Int J Comput Vis (2008) 77: 125–141
DOI 10.1007/s11263-007-0075-7

# Incremental Learning for Robust Visual Tracking

**David A. Ross · Jongwoo Lim · Ruei-Sung Lin · Ming-Hsuan Yang**

Received: 6 September 2005 / Accepted: 17 July 2007 / Published online: 17 August 2007

**Abstract** Visual tracking, in essence, deals with non-stationary image streams that change over time. While most existing algorithms are able to track objects well in controlled environments, they usually fail in the presence of significant variation of the object's appearance or surrounding illumination. One reason for such failures is that many algorithms employ fixed appearance models of the target. Such models are trained using only appearance data available before tracking begins, which in practice limits the range of appearances that are modeled, and ignores the large volume of information (such as shape changes or specific lighting conditions) that becomes available during tracking. In this paper, we present a tracking method that incrementally learns a low-dimensional subspace representation, efficiently adapting online to changes in the appearance of the target. The model update, based on incremental algorithms for principal component analysis, includes two important features: a method for correctly updating the sample mean, and a forgetting factor to ensure less modeling power is expended fitting older observations. Both of these features contribute measurably to improving overall tracking performance. Numerous experiments demonstrate the effectiveness of the proposed tracking algorithm in indoor and outdoor environments where the target objects undergo large changes in pose, scale, and illumination.

**Keywords** Visual tracking · Subspace update · Online algorithms · Adaptive methods · Particle filter · Illumination

## 1 Introduction

Visual tracking essentially deals with non-stationary data, both the target object and the background, that change over time. Most existing algorithms are able to track objects, either previously viewed or not, in short durations and in well controlled environments. However these algorithms usually fail to observe the object motion or have significant drift after some period of time, due to drastic change in the object's appearance or large lighting variation in its surroundings. Although such situations can be ameliorated with recourse to richer representations, effective prediction schemes or combination, most algorithms typically operate on the premise that the model of the target object does not change drastically over time. Examples abound, ranging from representation methods based on view-based appearance models (Black and Jepson 1996), contours (Isard and Blake 1996), parametric templates of geometry and illumination (Hager and Belhumeur 1996), integration of shape and color (Birchfield 1998), mixture models (Black et al. 1998), 3D models (La Cascia and Sclaroff 1999), exemplars (Toyama and Blake 2001), foreground/background models (Harville 2002) templates with updating (Matthews et al.

D.A. Ross (✉)
University of Toronto, 10 Kings College Road, Toronto, ON, M55 3G4, Canada
e-mail: dross@cs.toronto.edu

J. Lim · M.-H. Yang
Honda Research Institute, 800 California Street, Mountain View, CA 94041, USA

J. Lim
e-mail: jlim@honda-ri.com

M.-H. Yang
e-mail: mhyang@ieee.org

R.-S. Lin
Motorola Labs, 1303 E Algonquin Rd., Schaumburg, IL 60196, USA
e-mail: ruei-sung.lin@motorola.com

2004); prediction schemes using particle filters (Isard and Blake 1996), joint probabilistic data association filters (Rasmussen and Hager 1998), kernel-based filters (Comaniciu et al. 2003; Georgescu et al. 2004), support vector machines (Avidan 2001; Williams et al. 2003) and variational inference (Vermaak et al. 2003). These algorithms usually build or learn a model of the target object first and then use it for tracking, without adapting the model to account for changes in the appearance of the object, e.g., large variation of pose or facial expression, or the surroundings, e.g., lighting variation. Furthermore, it is assumed that all images are acquired with a stationary camera. Such an approach, in our view, is prone to performance instability, thus needs to be addressed when building a robust visual tracker.

The chief challenge of visual tracking can be attributed to the difficulty in handling the appearance variability of a target object. Intrinsic appearance variability includes pose variation and shape deformation,whereas extrinsic illumination change, camera motion, camera viewpoint, and occlusions inevitably cause large appearance variation. Due to the nature of the tracking problem, it is imperative for a robust algorithm to model such appearance variation.

In this paper we propose a method[1] that, during visual tracking, efficiently learns and updates a low dimensional subspace representation of the target object. The advantages of this adaptive subspace representation are several fold. The subspace representation provides a compact notion of the "thing" being tracked rather than treating the target as a set of independent pixels, i.e., "stuff" (Adelson and Bergen 1991), and facilitates object recognition. An efficient incremental method continually updates the subspace model to reflect changes in appearance caused by intrinsic and extrinsic factors, thereby facilitating the tracking process. Incrementally updating the subspace removes the offline learning phase required by other eigentrackers, allowing one to track objects for which a database of training images is not even available. To estimate the locations of the target objects in consecutive frames, we use a sampling algorithm with likelihood estimates, which is in contrast to other tracking methods that usually solve complex optimization problems using gradient descent. Furthermore, while numerous algorithms operate under the assumption that there is no camera motion, our method is able to track objects without this constraint.

The remaining part of this paper is organized as follows. We begin, in the next section, by reviewing the most relevant algorithms that motivated this work. The details of our algorithm are described in Sect. 3, where we propose an efficient incremental subspace method with a mean update and forgetting factor, followed by an effective tracking algorithm. The results of numerous experiments and performance evaluation are presented in Sect. 4. We conclude this paper with remarks on potential extensions for future work. The data, source code, and videos corresponding to this work can all be found at http://www.cs.toronto.edu/~dross/ivt/.

## 2 Related Work and Motivation

There is a rich literature in visual tracking and a thorough discussion on this topic is beyond the scope of this paper. In this section we review only the most relevant visual tracking work, focusing on algorithms that operate directly on grayscale images. We contrast our method with these methods in terms of their representation scheme, target prediction approach, and their ability to handle changes in illumination as well as appearance.

Visual tracking problems have conventionally been formulated as prediction tasks within which fixed templates and optical flow techniques are utilized to estimate the motion of a target object (Lucas and Kanade 1981). Such approaches do not take the appearance variability into consideration, and thus perform well only over short periods of time. To enhance the robustness of such object trackers, Black and Jepson proposed an algorithm using a pre-trained view-based eigenbasis representation and a robust error norm (Black and Jepson 1996). Instead of relying on the brightness constancy principal assumed in optical flow techniques, they advocated the use of a subspace constancy assumption for motion estimation. Although their algorithm demonstrated excellent empirical results, it entailed learning a set of view-based eigenbases before the tracking task began. To achieve robust visual tracking with this method, it is imperative to collect a large set of training images covering the range of possible appearance variation (including viewing angles and illumination) from which to construct the eigenbasis, as this representation, once learned, is not updated.

Observing that low-dimensional linear subspaces are able to effectively model image variation due to illumination (Belhumeur and Kreigman 1997), Hager and Belhumeur developed a tracking algorithm to handle the appearance variation caused by illumination and pose change using parametric models (Hager and Belhumeur 1996). Their method extends a gradient-based optical flow algorithm by incorporating low-dimensional representations (Belhumeur and Kreigman 1997) for object tracking under varying illumination conditions. Before tracking begins, a set of illumination bases needs to be constructed at a fixed pose in order to account for changes in appearance due to lighting variation. However, this basis does not attempt to account for changes in pose such as out-of-plane rotations.

Realizing the limitations of having a single (unimodal or Gaussian) hypothesis of target location at each timestep—as produced by the Kalman filter and its relatives—Isard

[1] Preliminary results of this paper were presented in Ross et al. (2004) and Lim et al. (2005).

and Blake introduced particle filters to visual tracking and presented the Condensation algorithm for contour tracking in which multiple plausible interpretations are propagated over time (Isard and Blake 1996). This probabilistic approach has demonstrated success in tracking the outline of target objects in clutter. However, the representation scheme employed (curves or splines) ignores the internal appearance of the target, and is not updated to account for variations in its appearance, due to pose or illumination change.

Supervised discriminative methods for classification and regression have also been exploited to solve visual tracking problems. For example, Avidan (2001) developed a tracking algorithm that employs the support vector machine (SVM) classifier within a optic flow framework. Avidan modified the conventional use of the SVM classification score to instead predict target location, by computing image gradients as is done in optical flow algorithms. Although this algorithm has demonstrated success in tracking specific objects, e.g., cars from a mounted camera in a moving vehicle, significant effort is required in training a SVM. Along similar lines, Williams et al. developed a method in which an SVM-based regressor was used for tracking (Williams et al. 2003). Instead of relying on optical flow to predict object location, they learned a perturbation function of spatial in-plane displacements between frames, thereby predicting the most likely object location. As a result of training the regressor on in-plane image motion, this method is not effective in tracking objects with out-of-plane movements.

Mixture models have been studied as alternatives to linear representations, to better account for appearance change in motion estimation. Black et al. (1998) identified four possible factors causing appearance change, fitting them with a mixture model which was then used to estimate image motion. A more elaborate mixture model fit via an online EM algorithm was recently proposed by Jepson et al. (2003), in which three components were used to model the responses of wavelet filters, and thereby account for appearance variation during tracking. Their method is able to handle variations in pose, illumination and expression. However, their appearance model treats pixels within the target region independently (ignoring their covariance) and thus does not have notion of the "thing" being tracked. This can result in modeling background rather than the foreground, thereby failing to track the target object (Jepson et al. 2003).

Attempts to improve the classic Lucas–Kanade tracker (Lucas and Kanade 1981) with updates was recently made by Matthews et al. (2004). They developed a template update method for visual tracking, which employs an active appearance model (Cootes et al. 2001) to account for image variation. Thus instead of using a fixed template, the object appearance is modeled by a linear combination of appearance images. The tracking problem is then formulated as a search (using gradient descent) for the affine parameters and linear combination which minimize the difference between the target object and the current appearance model. The newly tracked object is then used to update appearance model, as necessary. They demonstrated good tracking results on vehicles and faces with varying expressions. However, the authors noted that the computation cost for updating the template increases dramatically as principal component analysis is carried out at each update, and that their work covers the case where the visibility of the target object does not change.

Our work is motivated in part by the prowess of subspace representations as appearance models (Murase and Nayar 1995; Belhumeur and Kreigman 1997), the effectiveness of particle filters (Isard and Blake 1996), and the adaptability of on-line update schemes (Jepson et al. 2003). In contrast to the eigentracking algorithm (Black and Jepson 1996), our algorithm does not require a training phase but learns the eigenbases on-line during the object tracking process. Thus our appearance model can adapt to changes in pose, view angle, and illumination not captured by the set of training images—in fact the need to manually collect training images prior to tracking is eliminated. Further, our method uses a particle filter for motion parameter estimation rather than the gradient descent method, which often gets stuck in local minima or is distracted by outliers (Black and Jepson 1996). Our appearance-based model provides a richer description than simple curves or splines as used in (Isard and Blake 1996), and has a notion of the "thing" being tracked. In addition, the learned representation can be utilized for other tasks such as object recognition. With respect to the template update method (Cootes et al. 2001), we concurrently developed an efficient subspace update algorithm that facilitates object tracking under varying pose and lighting conditions. Furthermore, our algorithm is able to handle camera motion while learning compact representations and tracking objects. In this work, an eigenbasis representation is learned directly from pixel values corresponding to a target object in the image space. Experiments show that good tracking results can be obtained using this representation without employing more complicated wavelet features as in Jepson et al. (2003), although this elaboration is still possible and may lead to even better results. Note also that the view-based eigenbasis representation has demonstrated its ability to model the appearance of objects in different poses (Murase and Nayar 1995), and under different lighting conditions (Belhumeur and Kreigman 1997). Consequently, the learned eigenbasis facilitates tracking objects undergoing illumination and pose change.

## 3 Incremental Learning for Tracking

We present details of the proposed incremental learning algorithm for object tracking in this section. First we propose an efficient method that incrementally updates an eigenbasis as new observations arrive, which is used to learn the appearance of the target while tracking progresses. Next we describe our approach for drawing particles in the motion parameter space and predicting the most likely object location with the help of the learned appearance model. Collectively, we show how these two modules work in tandem to track objects well under varying conditions.

### 3.1 Incremental Update of Eigenbasis and Mean

The appearance of a target object may change drastically due to intrinsic and extrinsic factors as discussed earlier. Therefore, to produce a robust tracker, it is important to adapt the appearance model online, while tracking, to reflect these changes. The appearance model we have chosen, a eigenbasis, is typically learned off-line from a set of training images $\{I_1, \dots, I_n\}$, by taking computing the eigenvectors $U$ of the sample covariance matrix $\frac{1}{n-1}\sum_{i=1}^{n}(I_i - \bar{I})(I_i - \bar{I})^\top$, where $\bar{I} = \frac{1}{n}\sum_{i=1}^{n} I_i$ is the sample mean of the training images. Equivalently one can obtain $U$ by computing the singular value decomposition $U\Sigma V^\top$ of the centered data matrix $[(I_1 - \bar{I}) \cdots (I_n - \bar{I})]$, with columns equal to the respective training images minus their mean.

Adapting the appearance model to account for novel views of the target can be thought of as retraining the eigenbasis with an additional $m$ images $\{I_{n+1}, \dots, I_{n+m}\}$, for some value of $m$. Naively, this update could be performed by computing the singular value decomposition $U'\Sigma'V'^T$ of the augmented (centered) data matrix $[(I_1 - \bar{I}') \cdots (I_{n+m} - \bar{I}')]$, where $\bar{I}'$ is the average of the entire $n+m$ training images.

Unfortunately this approach is unsatisfactory for online applications, like visual tracking, due to its storage and computational requirements. First, the naive approach uses the entire set of training images for each update. If an update is made at each video frame, then the number of images which must be retained grows linearly with the length of the sequence. Second, the cost of computing the mean and singular value decomposition grows with the number of images, so the algorithm will run ever slower as time progresses. Instead, the requirements of our application dictate that any algorithm for updating the mean and eigenbasis must have storage and computational requirements that are constant, regardless of the number of images seen so far.

Numerous, more-sophisticated algorithms have been developed to efficiently update an eigenbasis as more data arrive (Golub and Van Loan 1996; Hall et al. 1998; Levy and Lindenbaum 2000; Brand 2002). However, most methods assume the sample mean is fixed when updating the eigenbasis, or equivalently that the data is inherently zero-mean. Neither assumption is appropriate in our application. An exception is the method by Hall et al. (2002), which does consider the change of the mean as each new datum arrives. Although similar to our (independently-developed) algorithm, it lacks the forgetting factor, which hurts its suitability for tracking, and has a greater computational cost. (Both of these disadvantages are demonstrated quantitatively in Sect. 4.3.) Part of the additional complexity comes, because Hall's algorithm is based on the notion of adding eigenspaces, from computing the eigenvalue decomposition of each block of new data as it arrives. In this respect our algorithm is simpler, since it incorporates new data directly, without the additional step.

Here we extend one of these efficient update procedures—the Sequential Karhunen–Loeve (SKL) algorithm of Levy and Lindenbaum (2000)—presenting a new incremental PCA algorithm that correctly updates the eigenbasis as well as the mean, given one or more additional training data. Our algorithm, a variation of which was first presented in Lim et al. (2005), has also been applied to algorithms where the subspace mean plays an important role. For example, it can be applied to adaptively update the between-class and within-class covariance matrices used in Fisher linear discriminant analysis (Lin et al. 2005). We begin with a summary of the SKL algorithm, then describe our new incremental PCA algorithm, and follow with a discussion of a forgetting factor which can be used to down-weight the effect of earlier observations on the PCA model.

Putting aside for the moment the problem of the sample mean, suppose we have a $d \times n$ data matrix $A = \{\mathbf{I}_1, \dots, \mathbf{I}_n\}$ where each column $\mathbf{I}_i$ is an observation (a $d$-dimensional image vector in this paper), for which we have already computed the singular value decomposition $A = U\Sigma V^\top$. When a $d \times m$ matrix $B$ of new observations is available, the goal is to efficiently compute the SVD of the concatenation of $A$ and $B$: $[A\ B] = U'\Sigma'V'^\top$. Letting $\tilde{B}$ be the component of $B$ orthogonal to $U$, we can express the concatenation of $A$ and $B$ in a partitioned form as follows:

$$\begin{bmatrix} A & B \end{bmatrix} = \begin{bmatrix} U & \tilde{B} \end{bmatrix} \begin{bmatrix} \Sigma & U^T B \\ 0 & \tilde{B}^\top B \end{bmatrix} \begin{bmatrix} V^\top & 0 \\ 0 & I \end{bmatrix}. \tag{1}$$

Let $R = \begin{bmatrix} \Sigma & U^T B \\ 0 & \tilde{B}^\top B \end{bmatrix}$, which is a square matrix of size $k+m$, where $k$ is the number of singular values in $\Sigma$. The SVD of $R$, $R = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$, can be computed in constant time regardless of $n$, the initial number of data. Now the SVD of $[A\ B]$ can be expressed as

$$\begin{bmatrix} A & B \end{bmatrix} = \left( \begin{bmatrix} U & \tilde{B} \end{bmatrix} \tilde{U} \right) \tilde{\Sigma} \left( \tilde{V}^\top \begin{bmatrix} V^\top & 0 \\ 0 & I \end{bmatrix} \right).$$

Since an incremental PCA is only interested in computing $U'$ and $\Sigma'$, $V'$, whose size scales with the number of observed data, need not be computed. Thus we arrive at the following formulation of the SKL algorithm.

Given $U$ and $\Sigma$ from the SVD of $A$, compute $U'$ and $\Sigma'$ from the SVD of $[A\ B]$:

1. Obtain $\tilde{B}$ and $R$ by taking the QR decomposition of $[U\Sigma\ \ B]$: $[U\ \tilde{B}]R \overset{\text{QR}}{=} [U\Sigma\ \ B]$.
2. Compute the SVD of $R$: $R \overset{\text{SVD}}{=} \tilde{U}\tilde{\Sigma}\tilde{V}^\top$.
3. Finally $U' = [U\ \tilde{B}]\tilde{U}$ and $\Sigma' = \tilde{\Sigma}$. If the desired number of basis vectors in $U'$ is less than the number of non-zero singular values, then these excess vectors and singular values may be discarded.

The algorithm can also be made slightly faster, although somewhat more complicated, by modifying the arrangement of calculations in Step 1. Instead of computing the QR decomposition of $[U\Sigma\ \ B]$, $\tilde{B}$ and $R$ can be obtained directly as follows: $\tilde{B} = \text{orth}(B - UU^\top B)$ and $R = \begin{bmatrix} \Sigma & U^\top B \\ 0 & \tilde{B}(B - UU^\top B) \end{bmatrix}$, where orth() performs orthogonalization, perhaps via QR. This reorganization, which follows from (1), avoids performing QR on the entire matrix $[U\Sigma\ \ B]$ (note that the columns corresponding to $U$ are already orthogonal), instead only orthogonalizing $(B - UU^\top B)$, which is the component of $B$ not already in the subspace $U$.

The computational advantage of the SKL algorithm over the naive approach is that it has space and time complexity that is constant in $n$, the number of training data seen so far. Specifically each update makes use of only the $k$ largest singular values and basis vectors from the previous stage. This, together with the storage required for the $m$ new images, reduces the space complexity to $O(d(k+m))$, down from $O(d(n+m)^2)$ with the naive approach. Similarly, the computational requirements are also reduced to $O(dm^2)$, versus $O(d(n+m)^2)$ for recomputing the entire SVD. More details and complexity analysis of the SKL algorithm are described in Levy and Lindenbaum (2000).

The problem with the SKL algorithm as stated above is that it makes no attempt to account for the sample mean of the training data, which changes over time as new data arrive. We will now show how this can be overcome. The essence of the approach is, at each update of the eigenbasis, to augment the new training data with an additional vector carefully chosen to correct for the time-varying mean. We begin by proving the following lemma:

**Lemma 1** *Let* $A = [\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_n]$, $B = [\mathbf{I}_{n+1}, \mathbf{I}_{n+2}, \ldots, \mathbf{I}_{n+m}]$ *be data matrices and* $C = [A\ \ B]$ *be their concatenation. Denote the means and scatter matrices of* $A$, $B$, $C$ *as* $\bar{\mathbf{I}}_A, \bar{\mathbf{I}}_B, \bar{\mathbf{I}}_C$, *and* $\mathcal{S}_A, \mathcal{S}_B, \mathcal{S}_C$ *respectively. It can be shown that* $\mathcal{S}_C = \mathcal{S}_A + \mathcal{S}_B + \frac{nm}{n+m}(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A)(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A)^\top$.

In this lemma, we define a scatter matrix to be the outer product of the centered data matrix, for example $\mathcal{S}_B = \sum_{i=n+1}^{m}(I_i - \bar{I}_B)(I_i - \bar{I}_B)^\top$. Thus a scatter matrix differs from the sample covariance matrix by only a scalar multiple $\mathcal{S}_B = m\,\text{cov}(B)$. The proof of this lemma appears in the Appendix.

From Lemma 1 we can see that the SVD of $(C - \bar{I}_C)$ is equal to the SVD of the horizontal concatenation of $(A - \bar{I}_A)$, $(B - \bar{I}_B)$, and one additional vector $\sqrt{\frac{nm}{n+m}}(\bar{I}_B - \bar{I}_A)$. (The slight abuse of notation $(A - \bar{I}_A)$ is meant as a shorthand for the matrix $[(I_1 - \bar{I}_A) \cdots (I_n - \bar{I}_A)]$.) This motivates our new algorithm, appearing in Fig. 1.

As can be seen, this algorithm shares the favorable complexity of the SKL algorithm, incurring only a small constant overhead to store, update, and correct for the changing sample mean.

### 3.1.1 Forgetting Factor

In numerous vision applications it is desirable to focus more on recently-acquired images and less on earlier observations. For example, when tracking a target with a changing appearing, it is likely that recent observations will be more indicative of its appearance than would more distant ones. Down-weighting the contribution of earlier observations also plays an important role in online learning. As time progresses the observation history can become very large, to the point of overwhelming the relative contribution of each

Given $U$ and $\Sigma$ from the SVD of $(A - \bar{I}_A)$, as well as $\bar{I}_A$, $n$, and $B$, compute $\bar{I}_C$ as well as $U'$ and $\Sigma'$ from the SVD of $(C - \bar{I}_C)$:

1. Compute the mean vectors $\bar{I}_B = \frac{1}{m}\sum_{i=n+1}^{n+m} I_i$, and $\bar{I}_C = \frac{n}{n+m}\bar{I}_A + \frac{m}{n+m}\bar{I}_B$.
2. Form the matrix $\hat{B} = \left[(I_{m+1} - \bar{I}_B)\ \ldots\ (I_{n+m} - \bar{I}_B)\ \sqrt{\frac{nm}{n+m}}(\bar{I}_B - \bar{I}_A)\right]$.
3. Compute $\tilde{B} = \text{orth}(\hat{B} - UU^\top\hat{B})$ and $R = \begin{bmatrix} \Sigma & U^\top\hat{B} \\ 0 & \tilde{B}(\hat{B} - UU^\top\hat{B}) \end{bmatrix}$.
   Note that $\tilde{B}$ will be one column larger than in the SKL algorithm.
4. Compute the SVD of $R$: $R \overset{\text{SVD}}{=} \tilde{U}\tilde{\Sigma}\tilde{V}^\top$.
5. Finally $U' = [U\ \tilde{B}]\tilde{U}$ and $\Sigma' = \tilde{\Sigma}$.

**Fig. 1** The incremental PCA algorithm with mean update

block of new data, rendering the learner 'blind' to changes in the observation stream.

One way to moderate the balance between old and new observations is to incorporate a *forgetting factor* in the incremental eigenbasis update, as suggested by Levy and Lindenbaum (2000). To do this, at each update the previous singular values are multiplied by a scalar factor $f \in [0, 1]$, where $f = 1$ indicates no forgetting is to occur. Thus at Step 3 in Fig. 1, $R = \left[\begin{smallmatrix} f\Sigma & U^\top \hat{B} \\ 0 & \tilde{B}(\hat{B} - UU^\top \hat{B}) \end{smallmatrix}\right]$, which is equivalent to taking the QR decomposition of $[fU\Sigma \quad \hat{B}]$ instead of $[U\Sigma \quad \hat{B}]$.

Although they propose the use of a forgetting factor, Levy and Lindenbaum do not provide any analysis as to its effect on the resulting eigenbasis. We address this with the following lemma:

**Lemma 2** *A forgetting factor of $f$ reduces the contribution of each block of data to the overall covariance modeled by an additional factor of $f^2$ at each SVD update.*

Hence, after the $k$th update of the eigenbasis, the block of $m$ observations added during the $j$th update ($j < k$) will have its covariance down-weighted by a factor of $f^{2(k-j)}$. The proof of Lemma 2 appears in the Appendix. The objective of PCA is to locate a subspace of dimension $k$ that retains as much of the data covariance as possible (i.e., maximizes the determinant of the projected data covariance $|U^\top \mathrm{cov}(\mathrm{Data}) U|$, Jolliffe 2002). Therefore it is a 'win' for PCA to select as basis vectors directions of large covariance in recent data, at the expense of directions favored only by earlier data.

An important consideration not previously addressed is the affect of the forgetting factor on the mean of the eigenbasis. Since the contribution of the previously-observed data to the covariance is decreased, it is necessary to also reduce its contribution to the resulting mean. When a forgetting factor of $f$ is used, we propose the following modification to the mean update (Step 1 in Fig. 1):

$$\bar{I}_C = \frac{fn}{fn + m}\bar{I}_A + \frac{m}{fn + m}\bar{I}_B$$

and at each update to compute the *effective* size of the observation history as $n \leftarrow fn + m$.

A benefit of incorporating the forgetting factor into the mean update is that the mean can still change in response to new observations, even as the actual number of observations approaches infinity. Specifically, using $n \leftarrow fn + m$, the effective number of observations will reach equilibrium at $n = fn + m$, or $n = m/(1 - f)$. For instance, when $f = 0.95$ and $m = 5$ new observations are included at each update, the effective size of the observation history will approach $n = 100$.

### 3.2 Sequential Inference Model

The visual tracking problem is cast as an inference task in a Markov model with hidden state variables. The state variable $\mathbf{X}_t$ describes the affine motion parameters (and thereby the location) of the target at time $t$. Given a set of observed images $\mathcal{I}_t = \{\mathbf{I}_1, \ldots, \mathbf{I}_t\}$, we aim to estimate the value of the hidden state variable $\mathbf{X}_t$. Using Bayes' theorem, we have the familiar result

$$p(\mathbf{X}_t|\mathcal{I}_t) \propto p(\mathbf{I}_t|\mathbf{X}_t) \int p(\mathbf{X}_t|\mathbf{X}_{t-1}) p(\mathbf{X}_{t-1}|\mathcal{I}_{t-1})\, d\mathbf{X}_{t-1}. \tag{2}$$

The tracking process is governed by the observation model $p(\mathbf{I}_t|\mathbf{X}_t)$, where we estimate the likelihood of $\mathbf{X}_t$ observing $\mathbf{I}_t$, and the dynamical model between two states $p(\mathbf{X}_t|\mathbf{X}_{t-1})$. The Condensation algorithm (Isard and Blake 1996), based on factored sampling, approximates an arbitrary distribution of observations with a stochastically generated set of weighted samples. We use a variant of the Condensation algorithm to model the distribution over the object's location, as it evolves over time.

#### *3.2.1 Dynamical Model*

The location of a target object in an image frame can be represented by an affine image warp. This warp transforms the image coordinate system, centering the target within a canonical box such as the unit square, as illustrated in Fig. 2. In this work the state at time $t$ consists of the six parameters of an affine transformation $\mathbf{X}_t = (x_t, y_t, \theta_t, s_t, \alpha_t, \phi_t)$ where $x_t$, $y_t$, $\theta_t$, $s_t$, $\alpha_t$, $\phi_t$, denote $x$, $y$ translation, rotation angle, scale, aspect ratio, and skew direction at time $t$.

To develop a tracker for generic applications, the dynamics between states in this space is modeled by Brownian motion. Each parameter in $X_t$ is modeled independently by a Gaussian distribution around its counterpart in $X_{t-1}$, and thus the motion between frames is itself an affine transformation. Specifically,

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}_t; \mathbf{X}_{t-1}, \Psi), \tag{3}$$

where $\Psi$ is a diagonal covariance matrix whose elements are the corresponding variances of affine parameters, i.e., $\sigma_x^2$, $\sigma_y^2$, $\sigma_\theta^2$, $\sigma_s^2$, $\sigma_\alpha^2$, $\sigma_\phi^2$. These parameters dictate the kind of motion of interest to a tracker and this generic dynamical model assumes the variance of each affine parameter does not change over time. More the complex dynamics can be modeled, such as first or second order dynamic systems, as well as other adaptive techniques for specific applications (North and Blake 1998). Like all the other applications using particle filters, there is a trade off between the number of particles needed to be drawn (i.e., efficiency) and how well

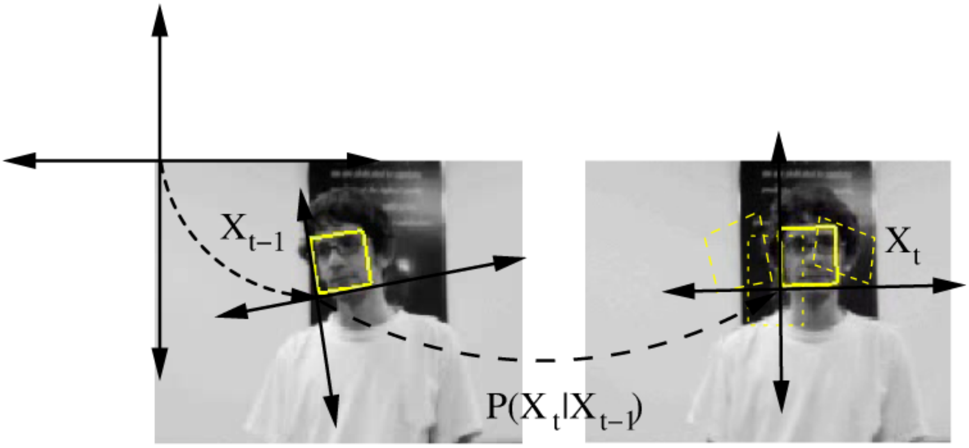

**Fig. 2** The model of dynamics. A location is represented by an affine transformation (e.g., $\mathbf{X}_{t-1}$), which warps the coordinate system so that the target lies within the unit square. Particles representing possible target locations $\mathbf{X}_t$ at time $t$ are sampled according to $P(\mathbf{X}_t|\mathbf{X}_{t-1})$, which in this case is a diagonal-covariance Gaussian centered at $\mathbf{X}_{t-1}$

particle filters approximate the posterior distribution (i.e., effectiveness). With larger values in the diagonal covariance matrix $\boldsymbol{\Psi}$ and more particles, it is possible to track the object with higher precision at the cost of increased computation. In this paper, we find a balance between these factors for efficient and effective visual tracking.

### 3.2.2 Observation Model

Since our goal is to use a representation to describe the "thing" that we are tracking, we model image observations using a probabilistic interpretation of principal component analysis (Tipping and Bishop 1999). Given an image patch $\mathbf{I}_t$ predicated by $\mathbf{X}_t$, we assume $\mathbf{I}_t$ was generated from a subspace of the target object spanned by $U$ and centered at $\boldsymbol{\mu}$. The probability of a sample being generated from this subspace is inversely proportional to the distance $d$ from the sample to the reference point (i.e., $\boldsymbol{\mu}$) of the subspace, which can be decomposed into the distance-to-subspace, $d_t$, and the distance-within-subspace from the projected sample to the subspace center, $d_w$. This distance formulation, based on a orthonormal subspace and its complement space, is similar to (Moghaddam and Pentland 1995) in spirit.

The probability of a sample generated from a subspace, $p_{d_t}(\mathbf{I}_t \mid \mathbf{X}_t)$, is governed by a Gaussian distribution:

$$p_{d_t}(\mathbf{I}_t \mid \mathbf{X}_t) = \mathcal{N}(\mathbf{I}_t; \boldsymbol{\mu}, UU^\top + \varepsilon I), \tag{4}$$

where $I$ is an identity matrix, $\boldsymbol{\mu}$ is the mean, and $\varepsilon I$ term corresponds to the additive Gaussian noise in the observation process. It can be shown (Roweis 1997) that the negative exponential distance from $\mathbf{I}_t$ to the subspace spanned by $U$, i.e., $\exp(-\|(\mathbf{I}_t - \boldsymbol{\mu}) - UU^\top(\mathbf{I}_t - \boldsymbol{\mu})\|^2)$, is proportional to $\mathcal{N}(\mathbf{I}_t; \boldsymbol{\mu}, UU^\top + \varepsilon I)$ as $\varepsilon \rightarrow 0$.

Within a subspace, the likelihood of the projected sample can be modeled by the Mahalanobis distance from the mean as follows:

$$p_{d_w}(\mathbf{I}_t \mid \mathbf{X}_t) = \mathcal{N}(\mathbf{I}_t; \boldsymbol{\mu}, U\Sigma^{-2}U^\top), \tag{5}$$

where $\boldsymbol{\mu}$ is the center of the subspace and $\Sigma$ is the matrix of singular values corresponding to the columns of $U$. Put together, the likelihood of a sample being generated from the subspace is governed by

$$\begin{aligned} p(\mathbf{I}_t|\mathbf{X}_t) &= p_{d_t}(\mathbf{I}_t|\mathbf{X}_t)p_{d_w}(\mathbf{I}_t|\mathbf{X}_t) \\ &= \mathcal{N}(\mathbf{I}_t; \boldsymbol{\mu}, UU^\top + \varepsilon I)\mathcal{N}(\mathbf{I}_t; \boldsymbol{\mu}, U\Sigma^{-2}U^\top). \end{aligned} \tag{6}$$

Given a drawn particle $\mathbf{X}_t$ and the corresponding image patch $\mathbf{I}_t$, we aim to compute $p(\mathbf{I}_t|\mathbf{X}_t)$ using (6). To minimize the effect of noisy pixels, we utilize a robust error norm (Black and Jepson 1996), $\rho(x, \sigma) = \frac{x^2}{\sigma^2+x^2}$ instead of the Euclidean norm $\|x\|^2$, to ignore the outliers (i.e., the pixels that are not likely to appear inside the target region given the current eigenbasis). We use a method similar to that used in Black and Jepson (1996) in order to compute $d_t$ and $d_w$. This robust error norm is helpful when we use a rectangular region to enclose the target object (which inevitably contains some noisy background pixels).

### 3.2.3 Distance Metric

The two distances, $d_t$ and $d_w$, have a probabilistic interpretation within a latent variable model, and care should be taken to leverage their contributions within the observation model specified in (6). Denote $x \in \mathcal{R}^d$ as a high dimensional data sample (i.e., image observation $\mathbf{I}_t$ at time $t$ discussed in the previous section) and $z \in \mathcal{R}^q$ as the corresponding low-dimensional latent variable. We can define a latent model with Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2 I_d)$ as:

$$x = Wz + \mu + \epsilon, \tag{7}$$

where $W \in \mathcal{R}^{d \times q}$ is an orthogonal matrix, $W^T W = I_q$, and $z$ is a zero mean Gaussian $z \sim \mathcal{N}(0, L)$ with $L$ being a diagonal matrix. From (7), we have

$$p(x|z) \sim \mathcal{N}(Wz + \mu, \sigma^2 I_d). \tag{8}$$

1. Locate the target object in the first frame, either manually or by using an automated detector, and use a single particle to indicate this location.
2. Initialize the eigenbasis $U$ to be empty, and the mean $\mu$ to be the appearance of the target in the first frame. The effective number of observations so far is $n = 1$.
3. Advance to the next frame. Draw particles from the particle filter, according to the dynamical model.
4. For each particle, extract the corresponding window from the current frame, and calculate its weight, which is its likelihood under the observation model.
5. Store the image window corresponding to the most likely particle. When the desired number of new images have been accumulated, perform an incremental update (with a forgetting factor) of the eigenbasis, mean, and effective number of observations. In our experiments, the update is performed every fifth frame.
6. Go to step 3.

**Fig. 3** A summary of the proposed tracking algorithm

In addition, we can compute the likelihood that observed sample $x$ is generated from the model, $p(x)$:

$$p(x) = \int p(x|z)p(z)dz \sim \mathcal{N}(\mu, WLW^T + \sigma^2 I_d). \tag{9}$$

By taking the log of $p(x)$, we get

$$\log p(x) = -\frac{1}{2}\{\log(2\pi)^d + \log|WLW^T + \sigma^2 I_d| + (x-\mu)^T(WLW^T + \sigma^2 I_d)^{-1}(x-\mu)\}. \tag{10}$$

When parameters $W$, $L$ and $\sigma^2$ are fixed. $\log p(x)$ is determined by $(x-\mu)^T(WLW^T + \sigma^2 I_d)^{-1}(x-\mu)$. According to the Sherman–Morrison–Woodbury formula

$$(WLW^T + \sigma^2 I_d)^{-1} = \frac{1}{\sigma^2}I_d - \frac{1}{\sigma^2}W\left(L^{-1} + \frac{1}{\sigma^2}I_q\right)^{-1}W^T\frac{1}{\sigma^2} \tag{11}$$

and since

$$\left(L^{-1} + \frac{1}{\sigma^2}I_q\right)^{-1} = \sigma^2(I_q - D^{-1}), \tag{12}$$

where $D$ is a diagonal matrix with $D_{ii} = L_{ii} + \sigma^2$, we can get:

$$(WLW^T + \sigma^2 I_d)^{-1} = \frac{1}{\sigma^2}(I_d - WW^T) + WD^{-1}W^T. \tag{13}$$

In the probabilistic PCA model, $W$ and $D$ correspond to the eigenvectors and the diagonal matrix of eigenvalues of the sample covariance matrix. Plugging (13) into (10), it is clear that $\log p(x)$ is determined by two terms: the Euclidean distance to the subspace, $(x-\mu)^T(I_d - WW^T)(x-\mu)$ (i.e., $d_t$ in our formulation) weighted by $\frac{1}{\sigma^2}$, and the Mahalanobis distance within the subspace, $(x-\mu)^T WD^{-1}W^T(x-\mu)$ (i.e., $d_w$ in our model).

In (13) $d_t$ is scaled by $\sigma^2$ in computing $\log p(x)$. As the observation noise $\sigma^2$ increases, the contribution of $d_t$ is reduced. On the other hand, the contribution of $d_w$ decreases if $D$ is increased when $L$ in (7) is increased. When $D$ goes to infinity, the contribution of $d_w$ becomes null and $\log p(x)$ is solely determined by $d_t$. In this case, the Gaussian model of latent variable $z$ has infinite variance, which is equivalent to the case that $z$ is uniformly distributed.

In summary, if we only use $d_t$ as a measurement of $p(x)$, effectively we assume latent variable $z$ is uniformly distributed in the subspace. If we use both $d_t$ and $d_w$ in computing $p(x)$, then we assume that the distribution of $z$ in the subspace is Gaussian as it covers only a local region in the subspace. In this work, we take the later view.

### 3.3 Summary of the Tracking Algorithm

We now provide a summary of the proposed tracking algorithm in Fig. 3.

At the very beginning when the eigenbasis is empty (i.e., before the first update), our tracker works as a template based tracker. There is a natural tradeoff between update frequency and speed of movement. Likewise, there is a tradeoff between the number of particles and granularity of movement. We will discuss these implementation issues in the next section.

## 4 Implementation and Experiments

To evaluate empirical performance of the proposed tracker, we collected a number of videos recorded in indoor and outdoor environments where the targets change pose in different lighting conditions. Each video consists of 320 × 240-pixel grayscale images recorded at 30 frames per second, unless specified otherwise. Note that there exists large and unpredictable camera motion in the videos. For the eigenbasis representation, each target image region is resized to a 32 × 32 patch, and the number of eigenvectors used in all experiments is set to 16, though fewer eigenvectors can also work well. The forgetting term is empirically set to be 0.95, and the batch size for the eigenbasis update is set to 5

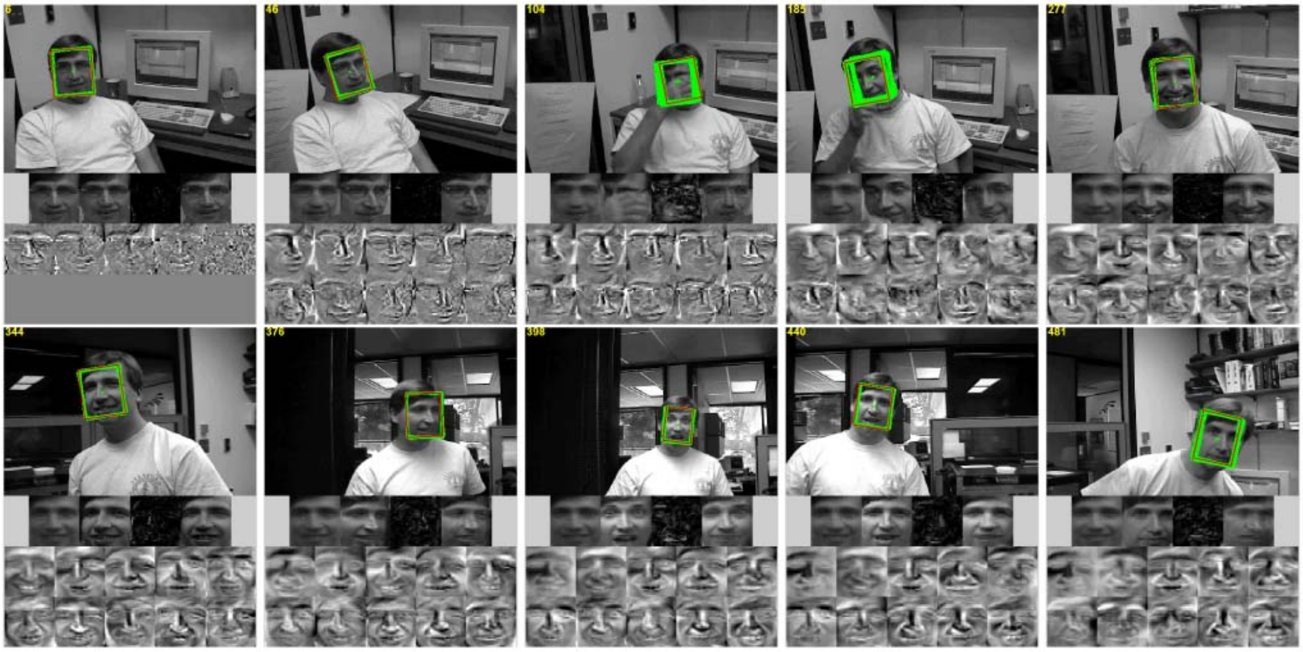

**Fig. 4** (Colour online) A person undergoing large pose, expression, appearance, and lighting change, as well as partial occlusions. The *red window* shows the maximum a posteriori estimate of the particle filter, and the *green windows* show the other particles with weight above a threshold. The results can be better viewed on high resolution displays or color printouts. More results are shown in the accompanying video

as a trade-off between computational efficiency and effectiveness of modeling appearance change during fast motion. Implemented in MATLAB with MEX, our algorithm runs at 7.5 frames per second with 600 particles on a standard 2.8 GHz computer. Here we present selected tracking results, with more tracking results as well as videos available at http://www.cs.toronto.edu/~dross/ivt/. Note that the results can be better viewed on high resolution displays or color printouts. Sample code and data sets are also available at the aforementioned website.

We begin by showing the results of our tracker on several sequences, then compare it qualitatively to two other state-of-the-art trackers. Next we evaluate and compare the trackers' quantitative performance, and empirically demonstrate the accuracy of our incremental PCA algorithm. We conclude with a discussion of the experimental results.

### 4.1 Experimental Results

We first tested our algorithm using a challenging video studied in Jepson et al. (2003). The image sequence was downsampled by one-half, retaining only every other frame. Figure 4 shows the empirical results using our proposed method, where the first row of each panel shows the tracked objects (enclosed with rectangles) and the second row shows (from left to right) the subspace center, tracked image patch, residue, and reconstructed image using current eigenbasis. The red window shows the maximum a posteriori estimate of the particle filter, and green windows show the other particles whose weights are above a threshold. The eigenbasis images of the current subspace are shown in the third row of each panel (sorted according to their eigenvalues). Note that our method is able to track the target undergoing pose (#46, #185, #344, #376, #481), expression (#277, #398), and lighting (#344, #440) variation. Further, our method is able to track the target with temporary occlusion (#104) and structured appearance change such as glasses (#6, #185). Compared with the results reported in Jepson et al. (2003), our method is able to efficiently learn a compact representation while tracking the target object without using wavelets. All the eigenbases are constructed automatically from scratch and constantly updated to model the appearance of the target object, while it undergoes intrinsic and extrinsic changes. The eigenbases capture the appearance details of the target in different pose, expression, and with or without glasses.

The second image sequence, shown in Fig. 5, contains an animal doll moving in different pose, scale, and lighting conditions. Once initialized in the first frame, our algorithm is able to track the target object as it experiences large pose change (#65, #272, #364, #521, #550, #609), a cluttered background (#65, #450, #521, #609), scale change (#65, #225), and lighting variation (#225, #364, #450, #609). Notice that the non-convex target object is localized within a rectangular window, and thus it inevitably contains some background pixels in its appearance representation. The robust error norm enables the tracker to ignore background

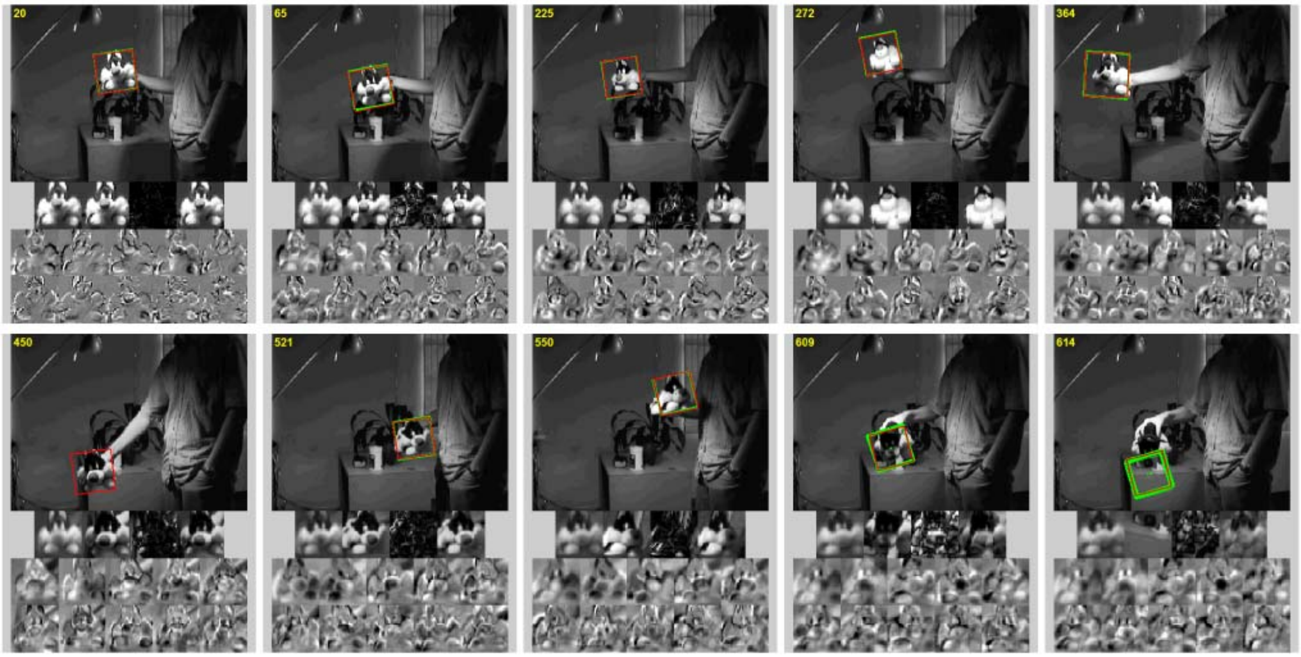

**Fig. 5** (Colour online) An animal doll moving with significant pose, lighting and scale variation in a cluttered scene. The results can be better viewed on high resolution displays or color printouts

pixels and estimate the target location correctly. The results also show that our algorithm faithfully models the appearance of an arbitrary object, as shown in eigenbases and reconstructed images, in the presence of noisy background pixels. Nevertheless, our tracker eventually fails after frame 614 as a result of a combination of drastic pose and illumination change. Since the proposed algorithm is not limited to rectangular patches when specifying a region of interest for tracking, better results can be expected with more compact enclosing windows for specific targets.

Figure 6 shows the tracking results using a challenging sequence, recorded at 15 frames per second with a moving digital camera, in which a person moves from a dark room toward a bright area while changing his pose, moving underneath spotlights, changing facial expressions and taking off his glasses. Notice that there is also a large scale variation in the target relative to the camera. Even with the significant camera motion and low frame rate (which makes the motions between frames more significant, as when tracking fast-moving objects), our algorithm is able to track the target throughout the sequence, experiencing only temporary drifts. In contrast, most gradient or contour based trackers are not expected to perform well due to the large lighting variation, cast shadows, and unknown camera motion. With the use of a particle filter, our tracker is able to recover from temporary drifts due to a sudden and large pose change (between frames #166 and #167 in the accompanying video). Furthermore, the eigenbasis is constructed from scratch and is updated to reflect the appearance variation of the target object.

We recorded a sequence to evaluate our tracker in outdoor environments, where lighting conditions often change drastically. In it, a person walks underneath a trellis covered by vines, resulting in a significant variation in appearance due to cast shadows. As shown in Fig. 7, the cast shadows change the appearance of the target face significantly (#96, #155, #170, #180, #223, #278). Furthermore, the pose and lighting variation combined with a low frame rate (15 fps) makes the tracking task rather challenging (#198, #303). Nevertheless, our algorithm is able track the target fairly accurately and robustly. In this sequence the forgetting term plays an important role, down-weighting the previously seen face images and focusing on the most recent ones, as the object appearance changes drastically.

Figure 8 shows the results of tracking a moving vehicle, as it passes beneath a bridge and under trees. Although there is a sudden illumination change (#12, #184, #192, #232) in the scene, our tracker is able to track the target well. Our algorithm is also able to track objects in low resolution images, such as the sequence of a vehicle driving at night, shown in Fig. 9. Despite the small size of the target relative to the image, and the difficult illumination conditions, our algorithm is able to track the vehicle well.

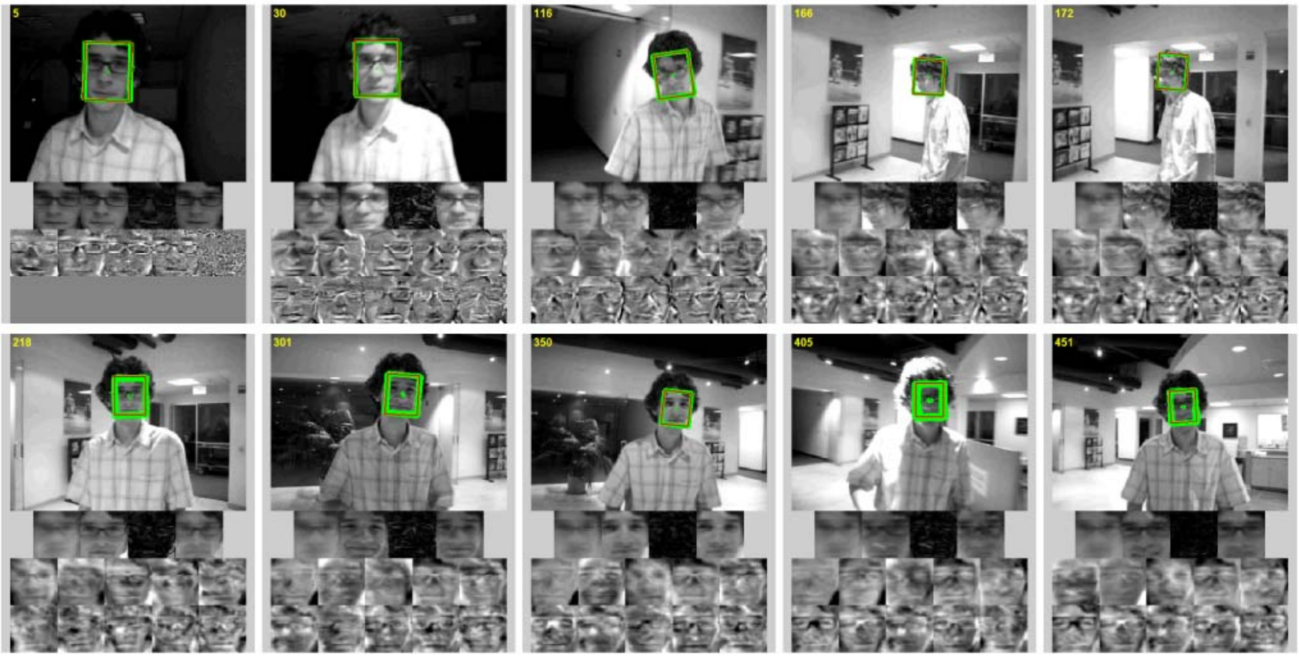

**Fig. 6** A person moves from a dark to a bright area, undergoing large lighting and pose changes. The images in the *second row* show the current sample mean, tracked region, reconstructed image, and the reconstruction error respectively. The *third and fourth rows* show the top 10 principal eigenvectors

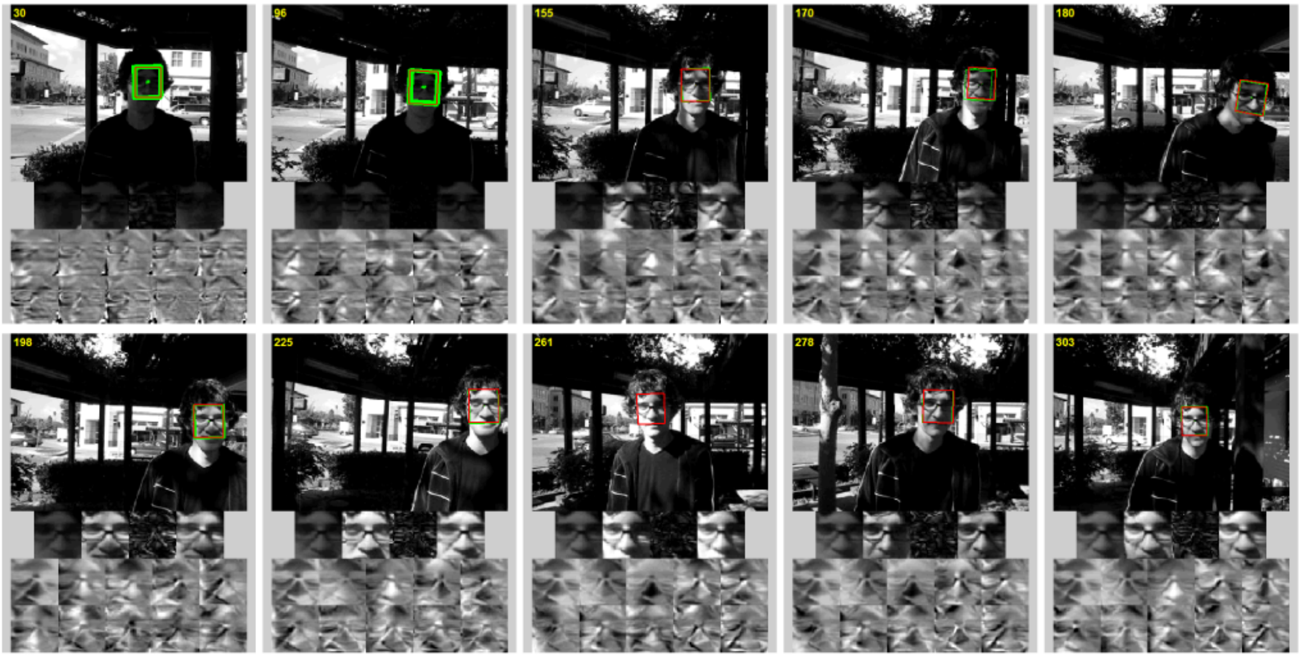

**Fig. 7** A person moves underneath a trellis with large illumination change and cast shadows while changing his pose. More results can be found in the project web page

### 4.2 Qualitative Comparison

As a qualitative benchmark, we ran two state-of-the-art algorithms, the WSL (Jepson et al. 2003) and Mean Shift (Comaniciu et al. 2003) trackers, on four of the sequences. The results are depicted in Fig. 10. As can be seen in the figure (and corresponding videos), our method provides comparable performance to the WSL tracker. In the case of the first "Dudek" sequence, it is able to do so despite using a tighter target window around the face. However in

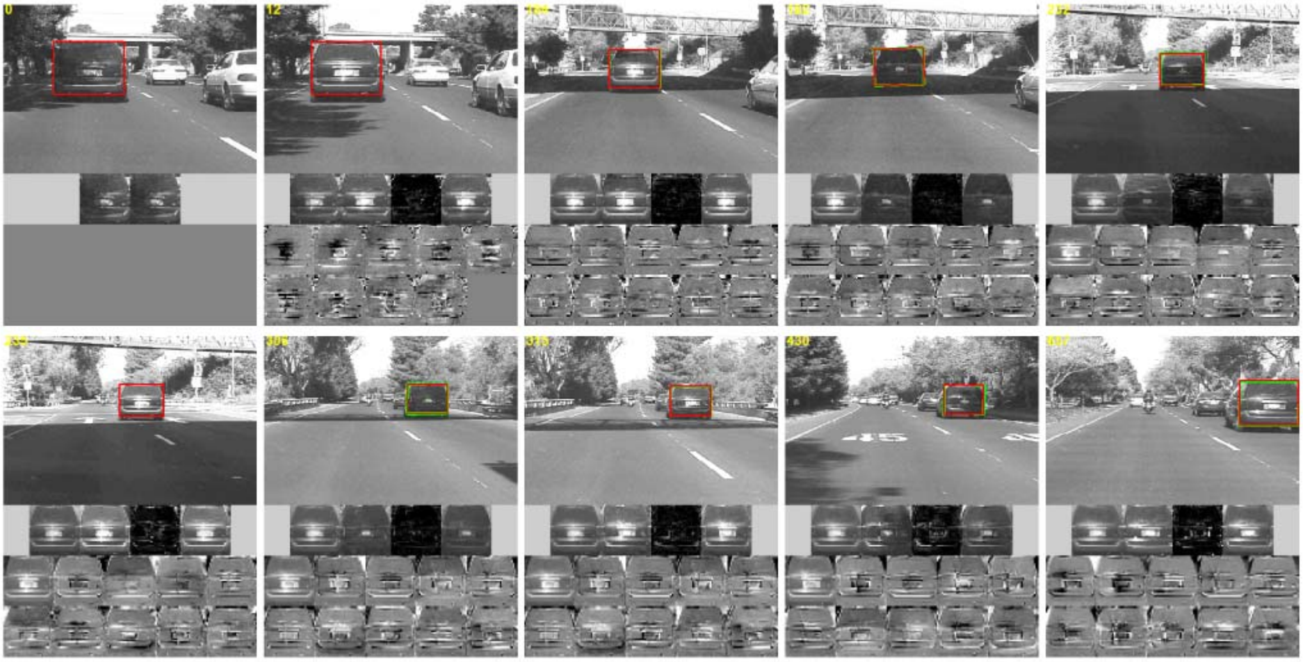

**Fig. 8** A vehicle moving underneath an overpass and trees. Our algorithm is able to track the target despite the large illumination variation

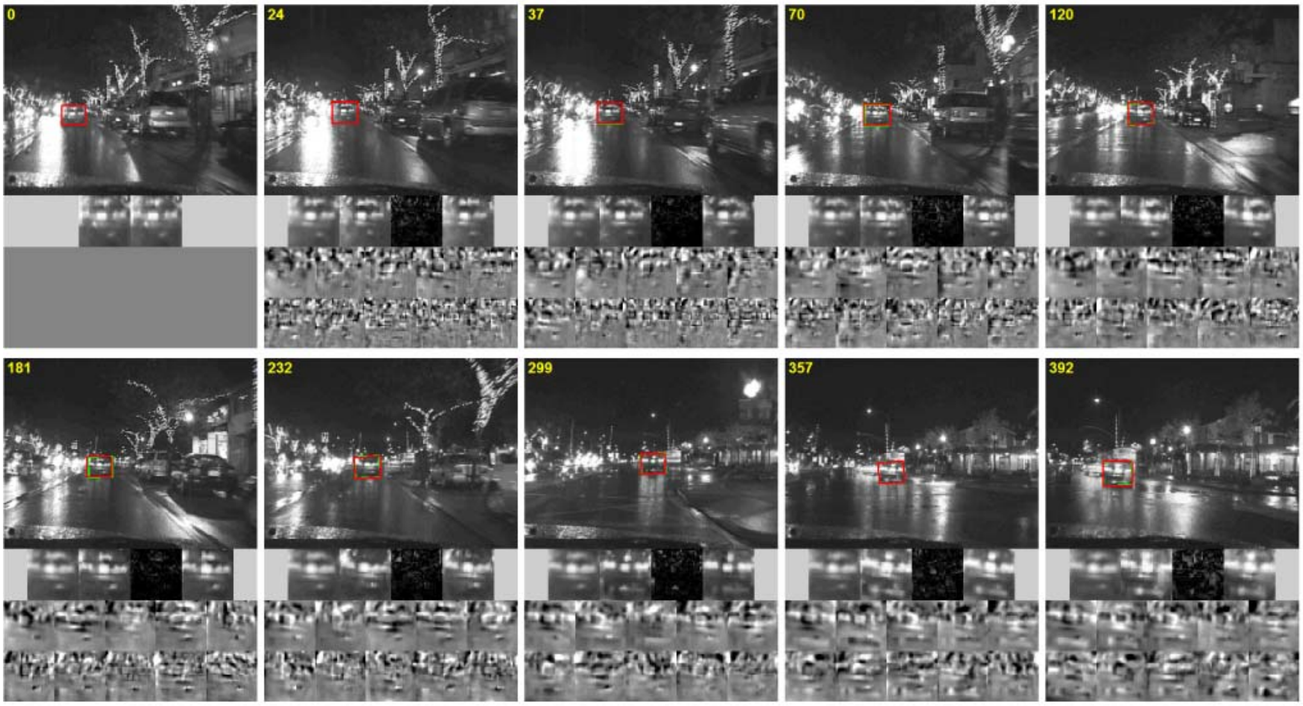

**Fig. 9** A vehicle moving in the night time with large illumination changes. Our algorithm is able to track the target when the images have low resolution and contrast

the "animal doll" and "trellis", sequences, the WSL tracker proves to be more robust, continuing to track after our method fails. In both cases the targets experience drastic non-uniform changes in illumination from directed light sources. The WSL tracker gains a distinct advantage in these cases, based on its use of wavelet phase features as an input representation. (It is possible that the performance of our tracker could also be improved by using a wavelet representation.) Further, our method not only tracks a target object, it also learns a compact low-dimensional representation

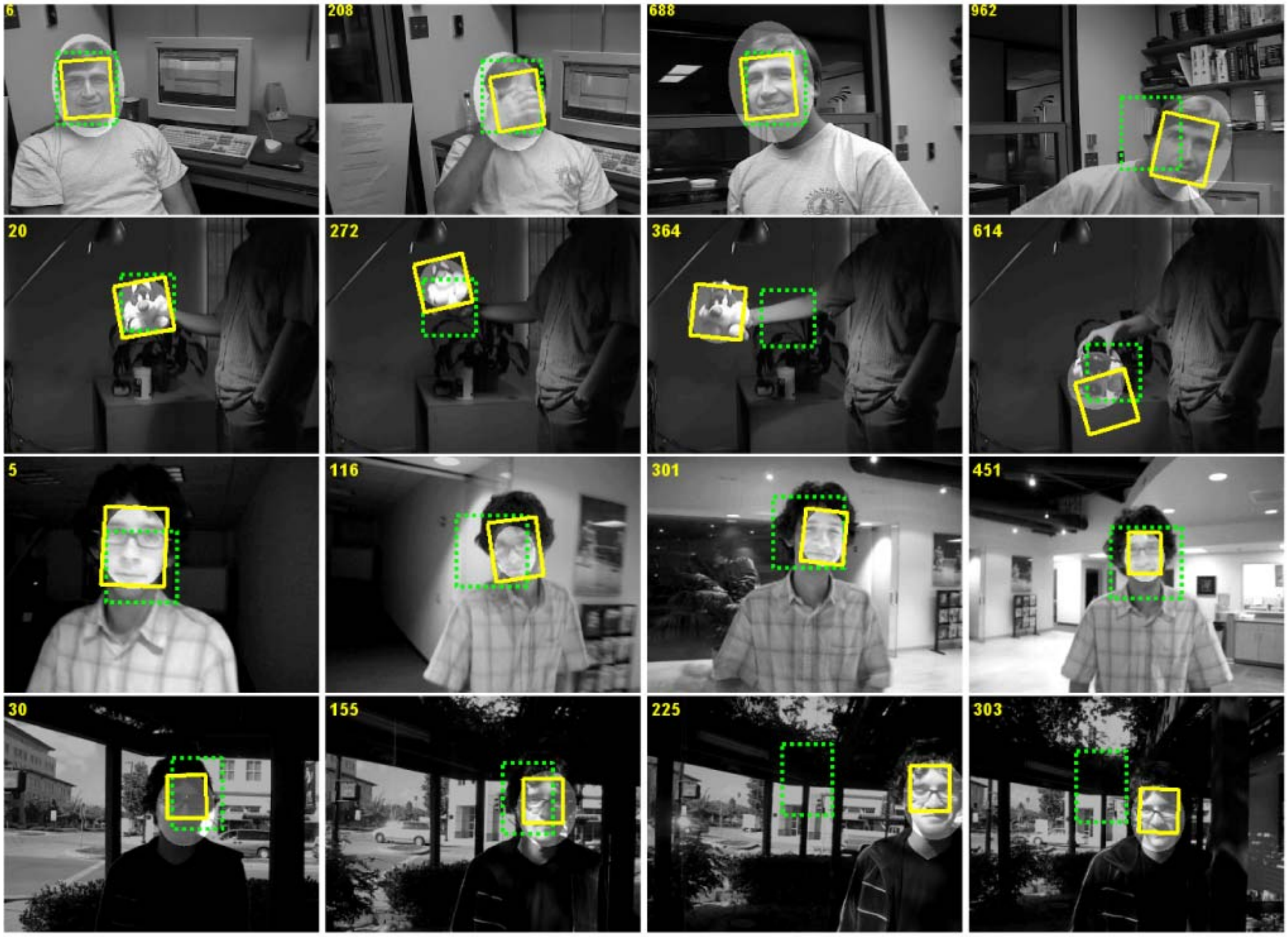

**Fig. 10** (Colour online) A comparison of our tracker (indicated with a *yellow box*) with the WSL (Jepson et al. 2003) (shown in *highlighted ellipse*) and the Mean Shift (Comaniciu et al. 2003) (depicted by a *green dashed box*) on four video sequences

which can be used for other applications such as recognition.

On the other hand, the Mean Shift tracker performs poorly, experiencing significant drift off the target objects. This can be attributed to the appearance model of the Mean Shift tracker, based on histograms of pixel intensities, which does not adapt over time, and is not sufficiently discriminative on these grayscale-only sequences. We expect that variants of the Mean Shift tracker using more sophisticated representations or adaptive multi-component models (Georgescu et al. 2004), would show improved performance.

### 4.3 Quantitative Analysis

To evaluate the tracking precision quantitatively, we tested the ability of our algorithm to consistently track seven facial feature points in the "Dudek" sequence. We compare our results with the manually-labeled "ground truth" locations of the features, as initially presented in Jepson et al. (2003).

To obtain estimates of the feature locations, we began by tracking the face, obtaining a sequence of similarity transformations approximately describing its motion from one frame to the next. For this image sequence, we used slightly larger variances, more particles (4000), and a forgetting factor of 0.99. Given the locations of the facial features in the first frame, we applied the sequence of transformations to these points, obtaining at each frame an estimate of where the features lay. Representative tracking results are shown in Fig. 11, with red x's used to indicate our estimates of the feature locations, and yellow x's for the ground-truth positions.

Finally we computed the root mean square (RMS) error between the estimated locations of the features and the ground-truth. The error is plotted for each frame in Fig. 12. For most frames our tracking results match the ground truth well, the largest errors occurring during brief occlusions or fast pose changes. The average RMS error of our method is 5.07 pixels per feature per frame, which is slightly better than the error of 5.2 pixels reported for the WSL tracker in

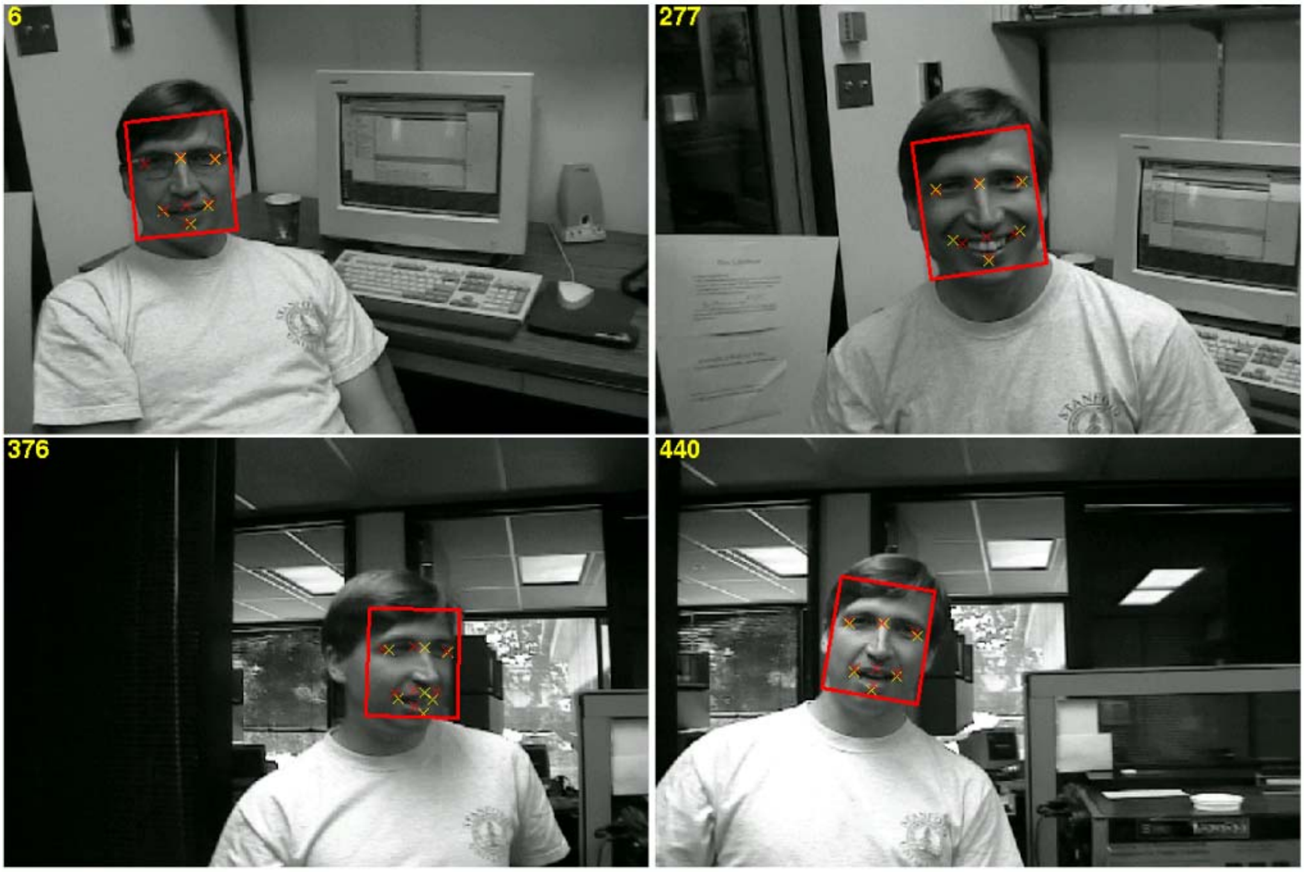

**Fig. 11** (Colour online) A person undergoing large pose, expression, appearance, and lighting change, as well as partial occlusions. The *yellow crosses* denote the ground truth data and the *red crosses* represent our tracking results. The differences can be best viewed when the images are magnified on displays

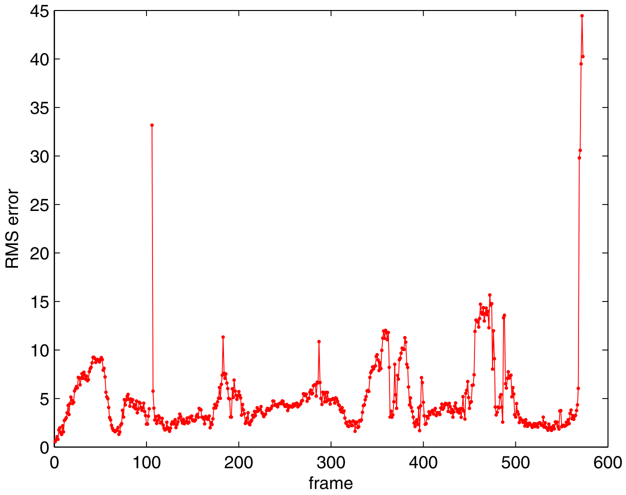

**Fig. 12** The RMS error at tracking feature points, for each frame in the "Dudek" sequence. The abrupt increases in error occur when there is temporary occlusion or motion blur

Jepson et al. (2003). In contrast, the Mean Shift tracker described in Sect. 4.2 has an average error of 48.7 pixels. Note that the errors in most frames are rather small and the errors in a few frames contribute most to the average RMS error.

Comparing the ability to track labeled features also allows us to quantitatively assess the contribution of the correct mean update and forgetting factor in our incremental algorithm to overall tracking performance. First, we re-ran the tracker without incrementally adapting the mean of the eigenbasis. The resulting average error increased to 5.86 pixels per feature per frame. Next, we removed the forgetting factor from the algorithm (while using the correct mean update) and re-ran the tracker. This caused an even larger increase in error, to 7.70 pixels. Substituting our incremental algorithm with that of Hall et al. (2002), which lacks the forgetting factor, also produced an error of 7.70. These results demonstrate that the mean update and, particularly, the forgetting factor provide a measurable boost to tracking performance.

To demonstrate the effectiveness of the proposed eigenbasis update algorithm in modeling object appearance, we compare the reconstruction results of our method to the conventional PCA algorithm, and to the incremental algorithm of Hall et al. (2002). For a fair comparison we do not use the

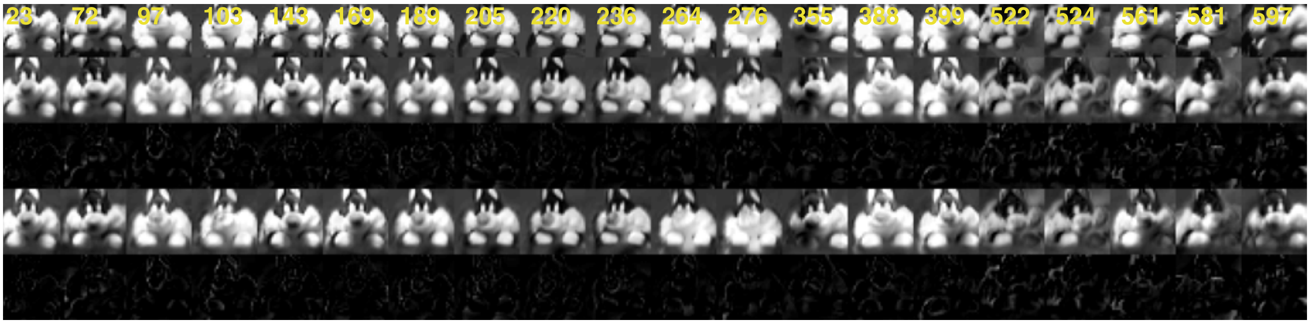

**Fig. 13** The *first row* shows a selection of test images. The *second and fourth rows* show the reconstructions of these images using the conventional batch algorithm and our incremental algorithm, respectively. Their corresponding residues are presented in the *third and fifth rows*

forgetting factor in this experiment, so that the reconstruction error of each input image is treated equally by all algorithms. Unlike conventional PCA, which constructs a subspace using all the frames in the video (i.e., batch mode), the incremental algorithms—Hall's and our own—update the subspace periodically as frames arrive. For this experiment, as with the tracker, we update the basis every five frames. At any given time the incremental algorithms retain only the top few eigenvectors, thereby providing an efficient method with a compact representation.

We used the "animal doll" sequence for experiments, extracting images of the target object from the first 605 frames of the sequence to use as training data. A selection of these images are depicted in the first row of Fig. 13. The conventional batch PCA algorithm, our algorithm, and that of Hall et al. were used to construct bases consisting of 16 top eigenvectors. For both incremental algorithms this entailed 121 incremental updates, retaining only the top 16 eigenvectors after each update. When the learned bases were used to reconstruct the training images, batch PCA incurred a RMS reconstruction error of $7.93 \times 10^{-2}$ per pixel, whereas the error of our algorithm was only slightly higher, at $8.03 \times 10^{-2}$. The reconstructed images using the batch PCA algorithm and our algorithm are shown in rows 2 and 4 of Fig. 13 respectively, and rows 3 and 5 contain the corresponding residue images.

In comparison, Hall's algorithm achieved the same reconstruction error as our own, $8.03 \times 10^{-2}$, however its runtime, averaged over 100 repetitions, was 38% greater than that of our algorithm. (The results of Hall's algorithm are not included in the figure since, when the forgetting factor is not used, they are visually indistinguishable from our own.) The batch PCA algorithm takes on average 6 times longer than our incremental algorithm, even after we rearrange the computation to calculate the eigenvectors of the Gram matrix ($X^T X$) rather than the covariance matrix ($X X^T$), as described in Murase and Nayar (1995).

Thus, from these experiments we can conclude that our incremental eigenbasis update method is able to effectively model the object appearance without losing detailed information, at a cost appreciably less than that of Hall et al.'s algorithm.

## 4.4 Discussion

The robust tracking performance of our algorithm can be attributed to several factors. One reason is that our incremental eigenbasis learning approach exploits the local linearity of appearance manifold for matching targets in consecutive frames. It is well known that the appearance of an object undergoing pose change can be modeled well with a view-based representation (Murase and Nayar 1995). Meanwhile at fixed pose, the appearance of an object in different illumination conditions can be approximated well by a low dimensional subspace (Belhumeur and Kreigman 1997). Our empirical results show that these variations can be learned online without any prior training phase, and also the changes caused by cast and attached shadows can still be approximated by a linear subspace to certain extent. Consequently the appearance of an object undergoing illumination and pose variation can be approximated by a local subspace within a short span of time, which in turns facilitates the tracking task. Notice that at any time instant, it suffices to use an eigenbasis to account for appearance variation if the object motion or illumination change is not instantly drastic. This work demonstrates that a tracker based on the idea of an incremental eigenbasis update can be both efficient and perform well empirically when the appearance change is gradual. A few additional failure cases for this algorithm can be seen at project the web site, mentioned earlier. Typically, failures happen when there is a combination of fast pose change and drastic illumination change.

In this paper, we do not directly address the partial occlusion problem. Empirical results show that temporary and partial occlusions can be handled by our method through constant update of the eigenbasis and the robust error norm. Nevertheless situations arise where we may have prior knowledge of the object being tracked, and can exploit such information for better occlusion handling.

## 5 Conclusions and Future Work

We have presented an appearance-based tracker that incrementally learns a low dimensional eigenbasis representation for robust object tracking while the target undergo pose, illumination and appearance changes. Whereas most algorithms operate on the premise that the object appearance or ambient environment lighting conditions do not change as time progresses, our method adapts the model representation to reflect appearance variation of the target, thereby facilitating the tracking task. In contrast to the existing incremental subspace methods, our eigenbasis update method updates the mean and eigenbasis accurately and efficiently, and thereby learns to faithfully model the appearance of the target being tracked. Our experiments demonstrate the effectiveness of the proposed tracker in indoor and outdoor environments where the target objects undergo large pose and lighting changes.

Although our tracker performs well, it occasionally drifts from the target object. With the help of particle filters, the tracker often recovers from drifts in the next few frames when a new set of samples is drawn. For specific applications, better mechanisms to handle drifts could further enhance robustness of the proposed algorithm. The current dynamical model in our sampling method is based on a Gaussian distribution, but for certain specific applications the dynamics could be learned from exemplars for more efficient parameter estimation. Our algorithm can also be extended to construct a set of eigenbases for modeling nonlinear aspects of appearance variation more precisely and automatically. We aim to address these issues in our future work.

**Acknowledgements** We thank the reviewers and editors for their comments and suggestions. We also thank Thomas El-Maraghi, Bohyung Han, and Allan Jepson for their assistance with the experimental evaluation. The core of this work was carried out at Honda Research Institute.

## Appendix

*Proof of Lemma 1*

By definition, ($\bar{\mathbf{I}}_C = \frac{n}{n+m}\bar{\mathbf{I}}_A + \frac{m}{n+m}\bar{\mathbf{I}}_B, \bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C = \frac{m}{n+m}(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C = \frac{n}{n+m}(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_A)$) and,

$$\begin{aligned}
\mathcal{S}_C &= \sum_{i=1}^{n}(\mathbf{I}_i - \bar{\mathbf{I}}_C)(\mathbf{I}_i - \bar{\mathbf{I}}_C)^\top \\
&\quad + \sum_{i=n+1}^{n+m}(\mathbf{I}_i - \bar{\mathbf{I}}_C)(\mathbf{I}_i - \bar{\mathbf{I}}_C)^\top \\
&= \sum_{i=1}^{n}(\mathbf{I}_i - \bar{\mathbf{I}}_A + \bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C)(\mathbf{I}_i - \bar{\mathbf{I}}_A + \bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C)^\top
\end{aligned}$$

$$\begin{aligned}
&\quad + \sum_{i=m+1}^{n+m}(\mathbf{I}_i - \bar{\mathbf{I}}_B + \bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C)(\mathbf{I}_i - \bar{\mathbf{I}}_B + \bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C)^\top \\
&= \mathcal{S}_A + n(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C)(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_C)^\top + \mathcal{S}_B \\
&\quad + m(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C)(\bar{\mathbf{I}}_B - \bar{\mathbf{I}}_C)^\top \\
&= \mathcal{S}_A + \frac{nm^2}{(n+m)^2}(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)^\top \\
&\quad + \mathcal{S}_B + \frac{n^2m}{(n+m)^2}(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)^\top \\
&= \mathcal{S}_A + \mathcal{S}_B + \frac{nm}{n+m}(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)(\bar{\mathbf{I}}_A - \bar{\mathbf{I}}_B)^\top. \qquad \square
\end{aligned}$$

*Proof of Lemma 2*

When a forgetting factor of $f$ is used, the incremental PCA algorithm in Fig. 1 computes the left-singular vectors $U'$ and singular values $\Sigma'$ of the matrix $[fU\Sigma \quad \hat{B}]$. This is equivalent to computing the eigenvectors and (the square-roots of) the eigenvalues of $[fU\Sigma \quad \hat{B}][fU\Sigma \quad \hat{B}]^\top$. Now

$$\begin{aligned}
[fU\Sigma \quad \hat{B}][fU\Sigma \quad \hat{B}]^\top &= f^2U\Sigma^2U^\top + \hat{B}\hat{B}^\top \\
&= f^2U\Sigma V^\top V\Sigma^\top U^\top + \hat{B}\hat{B}^\top \\
&= f^2(A - \bar{I}_A)(A - \bar{I}_A) + \hat{B}\hat{B}^\top \\
&= f^2\mathcal{S}_A + \mathcal{S}_B + ct,
\end{aligned}$$

where $ct$ is a correction term that adjusts the mean of the eigenbasis, and $\mathcal{S}_A$ and $\mathcal{S}_B$ are scatter matrices—a scalar times the covariance matrix—as defined in Lemma 1. □

## References

Adelson, E. H., & Bergen, J. R. (1991). The plenoptic function and the elements of early vision. In M. Landy & J. A. Movshon (Eds.), *Computational models of visual processing* (pp. 1–20). Cambridge: MIT Press.

Avidan, S. (2001). Support vector tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition* (Vol. 1, pp. 184–191).

Belhumeur, P., & Kreigman, D. (1997). What is the set of images of an object under all possible lighting conditions. In *Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 270–277).

Birchfield, S. (1998). Elliptical head tracking using intensity gradient and color histograms. In *Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 232–237).

Black, M. J., & Jepson, A. D. (1996). Eigentracking: robust matching and tracking of articulated objects using view-based representation. In B. Buxton & R. Cipolla (Eds.), *LNCS: Vol. 1064. Proceedings of the fourth European conference on computer vision* (pp. 329–342). Berlin: Springer.

Black, M. J., Fleet, D. J., & Yacoob, Y. (1998). A framework for modeling appearance change in image sequence. In *Proceedings of IEEE international conference on computer vision* (pp. 660–667).

Brand, M. (2002). Incremental singular value decomposition of uncertain data with missing values. In A. Heyden, G. Sparr, M. Nielsen & P. Johansen (Eds.), *LNCS: Vol. 2350. Proceedings of* *the seventh European conference on computer vision* (pp. 707–720). Berlin: Springer.

Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(5), 564–577.

Cootes, T., Edwards, G., & Taylor, C. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(6), 681–685.

Georgescu, B., Comaniciu, D., Han, T. X., & Zhou, X. S. (2004). Multi-model component-based tracking using robust information fusion. In *2nd workshop on statistical methods in video processing*, May 2004

Golub, G. H., & Van Loan, C. F. (1996) Matrix computations. The Johns Hopkins University Press.

Hager, G., & Belhumeur, P. (1996) Real-time tracking of image regions with changes in geometry and illumination. In *Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 403–410).

Hall, P., Marshall, D., & Martin, R. (1998). Incremental eigenanalysis for classification. In *Proceedings of British machine vision conference* (pp. 286–295).

Hall, P., Marshall, D., & Martin, R. (2002). Adding and subtracting eigenspaces with eigenvalue decomposition and singular value decomposition. *Image and Vision Computing*, *20*(13–14), 1009–1016.

Harville, M. (2002). A framework for high-level feedback to adaptive, per-pixel mixture of Gaussian background models. In A. Heyden, G. Sparr, M. Nielsen & P. Johansen (Eds.), *LNCS: Vol. 2352*. *Proceedings of the seventh European conference on computer vision* (pp. 531–542). Berlin: Springer.

Isard, M., & Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In B. Buxton & R. Cipolla (Eds.), *LNCS: Vol. 1064*. *Proceedings of the fourth European conference on computer vision* (pp. 343–356). Berlin: Springer.

Jepson, A. D., Fleet, D. J., & El-Maraghi, T. F. (2003). Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(10), 1296–1311.

Jolliffe, I. T. (2002). *Principal component analysis*. Berlin: Springer.

La Cascia, M., & Sclaroff, S. (1999). Fast, reliable head tracking under varying illumination. In *Proceedings of IEEE conference on computer vision and pattern recognition* (Vol. 1, pp. 604–608).

Levy, A., & Lindenbaum, M. (2000). Sequential Karhunen–Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, *9*(8), 1371–1374.

Lim, J., Ross, D., Lin, R.-S., & Yang, M.-H. (2005). Incremental learning for visual tracking. In L. Saul, Y. Weiss & L. Bottou (Eds.), *Advances in neural information processing systems* (pp. 793–800). Cambridge: MIT Press.

Lin, R.-S., Ross, D., Lim, J., & Yang, M.-H. (2005). Adaptive discriminative generative model and its applications. In L. Saul, Y. Weiss & L. Bottou (Eds.), *Advances in neural information processing systems* (pp. 801–808). Cambridge: MIT Press.

Lucas, B., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of international joint conference on artificial intelligence* (pp. 674–679).

Matthews, I., Ishikawa, T., & Baker, S. (2004). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(6), 810–815.

Moghaddam, B., & Pentland, A. (1995). Probabilistic visual learning for object detection. In *Proceedings of IEEE international conference on computer vision* (pp. 786–793).

Murase, H., & Nayar, S. (1995). Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, *14*(1), 5–24.

North, B., & Blake, A. (1998). Learning dynamical models using expectation-maximization. In *Proceedings of IEEE international conference on computer vision* (pp. 384–389).

Rasmussen, C., & Hager, G. (1998). Joint probabilistic techniques for tracking multi-part objects. In *Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 16–21).

Ross, D., Lim, J., & Yang, M.-H. (2004). Adaptive probabilistic visual tracking with incremental subspace update. In A. Heyden, G. Sparr, M. Nielsen & P. Johansen (Eds.), *LNCS: Vol. 2350*. *Proceedings of the eighth European conference on computer vision* (pp. 707–720). Berlin: Springer.

Roweis, S. (1997). EM algorithms for PCA and SPCA. In M. I. Jordan, M. J. Kearns & S. A. Solla (Eds.), *Advances in neural information processing systems 10* (pp. 626–632). Cambridge: MIT Press.

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, *61*(3), 611–622.

Toyama, K., & Blake, A. (2001). Probabilistic tracking in metric space. In *Proceedings of IEEE international conference on computer vision* (pp. 50–57).

Vermaak, J., Lawrence, N., & Perez, P. (2003). Variational inference for visual tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition* (Vol. 1, pp. 773–780).

Williams, O., Blake, A., & Cipolla, R. (2003). A sparse probabilistic learning algorithms for real-time tracking. In *Proceedings of IEEE international conference on computer vision* (Vol. 1, pp. 353–360).