Visual Tracking Using Learned Linear Subspaces

Jeffrey Ho[‡] Kuang-Chih Lee[†] Ming-Hsuan Yang[⋆] David Kriegman[‡]
jho@cs.ucsd.edu klee10@uiuc.edu myang@honda-ri.com kriegman@cs.ucsd.edu

[‡]Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093

[†]Computer Science
University of Illinois, Urbana-Champaign
Urbana, IL 61801

[⋆]Honda Research Institute
800 California Street
Mountain View, CA 94041

Abstract

This paper presents a simple but robust visual tracking algorithm based on representing the appearances of objects using affine warps of learned linear subspaces of the image space. The tracker adaptively updates this subspace while tracking by finding a linear subspace that best approximates the observations made in the previous frames. Instead of the traditional L^2-reconstruction error norm which leads to subspace estimation using PCA or SVD, we argue that a variant of it, the uniform L^2-reconstruction error norm, is the right one for tracking. Under this framework, we provide a simple and a computational inexpensive algorithm for finding a subspace whose uniform L^2-reconstruction error norm for a given collection of data samples is below some threshold, and a simple tracking algorithm is an immediate consequence. We show experimental results on a variety of image sequences of people and man-made objects moving under challenging imaging conditions, which include drastic illumination variation, partial occlusion and extreme pose variation.

1 Introduction

The main challenge in designing a robust visual tracking algorithm is the inevitable variation in the images of the tracked object over time. Various factors can be responsible for such variation, e.g., changes in the viewpoint, changes in illumination, changes to the shape (deformations, articulations) or reflectance of the object, or partial occlusion of the target. Therefore, an important theme in visual tracking research is the design of a flexible model or representation which can adapt to appearance changes.

Typically, this requires the problem to be formulated in probabilistic terms, and the most recent and successful works on visual tracking, e.g., [12, 14, 7, 15], are all along this line. Various pixel statistics (e.g., using color or intensity values) are computed from the image sequence according to the probabilistic model deemed appropriated for the problem (commonly a Gaussian model). The dynamic evolution of the model is reflected by the different probability densities estimated from each frame. The CONDENSATION technique [11] is widely-used in visual tracking for tackling probability density estimation of this sort. Robust head tracking was demonstrated to be feasible in [2, 15] by using the aforementioned probabilistic techniques. Somewhat differently, appearance-based techniques offer another approach with less recourse to probability theory. In these approaches [10, 3], the appearance of the target is modelled using a linear subspace. The subspace is usually computed by applying Principal Component Analysis (PCA) to a collection of training images. Modelling images using a subspace has been shown to be effective in many different problems in computer vision [13]. However, comparing with the probabilistic approaches, subspace-based techniques are more rigid in the sense that they generally allow only limited number of ways to update its appearance model [4, 9], e.g. updating the covariance matrix.

This paper proposes a robust and adaptive appearance model for tracking complex natural objects based on the subspace technique. Within the subspace framework, updating the model becomes how to define a subspace L that best *approximates* a given set of data $\{x_1, \cdots, x_N\}$, the observations from the previous frames. What constitute a good approximation depends on the underlying metric one uses to define the quality of the approximation. The traditional L^2 reconstruction error norm (where $d^2(L, x_i)$ is the usual squared L^2 distance between x_i and the subspace L)

$$Error^2(L, \{x_1, \cdots, x_N\}) = \sum_{i=1}^{N} d^2(L, x_i) \qquad (1)$$

leads immediately to the well-known linear techniques of Principle Component Analysis (PCA) and Singular Value Decomposition (SVD), and a (point-based) tracking algorithm based on updating linear subspaces using this technique has appeared in [4]. Our main contribution is the observation that for appearance-based tracking, the uniform L^2 reconstruction error norm

$$Error^\infty(L, \{x_1, \cdots, x_N\}) = \max_i d^2(L, x_i) \qquad (2)$$

may be a more appropriate metric in defining linear approximations. We show that, based on this error norm, a simple algorithm can be designed to update the linear subspace.

Our algorithm is entirely appearance-based in that no any other values besides the image intensity values entered into the computation. Nor is there any complicated probabilistic estimation or non-linear optimization in the algorithm. The unexpected surprise is how robust the tracker can be made against illumination variation, pose changes and partial occlusion starting with such a simple principle. Except the two-frame based tracker, which is not known to be robust, it is difficult to imagine another tracker that is simpler, both conceptually and implementation-wise. One of the main foci of this paper is to explain in detail how such a simple tracker can work at all.

This paper is organized as follows. In the next section, we detail our tracking algorithm. Our main focus will be on the algorithmic aspect of updating the linear subspace. In the third section, we compare our algorithm with several well-known subspace-based tracking algorithms in the literature. The experimental results are reported in section four[1]. We conclude this paper with a short summary and remark on future work.

2 Tracking Algorithm

In this section, we detail our tracking algorithm. Schematically, our algorithm is very simple. We assume that the tracking window has been initialized in the first frame. At each frame, the tracker maintains an up-to-date appearance model, and the tracking task becomes a detection problem.

To estimate the location of the target in the current frame, we sample S windows $\{w_1, \cdots, w_S\}$ of different sizes and orientations near the target's location in the previous frame[2]. The image content of each window is rectified to an image of fixed size. By rastering the rectified images in the usual way, the S windows can be viewed as a collection of points $\{x_1, \cdots, x_S\}$ in some vector space $\mathbb{R}^K$. Henceforth, we call this vector space $\mathbb{R}^K$ the image space [3]. At any frame, the tracker's appearance model is represented as a linear subspace L in $\mathbb{R}^K$. The L^2-distance between each x_i and L is computed and the state of the target at current frame is defined to be the window w_i such that its corresponding x_i minimizes the distance to L among all $\{x_1, \cdots, x_s\}$. The main focus of this paper is to study how should the subspace L adapt to change as time goes on, and to this we will turn our attention next.

2.1 Subspace Update

Under the subspace framework, the most reasonable update strategy is to search for a linear subspace L that best approximates a collection of data samples $\{x_1, \cdots, x_N\}$. The data samples are the observations (tracking results) from the previous frames. To define the quality of approximation, we use the uniform reconstruction error norm $Error^\infty$ introduced n Equation 2. Suppose a pair of input parameters (N, δ) has been specified. Here N denote the number of previous frames whose tracking results we retain, and $\delta > 0$ is a threshold parameter. We define the subspace L to be *any* subspace such that the uniform reconstruction error norm between L and $\{x_1, \cdots, x_N\}$ is less than the threshold δ:

$$Error^\infty(L, \{x_1, \cdots, x_N\}) < \delta \qquad (3)$$

This definition of L is exceedingly general and the solution is generally not unique. However, one immediate consequence is that as long as δ is greater than zero, there exists at least one L that satisfies the inequality in Equation 3, namely, the subspace L spanned by the entire collection of samples $\{x_1, \cdots, x_N\}$. In particular, the magnitude of δ will generally determine the minimal possible dimension L with smaller δ requiring larger dimension of L. The non-uniqueness of the solution appears troubling at first; however, the great advantage of this is that we only need to find one such L, and it is precisely the non-uniqueness of the solutions that allows us to design a simple and computationally inexpensive algorithm to find just one such L. Having a computationally inexpensive update algorithm is necessary if the tracking algorithm is expected to run in real-time.

Clearly, Equation 3 is very different from a typical approach where the data samples are almost always linearly fitted with the least reconstruction error norm $Error^2(L, \{x_1, \cdots, x_N\})$, and the unique optimal subspace L is defined as:

$$\arg\min_L Error^2(L, \{x_1, \cdots, x_N\}) \qquad (4)$$

There are three reasons why we prefer Equation 3 to Equation 4. First, the major difference between the two equations is that Equation 3 allows for explicit control on the approximation quality of the subspace L for each x_i while Equation 4 does not. In particular, an optimal solution L may be computed using PCA or SVD techniques, but one does not know how well it approximates each sample. For instance, when the external environment (lighting, pose variation or occlusion) starts to change significantly, the first few frames of such change will invariably be considered as *outliers* for the approximation using Equation 4. An optimal L computed via Equation 4 may not approximate these samples well and one immediately runs into the serious danger of losing the target. However, Equation 3 will in effect for the dimension of L to increase in order to preserve the inequality. In this way, the tracker reacts to environmental changes more swiftly than the solution based on Equation 4. Second, the uniqueness of the solution for Equation 4 may seem to

[1]Also with supplementary video sequences

[2]Assuming rectangular windows, there are five parameters that define any windows in the image : its location, its width and height and angular orientation. Based on the window configuration in the previous frame, we sample the current collection of windows using a Gaussian distribution.

[3]In our experiment, the images are rectified to size 19x19 with $K = 361$

be a blessing at first. However, PCA or SVD require finding eigenvalues and eigenvectors, and this can be expensive when the image space turns out to be large. With the freedom provided by Equation 3, the non-uniqueness of the solutions allows us to design a fast and inexpensive algorithm for finding just one such subspace. Finally, Equation 3 naturally lends itself to a simple (approximate) solution, and this we will describe next.

Because of the natural temporal coherence among the neighboring frames, one expects that for each batch of k-consecutive frames $\{x_1, \cdots, x_k\}$ in the video sequence, the target images will not deviate from the mean of the observations from these k-frames by more than an amount δ for some δ (if not, just subdivide the k-consecutive frames into two batches). If the mean is taken as one of the basis vector of L, it is clear that, for this batch of k consecutive frames, L satisfies Equation 3. The general construction is then straightforward. For a given N observations from the previous frames, $\{x_1, \cdots, x_N\}$ and a positive integer k, we form a $D = N/k$ dimensional vector space by breaking the sequence of N images into D batches of size k. See Figure 3A. For each batch i, we compute its mean m_i and the subspace L is defined as the subspace spanned by these batch means, $\{m_1, \cdots, m_D\}$. In this modified form, our algorithm takes in the two *integral* parameters (N, k) with N the number of previous frames retained in the tracker's memory and k is the size of the batch whose mean is used to form the basis vector of L [4]. Note that the basis vectors $\{m_1, \cdots, m_D\}$ are not orthonormal and in order to compute the distance between L and x_i, we need to have an orthonormal basis of L. Therefore, the computationally non-trivial part of our update algorithm becomes updating the orthonormal basis or an incremental Gramm-Schmidt process. Note also that only the batch means are retained in the memory. All previous observations can be discarded.

2.2 Updating Orthogonal Basis

The problem is the following. Suppose the current subspace L kept by the tracker has a basis of batch means $\{m_1, \cdots, m_D\}$. Let U denote an orthonormal basis of L. The update consists of computing a batch mean m_{D+1}, adding this newly computed mean to the collection $\{m_1, \cdots, m_D\}$, and deleting the oldest mean m_1 to form the new collection of batch means $\{m_2, \cdots, m_D, m_{D+1}\}$. The updated subspace L' is spanned by these new collection of means and an orthonormal basis U' of L' has to be computed. U' can be computed from $\{m_2, \cdots, m_D, m_{D+1}\}$ by applying the Gramm-Schmidt process. The most expensive part of the Gramm-Schmidt process is the computations of the inner product between the $\frac{D(D+1)}{2}$ pairs of vectors in $\{m_2, \cdots, m_D, m_{D+1}\}$. Any other coefficient appearing in the Gramm-Schmidt process is a suitable linear combination of some of these inner products. Note that the two sets $\{m_1, \cdots, m_D\}$ and $\{m_2, \cdots, m_D, m_{D+1}\}$ differ by only two elements, namely m_1 and m_{D+1}. Therefore, the inner products computed for $\{m_1, \cdots, m_D\}$ can be retained for the next update computation. The only new new inner-products that need to be evaluated are the D inner products between the new batch mean m_{D+1} and $\{m_2, \cdots, m_D\}$ and between m_{D+1} and itself.

[4]Henceforth, N and k will always denote the two parameters of our tracking algorithm. The dimension of L will be denoted by D.

This shows that this part of the update can be made computationally inexpensive. In fact, with today's processor speed, a full Gramm-Schmidt $\{m_1, \cdots, m_D\}$, with small D (say $D < 25$, which is always the case for our experiments reported below), does not produce any noticeable effect on the tracker's performance. For the reader's convenience, we summarize both the tracking and subspace update algorithm in Figures 1 and 2.

Update Algorithm: Input (M, $\{x_1, \cdots, x_k\}$, D)
M is the collection of local means $\{m_1, \cdots, m_s\}$ maintained by the tracker and $\{x_1, \cdots, x_k\}$ is the most recent batch of observations (tracking results). D is the maximal allowable dimension of the subspace.
Output : U, an orthonormal basis of the subspace L and new $M\}$

1. Compute the new local mean m_{s+1} of the new batch of observations $\{x_1, \cdots, x_k\}$.
2. If $s \neq D$, (i.e. at the beginning stage of the tracking), form the new set $M = \{m_1, \cdots, m_s, m_{s+1}\}$ by appending m_{s+1} to M. Otherwise, delete the oldest element m_1 from $M = \{m_2, \cdots, m_{s+1}\}$.
3. Apply Gramm-Schmidt to the set of vectors M to obtain a new orthonormal basis U.

Figure 1: Subspace Updating Algorithms

2.3 Illumination, Pose and Occlusion

The main difficulty facing a visual tracking algorithm is the inevitable variation in the images of the tracked object over time. The three most important variations are the external illumination variation, pose variation and the partial occlusion of the target. The tracking task becomes challenging when one or more of these three variations develops simultaneously. In this subsection, we provide some geometric reasons and motivations that support our claim that the proposed tracking algorithm can be made robust against these three sources of variations.

A clear explanation is offered by a direct sum decomposition of the subspace $L = L_1 \oplus L_2$. Consider Figure 3B. Each green circle represents the various batch means that we used to construct L. For simplicity, we assume that as

Tracking Algorithm: Input Parameter (Ω, N, k, S)
$\Omega = \{\omega_x, \omega_y, \omega_w, \omega_h, \omega_\theta\}$ is the set of five parameters for sampling windows on the screen and S is the number of windows sampled for each frame. N is the number of previous frames retained by the tracker and k is the batch size.
Output : t, current state of the tracked object.
Internal Variable: $D = \frac{N}{k}$ is the (maximal) dimension of the subspace L. U is an orthonormal basis of L, m a local mean and is set to be the mean of previous 30 observations. M is the set of batch means, and it is set to be empty at beginning. $t = (x, y, w, h, \theta)$ the location of the target, which is represented by a rectangular box on the image at location (x, y) and of size (w, h) with orientation θ. **Initialization**: The tracker is initialized by some method. Let x_1 be the observation at the first frame. U is set to the unit vector $\frac{x_1}{\|x_1\|}$. The initialization also specifies the initial t.

1. Sample Windows: Draw S samples of windows $\{W_1, \cdots, W_S\}$ at various location of different orientations and sizes according to a Gaussian distribution centered at t with diagonal variance specified by Ω.
2. Tracking: Rectified each window W_i to a 19-by-19 image and rasterize it to form a vector x_i in $\mathbb{R}^{361}$.
3. Compute the L^2 distance between each x_i and the local mean m. Choose half of $\{x_i\}$ which has smaller L^2 distance to m. Among these $\frac{S}{2}$ vectors, evaluate their distance to the subspace L using the orthonormal basis U. This rejects half of the samples and therefore, increases the speed of the tracker.
4. let x_i be the vector in the previous step that gives the minimal distance to L. The corresponding window W_i is then the estimate of the target for the current frame. t is set W_i.
5. Subspace Update: For the interval of k frames, collect the observations $\{x_1, \cdots, x_k\}$ from the k previous frames and apply the Subspace Update Algorithm. This updates both M and U.

Figure 2: Tracking Algorithms

current frame, the algorithm retains three batch means to compute L, as illustrated in the figure. In the decomposition $L = L_1 \oplus L_2$, L_1 is the one dimensional subspace formed by the 'radial vector' that passes through the mean m (of these three batch means), and L_2 is the linear subspace that models the affine space generated by the three batch means. A basis of L_2 is represented by the two red arrows. The key observation is in realizing that although it is not a real image, each batch m_i approximates well the real images that make up its batch. In particular, the ray generated by each batch mean approximates well the rays generated by the neighboring real images. These rays are some of the rays that make up the various illumination cones [1] of the target. Note that by definition, a illumination cone is associated only with one particular pose and therefore, the rays generated by a sequence of observations from consecutive frames swept through a cross-section of a collection of illumination cones. This cross section can be approximated well by the subspace L because it contains the rays generated by the batch means. The subspace L_1 in the decomposition of L represents the average of these rays, and we can identify L_1 as the component of L that is relevant for illumination variation.

On the other hand, by going to its complement, L_2, we immediately see that L_2 models the pose variation. Using the idea of appearance manifold [13], the local linearity that can model small pose variation is precisely represented using the affine space generated by the three batch means. For instance, to model local linearity, one can compute the principle components for the set of three batches shown in Figure 3B. Note that the affine space generated by these principle components will always be contained in the affine space generated by the batch means. In particular, the L_2 component of L which models this affine space, can be considered as relevant for pose variation. Although our discussion is rather informal, these simple geometric reasons do provide some validity for expecting our tracking algorithm to be robust against pose and illumination variation.

The idea of dealing with occlusion does not seem to have a straightforward explanation using the geometry of the image space. However, the important issue in dealing with the development of occlusions using our framework is how quickly the model can adjust itself to the occlusion. One can reason that, again using the idea of appearance manifold, when the occlusion starts developing, the trajectory of the tracking sequence jumps from one appearance manifold to another. If the model does not adapt swiftly and its linearity is still modelling the local linearity of the previous appearance manifold, then one can expect the tracker to lose its target quickly. As we explain before, under our framework, it is the metric norm $Error^{\infty}$ that is responsible for the swift adaptation of our appearance model.

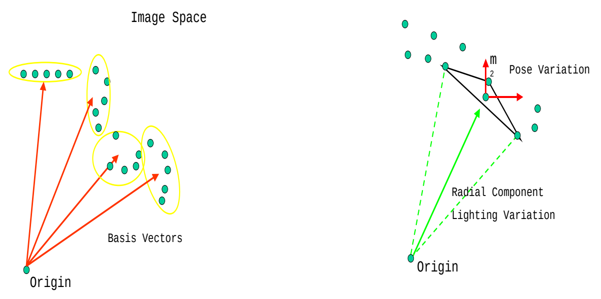


Figure 3: **Left(A)** Each batch is represented by a yellow oval and the corresponding batch mean is denoted by the red arrow. **Right(B)** Each circle represent a batch mean and currently the subspace L is spanned by the three batch means which are denoted by the three vertices of the triangle in the figure. The two red arrows represent a basis of the affine space generated by the three batch means.

2.4 Remarks and Discussion

The computational complexity of the algorithm is dominated by the number of windows generated from the normal sampling. Using 200 to 300 samples, our tracker performs comfortably in real-time on a 1.8-GHz Pentium machine in the experiments we reported below. The most prominent feature of our algorithm is its simplicity: The tracking algorithm simply takes the means of the tracking results over a constant interval and uses these batch means to form the linear subspace. No prior model learned off-line is used by the algorithm. The algorithm operates on the pixel intensity values only, and there is no sophisticated probability estimates, non-linear optimization or filtering of images as frequently seen in the tracking literature. The unexpected surprise is the robustness of the tracking algorithm given its simplicity and parsimonious starting representation (a frame).

One criticism with the proposed tracker concerns the usual problem of drift. In fact, without any prior model and learning everything on-line, it is impossible to guarantee a drift-less tracking algorithm. Our point is clear. We adhere our appearance model as close to the current observation as possible. This makes the use of the uniform norm particularly apparent. In this way, we are locally greedy and hope that the drift can be prevented as much as possible. One possible method for enhancing the tracker's ability against drift is to always include the observation made in the first frame in the appearance model. Among all the tracking results made along the video sequence, only the first observation is unquestionably the appearance of the tracked object. Observations made in the subsequent frames will invariably associated with non-zero probabilities that they are not the tracker object. Therefore, it makes sense to include the only trustworthy result in the appearance model permanently. In our implementation, the orthonormal basis always contains the normalized initial image (a unit vector in the image space). Our results show that with this small enhancement, the tracker's ability against drift can be greatly improved.

There are two important free parameters in our algorithm, N, the number of previous frames retained by the tracker and k the size of the interval used to update the subspace. In this paper, we did not address of problem of assigning the correct values for these two parameters. In the experiments we reported below, we let k range from 3 to 6 and N range from 80 to 150. We leave the problem of adaptive determination the values of N and k for future research. However, one fact is clear. That is N should not be too large as to incorporate almost all the observations made till current frame. There is an argument to be made that the best appearance model that can be used to predict the appearance of the target at the next frame should not contain the observations made way back in the sequence. Incorporating too much (or too old) information will contaminate the current appearance model with useless information, and one runs into the danger of diminishing the discriminative power of the appearance model.

3 Related Work

Needless to say, there are numerous tracking algorithms proposed in the literature. The type of tracking algorithm that is most similar to our work is the subspace-based algorithm originated with the papers of Jepson and Black [3] and Hager and Belhumeur [10]. In these pioneering papers, the subspaces are always learned off-line from some training images. In [3], a single subspace is used to provide an appearance model for tracking across different poses while in [10], the subspace is used for illumination modelling. Subsequent works along this line (e.g. [9]) has extended these earlier works by incorporating the capability of updating the eigen-model. Staying within the eigen framework, these subsequent works invariably focus their attention on methods and ways that allow the covariance matrix to be updated efficiently.

As we explained above, the fundamental difference between these earlier works and our is the metric used to define the approximations. We mentioned that one of the possible dangers of using the usual L^2 reconstruction error norm is that the model may not adapt to external change sufficiently fast. In most of these earlier papers, a separate mechanism is need to deal with occlusion. In [3], the occlusion is dealt with in a robust matching algorithm that uses a non-linear optimization technique, and in [10], it is dealt with using an iterative Re-Weighted Least Square (IRLS) algorithm that assigns an importance weight to each pixel, which down weight the pixels corresponding to the occlusion. In our algorithm, there is no separate mechanism for dealing with occlusion. We simply let the appearance model adapt to the changing imaging condition quickly and the usual pestering problem of how to deal with occlusion for tracking 'folds' naturally under our appearance model.

Pixel based algorithms (e.g. [12]) offer another approach. Here various statistics of each pixel are computed at each frame and typically it is used in the tracking algorithm through some types of EM or MAP estimation. Because the statistics are gathered at pixel level, special care is needed to guard against external illumination changes. This is usually done by passing the images through illumination insensitive filters, such as the steerable filters of [8]. Here, our appearance-based algorithm clearly offers an advantage in that by incorporating illumination variation in the subspace as we explained earlier, this type of procedure can be completely avoided. Our experiments shows that even under intense and drastic illumination variations, our simple tracker can still perform robustly.

Contour-based tracking algorithms (e.g. [2, 15, 6] operate on a domain that is somewhat different from ours. Since these algorithms track the contour of the object, the actual image content of the object is less important for this type of tracker, i.e. pose variation generally does not offer too much difficulty. However, because these algorithms operate directly with pixels, special attention is always needed to guard against both the illumination change and occlusion.

4 Experiments and Results

We have implemented the proposed method in C++ under the Microsoft Windows environment. Our current implementation runs at 30 frames/sec comfortably with 320x240 video input without any code optimization on a standard Dell P4 1.8 GHz machine. The tracking area is described by a rectangle window modelled by a 5 dimensional state vector $S = [x, y, w, h, \theta]$, where (x, y) represents the position of the tracking window, (w, h) represents the width and height of the tracking window, and θ represents the $2D$ rotation angle of the tracking window. Currently the parameters are initialized manually. For specific classes of objects (e.g., faces), the tracker could be initialized by the results of a detector.

In order to demonstrate the robustness and efficacy of our approach, we have tested the tracking algorithm on many real-world sequences. These sequences contain many difficult scenarios which a real-world tracker would likely to encounter, including changes in appearance, large pose variations, significant lighting variation and shadowing, partial occlusion, tracking object partly leaving field of view, large scale changes, cluttered backgrounds, and quick motion resulting in motion blur. We list the most difficult and representative four video sequences in the following subsections. Figures 4-7 show several key frames from these sequences and the rectified tracking window is shown in the upper left corner. The complete tracking results for these four sequences and other video sequences are included in the supplemental material accompanying this submission. All of the test video sequences and result videos will be made available to the vision community upon the publication of this paper.

4.1 Tracking a Woman's Face

The first video sequence shown in Figure 4 is a young woman walking in a cluttered office environment with 1750 frames. The long sequence demonstrates the stability of our proposed algorithm. The challenge of this video sequence includes large pose variation during walking, significant lighting variation and shadowing when she turns on the desk lamp, and the partial occlusion when she drinks coffee and walks behind a cubical wall. In addition, the cluttered background and shaky motion from a handheld camera also increase the tracking difficulty.

4.2 Tracking a Man's Face

Figure 5 shows some difficult tracking frames in the Dudek Sequence from the University of Toronto, which originally appeared in [12]. The sequence contains lots of activities which cause significant appearance changes, such as a hand totally occluding the face for a short time, and taking the glasses on and off.

4.3 Tracking A Man's Face with Occlusion and Pose Variation

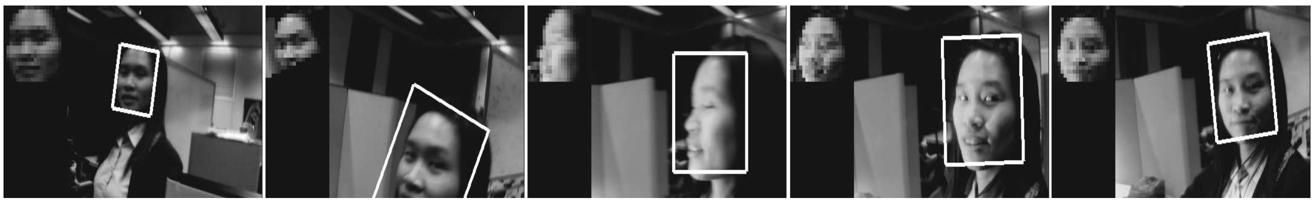
Figure 6 shows the result of tracking a man's face while it is partially occluded by a book. The pose of the man's face varies simultaneously with the development of occlusion.

4.4 Tracking a Human Face under Lighting Changes

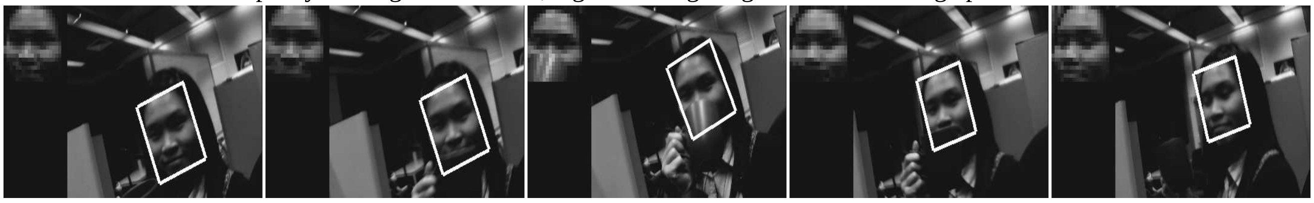
The last video sequence shown in Figure 7 is from Boston University, where it shows the capability of our tracker in keeping good result when both pose and lighting condition varied. The sequence can be downloaded in the public website [5].

5 Summary and Conclusions

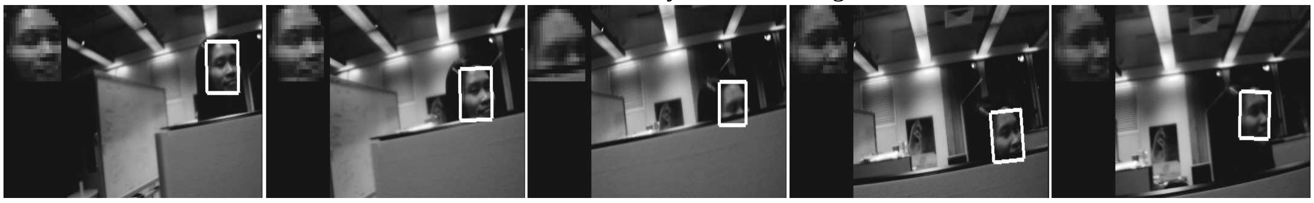
In this paper, we have introduced a technique for learning on-line a representation of the appearances of an object that is being tracked. By representing the appearances as a linear subspace and choosing to satisfy a constraint using a well-chosen metric, the resulting tracker is both simple and fast. As demonstrated in eight challenging video sequences, the method can robustly track an object in the presence of large viewpoint changes, partial occlusion, drastic lighting variation, changes to the shape of the object (facial expressions, adding glasses), shaky cameras, and motion blur.



Face partly leaving field of view, significant lighting variation and large pose variation



Partial occlusion by the coffee mug



Partial occlusion by walking behind a cubic wall

Figure 4: Sequence of tracking a woman's face. Each row represents a set of 5 key frames of a particular event. The image in the upper left corner is the first frame of the video. The upper left corner of each displayed frame shows the cropped image.

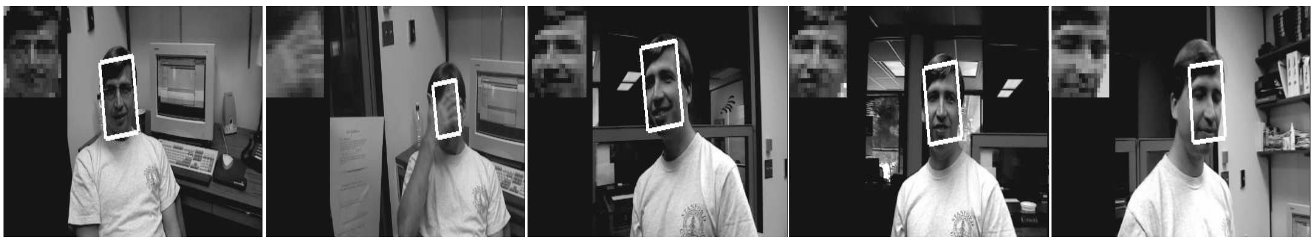


Figure 5: Demonstration of tracking a man's face. The tracking results of some difficult frames are selected for this figure.

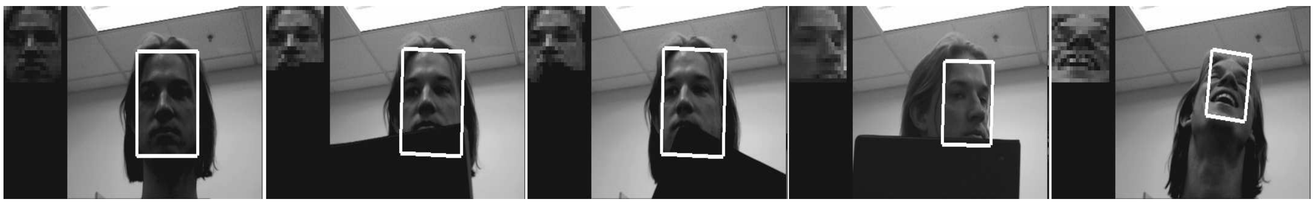


Figure 6: Demonstration of tracking a man's face while it undergoes pose change and occlusion. 5 key frames are displayed.

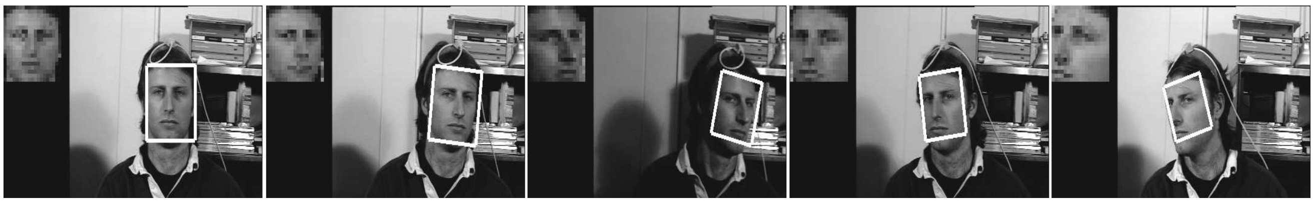


Figure 7: Demonstration of tracking a man's face under lighting variation. The tracking results of some difficult frames are selected in this figure. The wire loop on the head is part of the original sequence.

Within a larger context, one intermediate goal of a tracker might be to learn all appearances of an object online, i.e., the appearance manifold. By constructing a subspace using only some of the most recent images in a video, one basically arrives at a representation which is an embedding of a small neighborhood of the appearance manifold in a low dimensional linear subspace. Rather than using the neighborhood of the manifold, we instead use the linear subspace as the representation. Note that if the neighborhood were very small, the subspace would just be a subset of the tangent space to the appearance manifold, but here the neighborhood is larger, and so the dimension of the subspace must be larger. As a consequence of only learning a neighborhood of the appearance manifold, the representation does not contain all appearances of the object. Yet for tracking, the viewing parameters generally change continuously, and so representing a neighborhood of appearances is sufficient for effective tracking. Though not revealed in our experiments, a potential challenge remains which is a long term drift of the tracking window off the tracked object since our local representation may become corrupted with the background or occluding objects. One potential way to mitigate this problem is to enhance the representation with an approximation to the global appearance manifold coupled with our local representation. For example, one knows a priori that the image used to initialize the tracker is always a valid image on the appearance manifold, and likewise a sparse sampling of representative images from the entire tracking sequence could be retained. How to effectively, efficiently, and quickly sample, represent, utilize and integrate such information in capturing the global appearance structure of an object remain open challenges.

Acknowledgments

We would like to thank Mei-Fang Huang for her great help to record and build the supplemental video sequences, and all of the anonymous reviewers for their comments and suggestions.

References

[1] P. Belhumeur and D. Kriegman. What is the set of images of an object under all possible lighting conditions. *Int'l. J. Computer Vision*, 28:245–260, 1998.

[2] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 232–237, 1998.

[3] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proc. European Conf. on Computer Vision*, pages 329–342, 1996.

[4] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proc. European Conf. on Computer Vision*, volume 2350, pages 707–720, 2002.

[5] M. L. Cascia, J. Isidoro, and S. Sclaroff. Image and video computing group. In *http://www.cs.bu.edu/groups/ivc/HeadTracking/publications.html*.

[6] Y. Chen, Y. Rui, and T. Huang. Jpdaf based hmm for real-time contour tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 543–550, 2001.

[7] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 142–149, 2000.

[8] W. Freeman and E. H. Adelson. The design and use of steerable filters. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 13, pages 891–906, 1991.

[9] N. Gupta, P. Mittal, S. Roy, S. Chaudhury, and S. Banerjee. Condensation-based predictive eigentrackin. *Microsoft Preprint*, 2002.

[10] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 100–100, 1998.

[11] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. In *Int'l. J. Computer Vision*, 1998.

[12] A. Jepson, D. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 415–422, 2001.

[13] H. Murase and S. K. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int'l. J. Computer Vision*, 14, 1995.

[14] K. Toyama and A. Blake. Probabilistic tracking in a metric space. In *Proc. Int'l. Conf. on Computer Vision*, volume 2, pages 50–59, 2001.

[15] Y. Wu and T. S. Huang. A co-inference approach to robust visual tracking. In *Proc. Int'l. Conf. on Computer Vision*, volume 2, pages 26–33, 2001.