $\rho = \frac{1}{2}$, PPK reduces to a special case, termed the Bhattacharyya kernel

$$K_{\frac{1}{2}}^\star(\mathbf{q}(\mathbf{x}), \mathbf{p}(\mathbf{x})) = \int_\chi \sqrt{\mathbf{q}(\mathbf{x})} \sqrt{\mathbf{p}(\mathbf{x})} \, d\mathbf{x}. \qquad (6)$$

In the case of discrete histograms, i.e., $\mathbf{q}(\mathbf{x}) = [q_1 \cdots q_m]^T$ and $\mathbf{p}(\mathbf{x}) = [p_1 \cdots p_m]^T$, (6) becomes

$$K_{\frac{1}{2}}^\star(\mathbf{q}(\mathbf{x}), \mathbf{p}(\mathbf{x})) = \sqrt{\mathbf{q}(\mathbf{x})}^T \sqrt{\mathbf{p}(\mathbf{x})} = \sum_{u=1}^m \sqrt{q_u p_u}. \qquad (7)$$

When $\rho = 1$, $K_1^\star(\cdot, \cdot)$ computes the expectation of one distribution over the other, and hence is termed the expected likelihood kernel [17]. In [27], its corresponding statistical affinity is used as similarity measurement for tracking.

The Bhattacharyya kernel is adopted in this paper due to the following reasons.

1) The standard MS tracker [1] uses the Bhattacharyya distance. It is clearer to show the connection between the proposed tracker and the standard MS tracker by using Bhattacharyya kernel.
2) It has been empirically shown, at least for image classification, that the generalization capability of expected likelihood kernel $K_1^\star(\cdot, \cdot)$ is weaker than the

Bhattacharyya kernel. Meanwhile, nonlinear probabilistic kernels including Bhattacharyya kernel, Kullback-Leibler kernel, Rényi kernel, etc., perform similarly [28]. Moreover, Bhattacharyya kernel is simple and has no kernel parameter to tune.

The PPK has an interesting characteristic that the mapping function $\Phi(\cdot)$ is explicitly known: $\Phi(\mathbf{q}(\mathbf{x})) = \mathbf{q}(\mathbf{x})^\rho$. This is equivalent to directly setting $\mathbf{x} = \mathbf{q}(\mathbf{x})^\rho$ and the kernel $K_\rho^\star(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$. Consequently for discrete PPK-based SVMs, in the test phase the computational complexity is independent of the number of support vectors. This is easily verified. The decision function is

$$f(\mathbf{x}) = \sum_{i=1}^{N_S} \beta_i \left[\mathbf{q}(\mathbf{x}_i)^\rho\right]^T \mathbf{p}(\mathbf{x})^\rho + b = \left[\sum_{i=1}^{N_S} \beta_i \mathbf{q}(\mathbf{x}_i)^\rho\right]^T \mathbf{p}(\mathbf{x})^\rho + b.$$

The first term in the bracket can be calculated beforehand. For example, for histogram-based image classification like [29], given a test image $\mathbf{x}$, the histogram vector $\mathbf{p}(\mathbf{x})$ is immediately available. In fact, we can interpret discrete PPK-based SVMs as *linear* SVMs in which the input vectors are $\mathbf{q}(\mathbf{x}_i)^\rho$—the features *nonlinearly*[3] extracted from image densities. Again, one might argue that, since the Bhattacharyya kernel is very similar to the linear SVM, it might not have the same power in modeling complex classification boundaries as the traditional nonlinear kernels like the Gaussian or polynomial kernel. The experiments in [28] indicate that the classification performance of a probabilistic kernel which consists of an exponential calculation is not clearly better: exponential kernels like the Kullback–Leibler kernel and Rényi kernel perform similarly as Bhattacharyya kernel on various datasets for image classification. Moreover our main purpose is to learn a representation model for visual tracking. Unlike other image classification tasks—in which high generalization accuracy is demanded—for visual tracking achieving very high accuracy might not be necessary and may not translate to a significant increase in tracking performance.

Note that PPKs are less compelling when the input data are vectors with no further structure. However, even the Gaussian kernel is a special case of PPK ($\rho = 1$ in (5) and $\mathbf{p}(\mathbf{x})$ is a single Gaussian fit to $\mathbf{x}_i$ by maximum likelihood) [17].

By contrast, the reduced set method is applied in [18] to reduce the number of support vectors for speeding up the classification phase. Applications which favor fast computation in the testing phase, such as large scale image retrieval, might also benefit from this discrete PPK's property.

B. Decision Score Maximization

It is well known that the magnitude of the SVM score $|f(\mathbf{x})|$ measures the *confidence* in the prediction. The proposed tracking is based on the assumption that the local maximum of the SVM score corresponds to the target location we seek, starting from an initial guess close to the target.

If the local maximum is positive, the tracker accepts the candidate. Otherwise an exhaustive search or localization

---

[3]When $\rho = 1$, it is linear. The nonlinear probabilistic kernels induce a transformed feature space (as the Bhattacharyya kernel does) to smooth density such that they significantly improve classification over the linear kernel [28].

process will start. The tracked position at time t is the initial guess of the next frame $t+1$ and so forth. We now show how the local maximum of the decision score is determined.

As in [1], a histogram representation of the image region can be computed as (1).

With (3), (7), and (1), we have[4]

$$f(\mathbf{c}) = \sum_{i=1}^{N_S} \beta_i \sum_{u=1}^{m} \sqrt{q_{i,u} p_u(\mathbf{c})} + b. \tag{8}$$

We assume the search for the new target location starts from a near position $\mathbf{c}_0$, and then a Taylor expansion of the kernel around $p_u(\mathbf{c}_0)$ is applied, similar to [1]. After some manipulations and putting those terms independent of $\mathbf{c}$ together, denoted by Δ, (8) becomes

$$\begin{aligned} f(\mathbf{c}) &= \frac{1}{2} \sum_{i=1}^{N_S} \beta_i \sum_{u=1}^{m} p_u(\mathbf{c}) \sqrt{\frac{q_{i,u}}{p_u(\mathbf{c}_0)}} + \Delta \\ &= \frac{\lambda}{2} \sum_{i=1}^{N_S} \beta_i \sum_{\ell=1}^{n} w_{i,\ell} k\left(\left\|\frac{\mathbf{c}-\mathbf{I}_\ell}{h}\right\|^2\right) + \Delta \\ &= \frac{\lambda}{2} \sum_{\ell=1}^{n} \hat{w}_\ell k\left(\left\|\frac{\mathbf{c}-\mathbf{I}_\ell}{h}\right\|^2\right) + \Delta \end{aligned} \tag{9}$$

where

$$w_{i,\ell} = \sum_{u=1}^{m} \sqrt{\frac{q_{i,u}}{p_u(\mathbf{c}_0)}} \delta(\vartheta(\mathbf{I}_\ell) - u) \tag{10}$$

and

$$\hat{w}_\ell = \sum_{i=1}^{N_S} \beta_i w_{i,\ell} = \sum_{u=1}^{m} \frac{\left[\sum_{i=1}^{N_S} \beta_i \sqrt{q_{i,u}}\right]}{\sqrt{p_u(\mathbf{c}_0)}} \delta(\vartheta(\mathbf{I}_\ell - u)). \tag{11}$$

Here (9) is obtained by swapping the order of summation. The first term of $f(\mathbf{c})$ is the weighted kernel density estimate with kernel profile $k(\cdot)$ at $\mathbf{c}$. It is clear now that our cost function $f(\mathbf{c})$ has an identical format as the standard MS tracker.

Can we simply set $\nabla_{\mathbf{c}} f(\mathbf{c}) = 0$ which leads to a fixed-point iteration procedure to *maximize* $f(\mathbf{c})$ as the standard MS does? If it works, the optimization would be similar to (2).

Unfortunately, $\nabla_{\mathbf{c}} f(\mathbf{c}) = 0$ cannot guarantee a local maximum convergence. This means that the fixed point iteration (2) can converge to a local minimum. We know that only when all the weights $\hat{w}_\ell$ are positive, (2) converges to a local maximum—as the standard MS does. See the Appendix for the theoretical analysis.

However, in our case, a negative support vector's weight β_i is negative, which means some of the weights computed by (11) could be negative. The traditional MS algorithm requires that the sample weights must be non-negative. Collins [30] has discussed the issue on MS with negative weights and a *heuristic* modification is given to make MS able to deal

with samples with negative weights. According to [30], the modified MS is

$$\mathbf{c}^{[\tau+1]} = \frac{\sum_{\ell=1}^{n} \mathbf{I}_\ell \hat{w}_\ell g\left(\left\|\frac{\mathbf{c}^{[\tau]}-\mathbf{I}_\ell}{h}\right\|^2\right)}{\sum_{\ell=1}^{n} |\hat{w}_\ell g\left(\left\|\frac{\mathbf{c}^{[\tau]}-\mathbf{I}_\ell}{h}\right\|^2\right)|}. \tag{12}$$

Here $|\cdot|$ is the absolute value operation. Alas this heuristic solution is problematic. Note that no theoretical analysis is given in [30]. We show that the methods in [30] cannot guarantee converging to a local maximum mode. See the Appendix for details.

The above problem may be avoided by using 1-class SVMs [31] in which $\hat{w}_\ell$ is strictly positive. However the discriminative power of SVM is also eliminated due to its unsupervised nature.

In this paper, we use a quasi-Newton gradient descent algorithm for maximizing $f(\mathbf{c})$ in (9). In particular, the L-BFGS algorithm [32] is adopted for implementing the quasi-Newton algorithm. We provide callbacks for calculating the value of the SVM classification function $f(\mathbf{c})$ and its gradient. Typically, only few iterations of the optimization procedure are performed at each frame. It has been shown that Quasi-Newton can be a better alternative to MS optimization for visual tracking [33] in terms of accuracy. Quasi-Newton was also used in [34] for kernel-based template alignment. Besides, in [2] the authors have shown that Quasi-Newton converges around twice faster than the standard MS does for data clustering.

The essence behind the proposed SVM score maximization strategy is intuitive. The cost function (8) favors both the dissimilarity to negative training data (e.g., background) and the similarity to positive training data. Compared to the standard MS tracking, our strategy provides the capability to utilize a large amount of training data. The terms with positive β in the cost function play the role to attract the target candidate while the negative terms repel the candidate. In [35], [36] Zhao *et al.* have extended MS tracking by introducing a background term to the cost function, i.e., $f(\mathbf{c}) = \lambda_f K_{\frac{1}{2}}^\star(\mathbf{q}, \mathbf{p}(\mathbf{c})) - \lambda_b K_{\frac{1}{2}}^\star(\mathbf{b}(\mathbf{c}), \mathbf{p}(\mathbf{c}))$. $\mathbf{b}(\cdot)$ is the background color histogram in the corresponding region. It also linearly combines both positive and negative terms into tracking and better performance has been observed. It is simple and no training procedure is needed. Nevertheless, it lacks an elegant means to exploit available training data and the weighting parameters λ_f and λ_b need to be tuned manually.[5]

The original MS tracker's analysis relies on kernel properties [1]. We argue that the main purpose of the kernel weighting scheme is to smooth the cost function such that iterative methods are applicable. Kernel properties then derive an efficient MS optimization. As observed by many other authors [33], [37], the kernels used as weighting kernel density estimation [38], [39]. We can simply treat the feature distribution as a weighted histogram to smooth the cost function and, at the same time, to account for the nonrigidity of tracked targets.

Note that: 1) the optimization reduces to the standard MS tracking if $N_S = 1$, and 2) other probability kernels like $K_1^\star(\cdot, \cdot)$

---

[4]We have used $\mathbf{x}$ to represent the image region. We also use the image center $\mathbf{c}$ to represent the image region $\mathbf{x}$. For clarity, we define notation $q_{i,u} \equiv q_u(\hat{\mathbf{x}}_i)$.

[5]Zhao *et al.* [35], [36] did not correctly treat MS iteration with negative weights, either, because they have used Collins' modified MS (12).

are also applicable here. The only difference is that $w_{i,\ell}$ in (10) will be in other forms.

In previous content, we have shown that in the testing phase discrete PPK's support vectors do not introduce extra computation. Again, for our tracking strategy, no computation overhead is introduced compared with the traditional MS tracking in [1]. This can be seen from (11). The summation in (11) (the bracketed term) can be computed offline. The only extra computation resides in the training phase; the proposed tracking algorithm has the *same* computation complexity as the standard MS tracker. It is also straightforward to extend this tracking framework to spatial-feature space [13] which has proved more robust.

C. Global Optimum Seeking

A technique is proposed in [2], dubbed annealed mean shift (AnnealedMS), to reliably find the global *density* mode. AnnealedMS is motivated by the observation that the number of modes of a kernel density estimator with a Gaussian kernel is monotonically non-increasing with respect to the bandwidth of the kernel.

Here we re-interpret this global optimization and show that it is essentially a special case of the *continuation* approach [40]. With the new interpretation, it is clear now that this technique is applicable to a broader type of cost functions, not necessary to a density function.

The continuation method is one of the unconstrained global optimization techniques which shares similarities with deterministic annealing. A series of gradually deformed but smoothed cost functions are successively optimized, where the solution obtained in the previous step serves as an initial point in the current step. This way the convergence information is conveyed. With sufficient smoothing, the first cost function will be concave/convex such that the global optimum can be found. The algorithm iterates until it traces the solution back to the original cost function. We now recall some basic concepts of the continuation method.

Definition 1: [40]. Given a non-linear function f, the transformation $\langle f \rangle_h$ for f is defined such that $\forall \mathbf{x}$

$$\langle f \rangle_h (\mathbf{x}) = C_h \int f(\mathbf{x}')k\left(\left\|\frac{\mathbf{x} - \mathbf{x}'}{h}\right\|^2\right)d\mathbf{x}' \qquad (13)$$

where $k(\cdot)$ is a smoothing function; usually the Gaussian is used. h is a positive scalar which controls the degree of smoothing. C_h is a normalization constant such that $C_h \int k(\|\frac{\mathbf{x}}{h}\|^2)d\mathbf{x} = 1$.

Note the similarity between the smoothing function $k(\cdot)$ and the definition of the kernel in KDE. From (13), the defined transformation is actually the convolution of the cost function with $k(\cdot)$. In the frequency domain, the frequency response of $\langle f \rangle_h$ equals the product of the frequency responses of f and k. Being a smoothing filter, the effect of $k(\cdot)$ is to remove high frequency components of the original function. Therefore, one of the requirements for $k(\cdot)$ is its frequency response must be a low-pass frequency filter. We know that popular kernels like Gaussian or Epanechnikov kernel are low-pass frequency filters. This is one of the principle justifications for using Gaussian or Epanechnikov to smooth a function. When h is
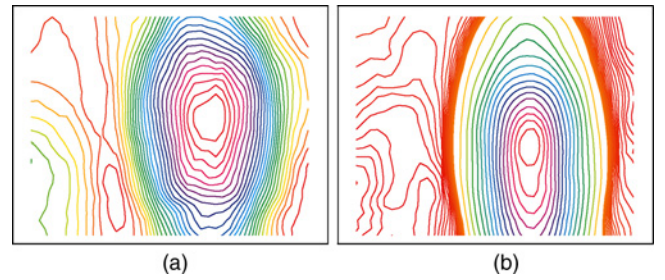

Fig. 2.   Close look at the cost function of the first example in Fig. 1. (a) SVM score. (b) Bhattacharyya distance of standard mean shift. Note that for the standard mean shift, the target model is extracted from the same test image; while for SVM, the target model is learned from a large number of training images that do not contain the test image.

increased, $\langle f \rangle_h$ becomes smoother and for $h = 0$, the function is the original function.

Theorem 1: The annealed version of mean shift introduced in [2] for global mode seeking is a special case of the general continuation method defined in (13).

Proof: Let the original function $f(\mathbf{x}')$ take the form of a Dirac delta comb (also known as impulse train in signal processing), *i.e.,* $f(\mathbf{x}') = \sum_i \delta(\mathbf{x}' - \hat{\mathbf{x}}_i)$, where $\hat{\mathbf{x}}_i$ is known. With the fundamental property that $\int F(\mathbf{x})\delta(\mathbf{x} - \hat{\mathbf{x}})d\mathbf{x} = F(\hat{\mathbf{x}})$ for any function $F(\cdot)$, we have $\langle f \rangle_h (\mathbf{x}) = C_h \sum_i k(\|\frac{\mathbf{x}-\hat{\mathbf{x}}_i}{h}\|^2)$. This is exactly the same as a KDE. This discovers that AnnealedMS is a special case of the continuation method. When $f(\mathbf{x}') = \sum_i w_i \delta(\mathbf{x}' - \hat{b}x_i)$ with $w_i \in \mathbb{R}$ (w_i can be negative), the above analysis still holds and this case corresponds to the SVM score maximization in Section III-B. ∎

It is not a trivial problem to determine the optimal scale of the spatial kernel bandwidth, i.e., the size of the target, for kernel-based tracking. A line search method is introduced in [30]. For AnnealedMS, an important open issue is how to design the annealing schedule. Armed with an SVM classifier, it is possible to determine the object's scale. If only the color feature is used, due to its lack of spatial information and insensitive to scale change, it is difficult to estimate a fine scale of the target. By combining other features, better estimates are expected. As we will see in the experiments, reasonable results can be obtained with only color. It is natural to combine AnnealedMS into a cascade structure, like the cascade detector of [41]. We start MS search from a large bandwidth h_0. After convergence, an extra verification is applied to decide whether to terminate the search. If $sign(f(\mathbf{I}_0)) = -1$, it means h_0 is too large. Then we need to reduce the bandwidth to h_1 and start MS with the initial location $\mathbf{I}_0$. This procedure is repeated until $sign(f(\mathbf{I}_m)) = +1$, $m \in \{0, \cdots, M\}$. h_m and $\mathbf{I}_m$ are the final scale and position. Little extra computation is needed because only a decision verification is introduced at each stage.

IV. EXPERIMENTS

In this section, we implement a localizer and tracker and discuss related issues. Experimental results on various data sets are shown.

A. Localization

For the first experiment, we have trained a face representation model. 404 faces cropped from CalTech-101 are used as
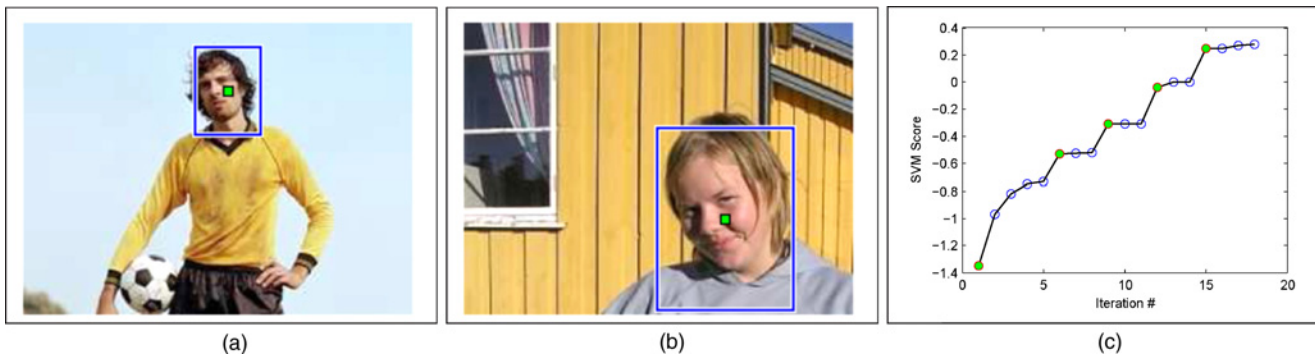
Fig. 3. Face localization. The final decision is marked with a rectangle. The image size in all tests is 240×180. In the first test (a), the proposed cascade localizer works very well. For the second one (b), the detected scale of the target is slightly big, but acceptable. The SVM scores for the first example are also plotted (c). The first iteration at each bandwidth is marked with a solid circle.
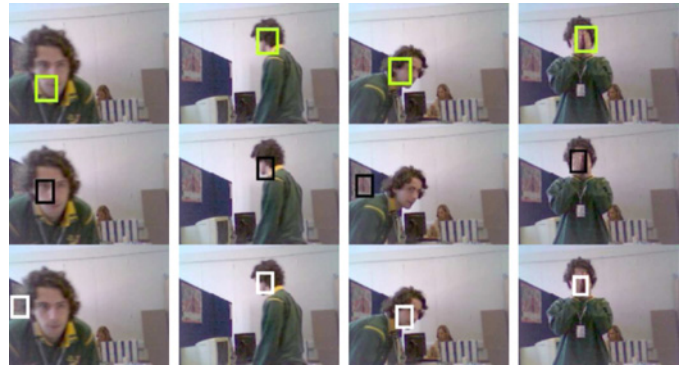

Fig. 4. *Face* sequence 1. Tracking results of the proposed tracker (top row); standard mean shift tracker (middle) and particle filtering (bottom row). Frames 26, 56, 318, and 432 are shown. The video size is 320×240 and the frame rate is 10 frames per second (f/s).

Fig. 5. *Face* sequence 2. Tracking results of the proposed tracker (top row); standard mean shift tracker (middle), and particle filtering (bottom row). Frames 86, 135, 204, and 512 are shown. The video size is 320×240 and the frame rate is 10 f/s.

positive raw images, and 1400 negative images are randomly cropped from images which do not contain faces. The image size is reduced to 42×56 pixels. Kernel-weighted RGB color histograms, consisting of $16 \times 16 \times 16$ bins, are extracted for classification. By default we use a soft SVM trained with library for support vector machines (LIBSVM) (slightly modified to use customized kernels). Test accuracy on the training data is 99.5% (1795/1804); and 91.7% (2752/3000) on a test data set which contains totally 3000 negative data. Note that our main purpose is not to train a powerful face detector; rather, we want to obtain an appearance model that is more robust than the single-view appearance model. We now test how well the algorithm maximizes the SVM score. First, we feed the algorithm a rough initial guess and run MS. See Fig. 1 for details.

The first example in Fig. 1 comes from the training data set. The initial SVM score is negative. In this case, a single step is required to switch to a positive score—it moves closely to the target after one iteration. We plot the corresponding cost function in Fig. 2. By comparison, the cost function of the standard MS is also plotted (the target template is cropped from the same image). We can clearly see the difference. The other two test images are from outside of the training data set. Despite the significant face color difference and variation in illumination, our SVM localizer works well in both tests. To compare the robustness, we use the first face as a template to

track the second face in Fig. 1, the standard MS tracker fails to converge to the true position.

We now apply the global maximum seeking algorithm to object localization. In [2], it has been shown that it is possible to locate a target no matter from which initial position the MS tracker starts. Here we use the learned classification rule to determine when to stop searching. We start the annealed continuation procedure with the initial bandwidth $h_0 = (42, 56)$. Then the bandwidth pyramid works with the rule $h_{m+1} = \frac{h_m}{1.25}$, $m \in \{0, \cdots, M\}$. M is the maximum number of iterations. We stop the search when for some m the SVM score is positive upon convergence. The image center is set to be the initial position of the search for these two tests. We present the results in Fig. 3.

In the first test, our proposed algorithm works well; it successfully finds the face location, and also the final bandwidth well fits the target. Fig. 3(c) shows how the SVM score evolves. It can be seen that every bandwidth change significantly increases the score. If the target size is large and there is a significant overlap between the target and a search region at a coarse bandwidth, h_m, the overlap can make the cascade search stop *prematurely* (see the second test in Fig. 3). Again this problem is mainly caused by the color feature's weak discriminative power. A remedy is to include more features. However, for certain applications where the scale-size is not critically important, our localization results have been usable. Furthermore, better results could be achieved when we

Fig. 6. *Walker* sequence 1. Tracking results of the proposed generalized kernel tracker. Frames 20, 40, 60, 90, 115, and 130 are shown. The video size is 384×288 and the frame rate is 25 f/s.

Fig. 7. *Walker* sequence 2. Tracking results of the proposed generalized kernel tracker. Frames 10, 55, 80, 105, 140, and 183 are shown. The video size and frame rate are the same as Fig. 6.

Fig. 8. *Walker* sequence 3. Tracking results of the proposed generalized kernel tracker. Frames 20, 98, 152, 220, 444, and 553 are shown. The video is of size 352×288 and frame rate 30 f/s.

train a model for a specific object (e.g., train an appearance model for a specific person) with a single color feature.

B. Tracking

Effectiveness of the proposed generalized kernel-based tracker is tested on a number of video sequences. We have compared with two popular color histogram-based methods: the standard MS tracker [1] and particle filters [4].

Unlike the first experiment, we do not train an *offline* SVM model for tracking. It is not easy to have a large amount of training data for a general object; therefore in the tracking experiment, an online SVM described in Section II-C is used for training. The user crops several negative data and positive data for initial training. During the course of tracking the online SVM updates its model by regarding the tracked region as a positive example and randomly selecting a few sub-regions (background area) around the target as negative examples. A $16 \times 16 \times 16$-binned color histogram is used for both the generalized kernel tracker and standard MS tracker. For the particle filter, with 1000 or 800 particles, the tracker fails at the first few frames. So we have used 1500 particles.

In the first experiment, the tracked person moves quickly. Hence, the displacement between neighboring frames is large. The illumination also changes. The background scene is cluttered and contains materials with similar color to the target. The proposed algorithm tracks the whole sequence successfully. Fig. 4 summarizes the tracking results. The standard MS tracker fails at frame #57, recovers at frame #74, and then fails again. The particle filter also loses the target due to motion blur and fast movement. Our online adaptive tracker achieves the most accurate results.
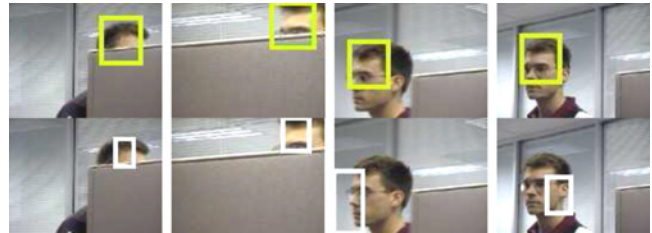

Fig. 9. *Cubicle* sequence 1. Tracking results of the proposed tracker (top) and particle filtering (bottom). Frames 16, 30, 41, 45 are shown. The video is of size 352×288 and 30 f/s.

Fig. 5 shows the results on a more challenging video. The target turns around and at some frames it even moves out of the view. At frame #194, the target disappears. Generalized kernel tracker and particle filter recovers at the following frames while the MS tracker fails. Again we can see the proposed tracker performs best due to its learned template model and online adaptivity. When the head turns around, all trackers can lock the target because compared with the background, the hair color is more similar to the face color. These two experiments show the proposed tracker's robustness to motion blur, large pose change, and target's fast movement over the standard MS tracker and particle filter-based tracker. In the experiments, to initialize the proposed tracker, we randomly pick up a few negative samples from the background. We have found this simple treatment works well.

We present more samples from three more sequences in Figs. 6–8. We mark only our tracker in these frames. From Figs. 6 and 7 we see that despite the target moving into shadow at some frames, our tracker successfully tracks the target through the whole sequences.
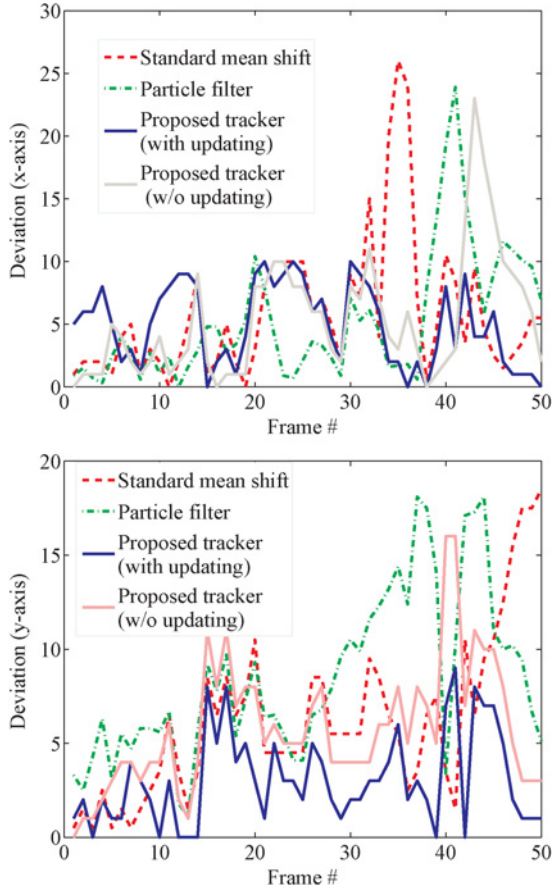
Fig. 10. ℓ_1-norm absolute error (pixels) of the object's center against the ground truth on the *Cubicle* sequence 1. The two figures correspond to x, and y-axis, respectively. The proposed tracker with online updating gives the best result. As expected, the proposed tracker without updating shows a similar performance to the standard MS tracker.
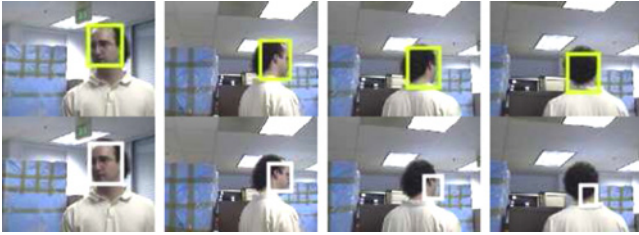

Fig. 11. *Cubicle* sequence 2. Tracking results of the proposed tracker (top) and particle filtering (bottom). Frames 9, 55, 60, and 64 are shown. The video is of size 352×288 and frame rate 30 f/s.

TABLE I
AVERAGE TRACKING ERROR AGAINST THE GROUND TRUTH (PIXELS) ON THE *Cubicle* SEQUENCE 1. THE MEAN AND STANDARD DEVIATION ARE REPORTED. WE ALSO REPORT THE TRACKING FAILURE RATES

	MS	Particle Filter	Ours (w/o Update)	Ours (Update)
Error	9.6 ± 5.7	10.5 ± 5.8	8.5 ± 4.9	6.5 ± 2.8
$FR_{0.20}$	44.0%	44.0%	28.0%	6.0%
$FR_{0.25}$	16.0%	34.0%	14.0%	0.0%

TABLE II
RUNNING TIME PER FRAME (SECONDS). THE STOCHASTIC PARTICLE FILTER TRACKER HAS RUN 5 TIMES AND THE STANDARD DEVIATION IS ALSO REPORTED

Sequence	MS	Particle Filter	Ours
Cubicle 1	0.0156	0.352 ± 0.025	0.0155
Walker 3	0.0169	0.331 ± 0.038	0.0142

TABLE III
AVERAGE TRACKING ERROR AGAINST THE GROUND TRUTH (PIXELS) ON THE *Cubicle* SEQUENCE 2. THE MEAN AND STANDARD DEVIATION ARE REPORTED. WE ALSO REPORT THE TRACKING FAILURE RATES

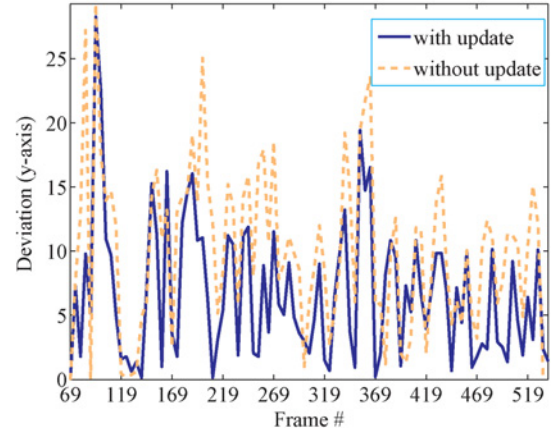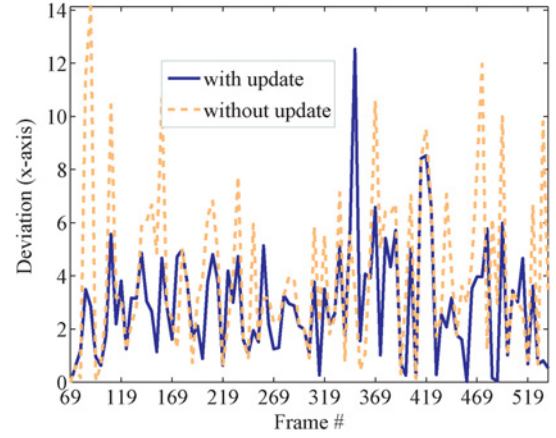|	MS	Particle Filter	Ours (w/o Update)	Ours (Update)
Error	5.7 ± 3.5	8.4 ± 3.4	5.5 ± 3.2	4.2 ± 2.8
$FR_{0.20}$	4.6%	15.4%	3.1%	0.0%
$FR_{0.25}$	0.0%	3.1%	0.0%	0.0%

Fig. 12. ℓ_1-norm absolute error (pixels) of the object's center against the ground truth on the *Walker* sequence 3. The two figures correspond to the x and y-axis, respectively. It clearly shows that the online update of the generalized kernel tracker is beneficial: without online update, the error is larger.

We have shown promising tracking results of the proposed tracker on several video clips. We now present some quantitative comparisons of our algorithm with other trackers.

First, we run the proposed tracker, MS, and particle filter trackers on the *Cubicle* sequence 1. In Fig. 9, we show some tracking frames of our method and particle filtering.

Compared with particle filtering, ours are much better in terms of accuracy and much faster in terms of the tracking speed. Our results are also *slightly* better than the standard MS tracker. But visually there is no significant difference, so we have not included MS results in Fig. 9.

Again, the particle filter tracker uses 1500 particles. We have run the particle filter five times and the best result is reported. Fig. 10 shows the absolute deviation of the tracked object's center at each frame. Clearly the generalized kernel tracker demonstrates the best result. We have reported the average tracking error (the Euclidean distance of the object's center against the ground truth) in Table I, which shows the proposed tracker outperforms MS and particle filter. In Table I, the error variance estimates are calculated from the tracking results of all frames regardless of the target is lost or not. We have also proved the importance of online SVM update. As mentioned, when we switch off the online update, our proposed tracker would behave similarly to the standard MS tracker. We see from Table I that even without updating, the generalized kernel tracker is slightly better than the standard MS tracker. This might be because the initialization schemes are different: the generalized kernel tracker can take multiple positive as well as negative training examples to *learn* an appearance model, while MS can only take a single image for initialization. Although we only use very few training examples (less than 10), it is already better than the standard MS tracker. In this sequence, when the target object is occluded, the particle filter tracker only tracks the visible region such that the deviation becomes large. Our approach updates the learned appearance model using online SVM. The region that partially contains the occlusion is added to the object class database gradually based on the online update procedure. This way our tracker tracks the object position close to the ground truth.

We also report the tracking failure rate (FR) for this video, which is the percentage of the number of failure frames in the total number of frames. If the distance between the tracked center and the ground truth's center is larger than a threshold, we mark it a failure. We have defined the threshold as 0.20 or 0.25 of the diagonal length of the ground truth's bounding box, which results in two criteria: $FR_{0.20}$ and $FR_{0.25}$ respectively. The former is more strict than the latter. As shown in Table I, our tracker with online update produces lowest tracking failures under either criterion.

We also compare the running time of trackers, which is an important issue for real-time tracking applications. Table II reports the results on two sequences.[6] The generalized kernel tracker (around 65 f/s) is comparable to the standard MS tracker, and much faster than the particle filter. This coincides with the theoretical analysis: our generalized kernel tracker's computational complexity is independent of the number of support vectors, so in the test phrase, the complexity is almost the same as the standard MS. One may argue that the online update procedure introduces some overhead. But the generalized kernel tracker employs the L-BFGS optimization
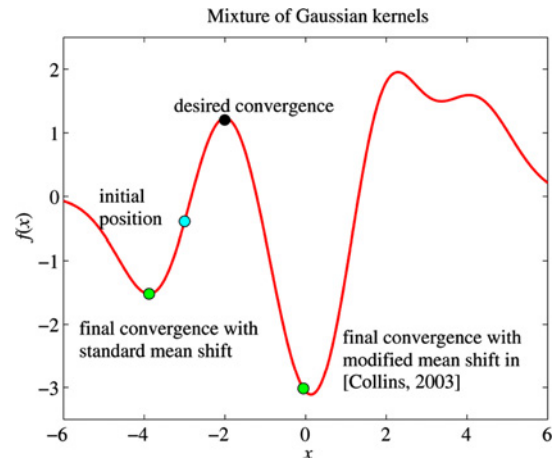

Fig. 13. With negative weights, the modified mean shift proposed in [30] may not be able to converge to the local maximum. In this case, it converges to a position close to a local minimum (not the exact minimum). The standard mean shift converges to the nearest minimum.

algorithm which is about twice faster than MS, as shown in [2]. Therefore, overall, the generalized kernel tracker runs as fast as the MS tracker. Because the particle filter is stochastic, we have run it 5 times and the average and standard deviation are reported. For our tracker and MS, they are deterministic and the standard deviation is negligible. Note the computational complexity if the particle filter tracker is linearly proportional to the number of particles.

We have run another test on *Cubicle* sequence 2. We show some results of our method and particle filtering in Fig. 11. Although all the methods can track this sequence successfully, the proposed method achieves most accurate results. We see that when the tracked object turns around, our algorithm is still able to track it accurately. Table III summarizes the quantitative performance. Our method is also slightly better MS. Again we see that online update does indeed improve the accuracy. We have also reported the tracking failure rates on this video. Our tracker with online update has the lowest tracking failures and the one without online update is the second best. These results are consistent with the previous experiments.

To demonstrate the effectiveness of the online SVM learning, we switch off the online update and run the tracker on the *Walker* sequence 3. We plot the ℓ_1-norm absolute deviation of the tracked object's center in pixels at each frame in Fig. 12. Apparently, at most frames, online update produces more accurate tracking results. The average Euclidean tracking error is 8.0 ± 4.9 pixels with online update and 12.7 ± 5.8 pixels without online update.

Conclusion that we can draw from these experiments are: 1) the proposed generalized kernel-based tracker performs better than the standard MS tracker on all the sequences that we have used, and 2) online learning often improves tracking accuracy.

V. CONCLUSION

To summarize, we have proposed a novel approach to kernel based visual tracking, which performs better than conventional

---

[6]All algorithms are implemented in ANSI C++. We have made the codes available at http://code.google.com/p/detect/. A desktop with Intel Core™ Duo 2.4-GHz CPU and 2-G RAM is used for running all the experiments.

single-view kernel trackers [1], [13]. Instead of minimizing the density distance between the candidate region and the template, the generalized MS tracker works by maximizing the SVM classification score. Experiments on localization and tracking show its efficiency and robustness. In this way, we show the connection between standard MS tracking and SVM based tracking. The proposed method provides a generalized framework to the previous methods.

Future work will focus on the following possible avenues.

1) Other machine learning approaches such as relevance vector machines (RVM) [42] might be employed to learn the representation model. Since in the test phrase, RVM and SVM take the same form, RVM can be directly used here. RVM achieves comparable recognition accuracy to the SVM, but requires substantially fewer kernel functions. It would be interesting to compare different approaches' performances.
2) The strategy in this paper can be easily plugged into a particle filter as an observation model. Improved tracking results are anticipated than for the simple color histogram particle filter tracker developed in [4].

APPENDIX

Generally, Collins' modified mean shift [30] (12) cannot guarantee to converge to a local maximum. It is obvious that a fixed point x^* obtained by iteration using (12) will not satisfy

$$\nabla f(x^*) = 0.$$

$f(\cdot)$ is the original cost function. Therefore, generally, x^* will not even be an extreme point of the original cost function. In the following example, x^* obtained by Collins' modified mean shift converges to a point which is close to a local *minimum*, but not the exact minimum.

In Fig. 13, we give an example on a mixture of Gaussian kernel which contains some negative weights. In this case, both the standard MS and Collins' modified MS fail to converge to a maximum.

REFERENCES

[1] 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.
[2] C. Shen, M. J. Brooks, and A. van den Hengel, "Fast global kernel density mode seeking: Applications to localization and tracking," *IEEE Trans. Image Process.*, vol. 16, no. 5, pp. 1457–1469, May 2007.
[3] W. Qu and D. Schonfeld, "Robust control-based object tracking," *IEEE Trans. Image Process.*, vol. 17, no. 9, pp. 1721–1726, Sep. 2008.
[4] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proc. Eur. Conf. Comp. Vis.*, Copenhagen, Denmark, Lecture Notes in Computer Science 2350, 2002, pp. 661–675.
[5] C. Shen, A. van den Hengel, and A. Dick, "Probabilistic multiple cue integration for particle filter-based tracking," in *Proc. Int. Conf. Digital Image Comput. Tech. Appl.*, Sydney, Australia, 2003, pp. 309–408.
[6] P. Pan and D. Schonfeld, "Dynamic proposal variance and optimal particle allocation in particle filtering for video tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 9, pp. 1268–1279, Sep. 2008.
[7] H. Wang, D. Suter, K. Schindler, and C. Shen, "Adaptive object tracking based on an effective appearance filter," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 9, pp. 1661–1667, Sep. 2007.
[8] J. Lim, D. Ross, R.-S. Lin, and M.-H. Yang, "Incremental learning for visual tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, vol. 17. Dec. 2004, pp. 801–808.
[9] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," in *Proc. IEEE Conf. Comp. Vis. Pattern Recognit.*, vol. 1. Kauai, 2001, pp. 415–422.
[10] O. Javed, S. Ali, and M. Shah, "Online detection and classification of moving objects using progressively improving detectors," in *Proc. IEEE Conf. Comp. Vis. Pattern Recognit.*, vol. 1. San Diego, CA, 2005, pp. 696–701.
[11] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.
[12] B. Han and L. Davis, "Online density-based appearance modeling for object tracking," in *Proc. IEEE Int. Conf. Comp. Vis.*, vol. 2. Beijing, China, Oct. 2005, pp. 1492–1499.
[13] A. Elgammal, R. Duraiswami, and L. S. Davis, "Probabilistic tracking in joint feature-spatial spaces," in *Proc. IEEE Conf. Comp. Vis. Pattern Recognit.*, vol. 1. Madison, WI, 2003, pp. 781–788.
[14] T. S. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Proc. Conf. Adv. Neural Inf. Process. Syst.*, 1998, pp. 487–493.
[15] K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller, "A new discriminative kernel from probabilistic models," *Neural Comput.*, vol. 14, no. 10, pp. 2397–2414, Oct. 2002.
[16] P. J. Moreno, P. Ho, and N. Vasconcelos, "A Kullback–Leibler divergence-based kernel for SVM classification in multimedia applications," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, Canada, 2003.
[17] T. Jebara, R. Kondor, and A. Howard, "Probability product kernels," *J. Mach. Learn. Res.*, vol. 5, pp. 819–844, Dec. 2004.
[18] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, Aug. 2004.
[19] O. Williams, A. Blake, and R. Cipolla, "Sparse Bayesian learning for efficient visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1292–1304, Aug. 2005.
[20] R. Jenssen, D. Erdogmus, J. C. Principe, and T. Eltoft, "Toward a unification of information theoretic learning and kernel methods," in *Proc. IEEE Workshop Mach. Learning Signal Process.*, Sao Luis, Brazil, 2004, pp. 93–102.
[21] R. Jenssen, D. Erdogmus, J. C. Principe, and T. Eltoft, "The Laplacian PDF distance: A cost function for clustering in a kernel feature space," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 17. 2004, pp. 625–632.
[22] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 409–415.
[23] G. Fung and O. L. Mangasarian, "Incremental support vector machine classification," in *Proc. Soc. Ind. Appl. Math. Int. Conf. Data Mining*, Arlington, VA, 2002.
[24] J. Kivinen, A. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.
[25] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *J. Mach. Learn. Res.*, vol. 6, pp. 1579–1619, Dec. 2005.
[26] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[27] C. Yang, R. Duraiswami, and L. Davis, "Efficient spatial-feature tracking via the mean-shift and a new similarity measure," in *Proc. IEEE Conf. Comp. Vis. Pattern Recognit.*, vol. 1. San Diego, CA, 2005, pp. 176–183.
[28] A. B. Chan, N. Vasconcelos, and P. J. Moreno, "A family of probabilistic kernels based on information divergence," Dept. Electr. Comput. Eng., Univ. California, San Diego, CA, Tech. Rep. SVCL-TR 2004/01, 2004.
[29] O. Chapelle, P. Haffner, and V. Vapnik, "SVMs for histogram-based image classification," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1055–1064, 1999.
[30] R. Collins, "Mean-shift blob tracking through scale space," in *Proc. IEEE Conf. Comp. Vis. Pattern Recognit.*, vol. 2. Madison, WI, 2003, pp. 234–240.
[31] B. Schölkopf, J. Platt, J. Shawe-Taylor, and A. Smola, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
[32] D. C. Liu and J. Nocedal, "On the limited memory method for large scale optimization," *Math. Programming B*, vol. 45, no. 3, pp. 503–528, Dec. 1989.
[33] T. Liu and H. Chen, "Real-time tracking using trust-region methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 3, pp. 397–402, Mar. 2004.
[34] I. Guskov, "Kernel-based template alignment," in *Proc. IEEE Conf. Comp. Vis. Pattern. Recognit.*, vol. 1. New York, 2006, pp. 610–617.

[35] T. Zhao and R. Nevatia, "Tracking multiple humans in crowded environment," in *Proc. IEEE Conf. Comp. Vis. Pattern. Recognit.*, vol. 2. Washington D.C., 2004, pp. 406–413.

[36] T. Zhao, R. Nevatia, and B. Wu, "Segmentation and tracking of multiple humans in crowded environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1198–1211, Jul. 2008.

[37] M. Dewan and G. D. Hager, "Toward optimal kernel-based tracking," in *Proc. IEEE Conf. Comp. Vis. Pattern Recognit.*, vol. 1. New York, Jun. 2006, pp. 618–625.

[38] M. P. Wand and M. C. Jones, *Kernel Smoothing*. London, U.K.: Chapman and Hall/CRC Press, 1995.

[39] 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.

[40] Z. Wu, "The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation," *Soc. Ind. Appl. Math. J. Optimization*, vol. 6, no. 3, pp. 748–768, Aug. 1996.

[41] P. A. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004.

[42] M. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, Jun. 2001.

Chunhua Shen received the B.S. and M.S. degrees from Nanjing University, Nanjing, China, in 1999 and 2002, respectively, and the Ph.D. degree from the University of Adelaide, Adelaide, Australia, in 2006.

He has been working as a Research Scientist with NICTA, Canberra Research Laboratory, Canberra, Australia since October 2005. He is also an Adjunct Research Fellow at Australian National University, Canberra, Australia and an Adjunct Lecturer at the University of Adelaide. His research interests include statistical machine learning and convex optimization, and their application in computer vision.

Junae Kim received the B.S. degree from Ewha Womans University, Seoul, Korea in 2000, the M.S. degree from the Pohang University of Science and Technology, Pohang, Korea in 2002, and the M.Sc. degree from Australian National University, Canberra, Australia in 2007. She is pursuing the Ph.D. degree from the Research School of Information Sciences and Engineering, Australian National University.

She is also affiliated with NICTA, Canberra Research Laboratory, Canberra, Australia. She was a Researcher at the Electronics and Telecommunications Research Institute, Daejeon, Korea, for five years before she moved to Australia. Her research interests include computer vision and machine learning.

Hanzi Wang (M'05–SM'09) received the B.S. degree in physics and the M.S. degree in optics from Sichuan University, Chengdu, China, in 1996 and 1999, respectively. He received the Ph.D. degree in computer vision from Monash University, Melbourne, Australia, in 2004.

He is currently a Senior Research Fellow at the School of Computer Science, University of Adelaide, Adelaide, Australia. He is the author of more than 30 papers in major international journals and conferences. His current research interests include computer vision and pattern recognition, including robust statistics, model fitting, optical flow calculation, visual tracking, image segmentation, fundamental matrix estimation, and related fields.