Efficient appearance-based tracking

José Miguel Buenaposada
Dpto. de Informática, Estadística y Telemática
ESCET, Univ. Rey Juan Carlos
C/ Tulipán, s/n
28933 Móstoles, Madrid, Spain
Email: jmbuenaposada@escet.urjc.es

Enrique Muñoz, Luis Baumela
Departamento de Inteligencia Artificial
Fac. de Informática, Univ. Politécnica de Madrid
Campus de Montegancedo s/n
28660 Boadilla del Monte, Madrid, Spain
Email: kike@dia.fi.upm.es, lbaumela@fi.upm.es

Abstract

One of the major challenges that visual tracking algorithms face nowadays is being able to cope with changes in the appearance of the target during tracking. Linear subspace models have been extensively studied recently and are possibly the most popular way of modeling target appearance. Unfortunately, efficiency is one of the limitations of present linear subspace models, and this is a key feature for a good tracker. In this paper we present an efficient procedure for tracking based on a linear subspace model of target appearance (grey levels). A set of motion templates is built from the subspace base, which is used to efficiently compute target motion and appearance parameters. It differs from previous works in that we impose no restrictions on the subspace used for modeling appearance. In the experiments conducted we have built a modular PCA-based face tracker which shows that video-rate tracking performance can be achieved with a non optimized implementation of our algorithm.

1 Introduction

Tracking plays a fundamental role in many important applications of computer vision such as intelligent human computer interaction, autonomous robot guidance or video processing. One of the major challenges that visual tracking algorithms face nowadays is being able to cope with changes in the appearance of the target during tracking. These appearance changes can be caused by a variation in the illumination, an occlusion or a change in the aspect of the target itself caused by a change of pose or, for example, in the case of face tracking, by a change of facial expression. Tracking algorithms try to accommodate these variations by modeling target appearance in various ways. Some use texture [14], color [6] or shape [13] statistics, or both [4], others employ textured 3D models [17], and finally, many use linear subspace models of texture [5, 12] or shape and texture [8]. In this paper we will present an efficient algorithm for tracking which models changes in appearance with a linear subspace model of texture.

Linear subspace models are possibly the most popular way of representing appearance. Images of a target lie in a low dimensional manifold or subspace whose dimensions represent the underlying degrees of freedom of the imaged object. For example, the images of an eye lie on a three dimensional subspace, one dimension associated to the amount of eye aperture and the other to the orientation of the pupil. The popularity of these models comes from their simplicity and computational efficiency and because they have been thoroughly studied within the pattern recognition and statistics communities. In computer vision they have been successfully used for recognizing 3D objects under varying pose [19], representing and recognizing faces [3, 25], tracking with illumination changes [12] or with changes in pose [5], and tracking of deformable objects [24], among many others.

Normally, the relationship between the input image and the manifold is nonlinear, but useful results have been obtained using linear mappings between them. Principal Component Analysis (PCA) and Factor Analysis (FA) are two examples of this. Principal Component Analysis (PCA) can be obtained as the eigenvectors of the sample covariance matrix associated with the largest eigenvalues. This has proven to be an excellent tool for dimensionality reduction of multivariate data, hence, if and image is considered to be a multivariate datum, PCA can be a useful tool for manifold construction. Here we will use PCA for modeling the subspace of appearances of our target, a human face, under different facial expressions.

Several extensions to conventional linear models have been proposed over time. For example, Independent Component Analysis (ICA) is an attempt to attain independence among the components of a multivariate vector [7]. In cases where linear subspace models do not suffice, mixtures of linear models [23, 10, 11] or Locally Linear Embedding (LLE) [20] techniques can be used.

One of the major limitations of PCA is that it needs normalized sample images in the training data. This means that images have to be normalized and geometrically aligned both when building the subspace model and when projecting incoming images onto it. This has been solved either by using subspaces [16] and projection procedures [21] which are invariant to these geometrical transformations or by robustly registering the images [9, 22].

Effi ciency is an important limitation of present subspace models, which has not drawn much attention so far, with the exception of [12]. Very often tracking algorithms have to perform in real-time, as the flow of images reaches the computer vision system. Although some recent works claim to achieve near real-time performance [9], none has considered the issue of effi ciency. In this paper we present an effi cient procedure for tracking using a linear subspace model. During the training phase of our algorithm motion templates associated to the subspace image base are computed, so that a smaller number of calculations have to be made during tracking.

Motion templates have been successfully used previously for real-time tracking [17, 12], but they used models which could not deal with some changes of target appearance. For example, in the case of face tracking, a restricted subspace model was used in [12] to achieve robustness to illumination changes, which could not be used to model a change of facial expression.

The tracking algorithm presented in this paper can be seen as an extension of the one introduced in [12] in that we impose no restrictions on PCA-based subspace model used. It is also related to [5], but instead of computing the motion parameters by using a gradient descent procedure in which the target image Jacobian must be computed for each frame in the sequence, as in [5], we use a set of precomputed motion templates which alleviate the computations that have to be performed online.

Throughout the paper we will denote scalars with lowercase letters, vectors with lowercase letters with a bar on them (e.g. $\bar{x}$,$\bar{\mu}$) and matrices with uppercase boldface letters (e.g. $\mathbf{B}$).

2 Factored eigentracking

Let P be the image of a target. The subspace constancy equation holds for all pixels in the target [5]:

$$\mathbf{I}(f(\bar{x},\bar{\mu}),t) = [\mathbf{B}\bar{c}(t)](\bar{x}) \;\; \forall x \in P, \tag{1}$$

where $\bar{x}$ is the vector of co-ordinates of a point in image $\mathbf{I}$, $\mathbf{B}$ is the subspace base matrix, $\bar{c}$ is the vector of subspace coeffi cients, and $\mathbf{I}(f(\bar{x},\bar{\mu}),t)$ is the image acquired at time t rectifi ed with motion model $f(\bar{x},\bar{\mu})$ and motion parameters $\bar{\mu}$. By $[\mathbf{B}\bar{c}](x)$ we denote the value of $\mathbf{B}\bar{c}$ for the pixel with position $\bar{x}$ in the image. Matrix $\mathbf{B}$ is of dimension $N \times k$, where N is the number of pixels per image and k is the number of basis vectors in the subspace. Intuitively (1) states that the rigidly rectifi ed image $\mathbf{I}(f(\bar{x},\bar{\mu}),t)$ can be expressed as a linear combination of the appearance subspace basis vectors, $\mathbf{B}$[1]

Tracking consists on estimating for each image in the sequence the values of the motion, $\bar{\mu}$, and appearance, $\bar{c}$, parameters which minimize the error function

$$E(\bar{\mu},\bar{c}) = ||\mathbf{I}(f(\bar{x},\bar{\mu}),t) - [\mathbf{B}\bar{c}(t)](\bar{x})||^2. \tag{2}$$

In order to robustly estimate the minimum value of (2), the quadratic error norm can be replaced by a robust one (e.g. [5, 12]).

In general, minimizing (2) can be a diffi cult task as it defi nes a non-convex objective function. Several procedures have been proposed to solve this problem, which can be grouped into those using gradient descent [5] and those using Gauss-Newton iterations [12, 2, 15]. Black and Jepson [5] presented an iterative solution by using a gradient descent procedure and a robust metric with increasing resolution levels. Computationally, their algorithm is quite demanding as, for example, the Jacobian of each incoming image has to be computed once on every frame for each level in the multi-resolution pyramid.

In order to make Gauss-Newton iterations, a Taylor series expansion of $\mathbf{I}$ at $(\bar{x},t)$ is performed, producing a new error function

$$E(\delta\bar{\mu},\bar{c}) = ||\mathbf{M}\delta\bar{\mu} + \bar{i}(f(\bar{x},\bar{\mu})) - \mathbf{B}\bar{c}||^2, \tag{3}$$

where $\bar{i}(\bar{x})$ is $\mathbf{I}(\bar{x})$ in vector form, and $\mathbf{M} = \frac{\partial \bar{i}(f(\bar{x},\bar{\mu}))}{\partial \bar{\mu}}$ is the $N \times n$ ($n = \dim(\bar{\mu})$) Jacobian matrix of $\bar{i}$ (note that dependence on t has been dropped for convenience). Hager and Belhumeur [12], in the context of invariance to illumination changes, introduced an effi cient procedure for minimizing (3) by assuming $\nabla_{\bar{x}}[\mathbf{B}\bar{c}](\bar{x}) \approx 0$. In this case $\mathbf{M}$ can be expressed in terms of the gradient of a fi xed template image and can be partially precomputed off-line. The result of this off-line computation is a set of parametrized motion templates, which only depend on $\bar{\mu}$, and can be used to effi ciently track a planar object. In general, the previous assumption is not valid, and the computed motion templates can not be reliably used for tracking objects whose appearance changes due to causes other than illumination (e.g. changes in pose). In the following subsections we will introduce a procedure for precomputing a set of motion templates which effi ciently minimize (3) for any linear subspace model.

[1] We assume that that the average image has been included as the fi rst column of $\mathbf{B}$.

2.1 Jacobian matrix factorization

One of the obstacles for minimizing (3) online, while tracking, is the computational cost of estimating $\mathbf{M}$ for each frame. In this subsection, following an approach similar to [12], we will show that $\mathbf{M}$ can be factored into the product of two matrices, $\mathbf{M}_0\mathbf{\Sigma}(\bar{\mu},\bar{c})$, where $\mathbf{M}_0$ is a constant matrix, which can be computed off-line.

Each element m_{ij} of $\mathbf{M}$ can be written as

$$m_{ij} = \nabla_{\bar{f}}\mathbf{I}(f(\bar{x}_i,\bar{\mu}_j),t_n)^\top f_{\bar{\mu}}(\bar{x}_i,\bar{\mu}_j). \tag{4}$$

Taking derivatives w.r.t. $\bar{x}$ on both sides of (1) we get

$$\nabla_{\bar{f}}\mathbf{I}(f(\bar{x}_i,\bar{\mu}_j),t_n)^\top f_{\bar{x}}(\bar{x}_i,\bar{\mu}_j) = \nabla_{\bar{x}}[\mathbf{B}\bar{c}(t)](\bar{x}). \tag{5}$$

Finally, from (4) and (5) we get a new expression for $\mathbf{M}$,

$$\mathbf{M}(\bar{\mu},\bar{c})=\begin{bmatrix} (\sum_j \nabla_{\bar{x}}[\bar{b}_j c_j](\bar{x}_1))^\top f_{\bar{x}}(\bar{x}_1,\bar{\mu})^{-1} f_{\bar{\mu}}(\bar{x}_1,\bar{\mu}) \\ \vdots \\ (\sum_j \nabla_{\bar{x}}[\bar{b}_j c_j](\bar{x}_N))^\top f_{\bar{x}}(\bar{x}_N,\bar{\mu})^{-1} f_{\bar{\mu}}(\bar{x}_N,\bar{\mu}) \end{bmatrix}, \tag{6}$$

where $\bar{b}_j$ is the jth column of $\mathbf{B}$ and c_j is the jth element of the appearance vector $\bar{c}$.

Let

$$\mathbf{B}_\nabla(\bar{x}_i)=\left[\begin{bmatrix} \nabla_u[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_u[\bar{b}_k](\bar{x}_i) \end{bmatrix}^\top, \begin{bmatrix} \nabla_v[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_v[\bar{b}_k](\bar{x}_i) \end{bmatrix}^\top\right] \tag{7}$$

and

$$\mathbf{C} = \begin{bmatrix} c_1 & \cdots & c_k & 0 & \cdots & 0 \\ 0 & \cdots & 0 & c_1 & \cdots & c_k \end{bmatrix}^\top, \tag{8}$$

where u and v are the horizontal and vertical image coordinates respectively. Then (6) can be finally rewritten as

$$\mathbf{M}(\bar{\mu},\bar{c}) = \begin{bmatrix} \mathbf{B}_\nabla(\bar{x}_1)\mathbf{C} f_{\bar{x}}(\bar{x}_1,\bar{\mu})^{-1} f_{\bar{\mu}}(\bar{x}_1,\bar{\mu}) \\ \vdots \\ \mathbf{B}_\nabla(\bar{x}_N)\mathbf{C} f_{\bar{x}}(\bar{x}_N,\bar{\mu})^{-1} f_{\bar{\mu}}(\bar{x}_N,\bar{\mu}) \end{bmatrix}. \tag{9}$$

Therefore $\mathbf{M}$ can be expressed in terms of the gradient of the subspace basis vectors, $\mathbf{B}_\nabla$, which are constant, and the motion and appearance parameters $(\bar{\mu},\bar{c})$, which vary over time. If we choose a motion model f such that $\mathbf{C} f_{\bar{x}}(\bar{x}_i,\bar{\mu})^{-1} f_{\bar{\mu}}(\bar{x}_i,\bar{\mu}) = \mathbf{\Gamma}(\bar{x}_i)\mathbf{\Sigma}(\bar{\mu},\bar{c})$, then $\mathbf{M}$ can be factored into

$$\mathbf{M}(\bar{\mu},\bar{c})=\begin{bmatrix} \mathbf{B}_\nabla(\bar{x}_1)\mathbf{\Gamma}(\bar{x}_1) \\ \vdots \\ \mathbf{B}_\nabla(\bar{x}_N)\mathbf{\Gamma}(\bar{x}_N) \end{bmatrix} \mathbf{\Sigma}(\bar{\mu},\bar{c})=\mathbf{M}_0\mathbf{\Sigma}(\bar{\mu},\bar{c}), \tag{10}$$

where $\mathbf{M}_0$ is constant matrix and $\mathbf{\Sigma}$ depends on $\bar{c}$ and $\bar{\mu}$. The columns of $\mathbf{M}_0$ are the motion templates of our tracking algorithm.

2.2 Minimizing $E(\bar{\mu},\bar{c})$.

As $\mathbf{M}$ depends on both, $\bar{\mu}$ and $\bar{c}$, (3) defines a nonlinear cost function over $\delta\bar{\mu}$ and $\bar{c}$. The optimization algorithm that we use first assumes $\bar{c}$ constant and computes the minimum of $E(\bar{\mu},\bar{c})$ w.r.t. $\bar{\mu}$,

$$\delta\bar{\mu} = -(\mathbf{\Sigma}^\top\mathcal{M}\mathbf{\Sigma})^{-1}\mathbf{\Sigma}^\top\mathbf{M}_0^\top[\bar{i}(f(\bar{x},\bar{\mu}),t+\tau) - \mathbf{B}\bar{c}(t)], \tag{11}$$

where $\mathcal{M} = \mathbf{M}_0^\top\mathbf{M}_0$. Then it minimizes E over $\bar{c}$ assuming $\bar{\mu}$ constant,

$$\bar{c} = \mathbf{B}^\top[\mathbf{M}\delta\bar{\mu} + \bar{i}(f(\bar{x},\bar{\mu}),t+\tau)]. \tag{12}$$

The term $\mathbf{M}\delta\bar{\mu}$ is the grey level variation in $\mathbf{I}$ due to a motion of magnitude $\delta\bar{\mu}$. Intuitively equation (12) states that the appearance parameters are computed by projecting onto the subspace the rectified image corrected to take into account the incremental motion $\delta\bar{\mu}$. Once we have $\bar{c}$, we can refine the estimation of $\delta\bar{\mu}$ by using (11) again. Normally two or three iterations over this process are enough to reach a stable solution.

In summary, the steps of our tracking algorithm are:

- **Off-line:**
  1. Compute the basis images gradients, $\nabla[b_i](\bar{x})$.
  2. Compute all $\Gamma(\bar{x})$ matrices.
  3. Compute and store $\mathbf{M}_0$.
  4. Compute and store $\mathcal{M}$.
- **Online:**
  1. Warp $I(\bar{z},t+\tau)$ to compute $I(f(\bar{x},\bar{\mu}_t),t+\tau)$.
  2. Build the reconstructed image vector, $\mathbf{B}\bar{c}(t)$.
  3. Compute $\mathcal{E} = [\bar{i}(f(\bar{x},\bar{\mu}_t),t+\tau) - \mathbf{B}\bar{c}(t)]$.
  4. Compute $\mathbf{\Sigma}$.
  5. Compute $\mathbf{\Sigma}^\top\mathbf{M}_0^\top$.
  6. Compute $\mathbf{\Sigma}^\top\mathbf{M}_0^\top\mathcal{E}$.
  7. Compute $(\mathbf{\Sigma}^\top\mathcal{M}\mathbf{\Sigma})^{-1}$.
  8. From (11) compute $\delta\bar{\mu}_{t+\tau}$.
  9. From (12) compute $\bar{c}(t+\tau)$ using $\delta\bar{\mu}_{t+\tau}$.

Let k the number of basis vectors, n the number of motion parameters and N the number of pixels in the region to track. Then the computational cost of the off-line part of the algorithm is shown in table 1.

Step (1)	Step (2)	Step (3)	Step (4)	Total
O(kN)	O(kN)	O(k^2nN)	O(k^2n^2N)	O(k^2n^2N)

Table 1. Computational cost of the off-line part of the factored eigentracking algorithm.

The time required to make an iteration of the online part is shown in table 2. The total time comes mainly from steps

(5), O(kn^2N), and (7), O(k^2n^3). Only when the number of pixels, N, is very low (typically 10×10 images) step (7) dominates the computation time. When using images of size 20×20 and above, most of the time is spent multiplying and transposing the Jacobian matrix, $(\mathbf{\Sigma M}_0)^\top$. By optimizing the matrix to matrix multiplication procedure we could improve the performance of this step.

Step (1)	Step (2)	Step (3)	Step (4)	Step (5)	Step (6)
O(nN)	O(kN)	O(N)	O(k)	O(kn^2N)	O(nN)

Step (7)	Step (8)	Step (9)	Total
O(k^2n^3)	O(n^2)	O($kN + nN$)	O($kn^2N + k^2n^3$)

Table 2. Computational cost of the online part of the factored eigentracking algorithm.

2.3 Some usual motion models

In this subsection we will show how the previous tracking algorithm can be used with some motion models commonly used in computer vision.

2.3.1 Rotation, translation and scale model

This motion model can be described by four parameters, $\bar{\mu} = (\theta, t_u, t_v, s)$, corresponding to rotation, translation and scale, $f(\bar{x}, \bar{\mu}) = s\mathbf{R}(\theta)\bar{x} + \bar{t}$, where $\bar{x} = (u, v)^\top$, $\bar{t} = (t_u, t_v)^\top$ and $\mathbf{R}(\theta)$ is a 2D rotation matrix. Taking derivatives of f with respect to $\bar{x}$ y $\bar{\mu}$,

$$f_{\bar{x}}(\bar{x}, \bar{\mu}) = s\mathbf{R}(\theta), \tag{13}$$

$$f_{\bar{\mu}}(\bar{x}, \bar{\mu}) = \left[\mathbf{I}_{2\times 2} \,|\, -s\mathbf{R}(\theta)\begin{bmatrix}-v\\ u\end{bmatrix} \,|\, \mathbf{R}(\theta)\begin{bmatrix}u\\ v\end{bmatrix}\right], \tag{14}$$

where the $\mathbf{I}_{d\times d}$ is the $d \times d$ identity matrix. Introducing (13) and (14) into (9), we get the factorization:

$$\mathbf{\Gamma}(\bar{x}_i) = \left[\ \mathbf{I}_{2k\times 2k},\quad \begin{bmatrix} -v_i\mathbf{I}_{k\times k} & u_i\mathbf{I}_{k\times k} \\ u_i\mathbf{I}_{k\times k} & v_i\mathbf{I}_{k\times k}\end{bmatrix}\ \right],$$

$$\mathbf{\Sigma}(\bar{c}, \bar{\mu}) = \begin{bmatrix} \mathbf{C}\frac{1}{s}\mathbf{R}(-\theta) & 0 \\ 0 & \mathbf{C}\begin{bmatrix}1 & 0\\ 0 & \frac{1}{s}\end{bmatrix}\end{bmatrix}.$$

For this model $\mathbf{M}_0$ has dimensions $N \times 4k$ and $\mathbf{\Sigma}$, $4k \times 4$.

2.3.2 Affine model

The 2D affine motion model can be written as $f(\bar{x}, \bar{\mu}) = \underbrace{\begin{bmatrix} a & c \\ b & d\end{bmatrix}}_{\mathbf{A}}\bar{x} + \begin{bmatrix} e \\ f\end{bmatrix}$, where $\mathbf{A}$ is a nonsingular matrix and $\bar{\mu} = (a, b, c, d, e, f)^\top$ are the six model parameters. Taking derivatives of f with respect to $\bar{x}$ and $\bar{\mu}$,

$$f_{\bar{x}}(\bar{x}, \bar{\mu}) = \mathbf{A}, \quad f_{\bar{\mu}}(\bar{x}, \bar{\mu}) = [\mathbf{I}_{2\times 2}|x\mathbf{I}_{2\times 2}|y\mathbf{I}_{2\times 2}]. \tag{15}$$

From (15) and (9), we get the desired factorization:

$$\mathbf{M}_0 = \begin{bmatrix} \mathbf{B}_\nabla(\bar{x}_1)(\mathbf{I}_{2k\times 2k}|x_1\mathbf{I}_{2k\times 2k}|y_1\mathbf{I}_{2k\times 2k}) \\ \vdots \\ \mathbf{B}_\nabla(\bar{x}_N)(\mathbf{I}_{2k\times 2k}|x_N\mathbf{I}_{2k\times 2k}|y_N\mathbf{I}_{2k\times 2k})\end{bmatrix},$$

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{CA}^{-1} & 0 & 0 \\ 0 & \mathbf{CA}^{-1} & 0 \\ 0 & 0 & \mathbf{CA}^{-1}\end{bmatrix},$$

where $\mathbf{M}_0$ has dimensions $N \times 6k$ and $\mathbf{\Sigma}$ has $6k \times 6$.

2.3.3 Projective model

Let $\bar{x} = (u, v)^\top$ and $\bar{x}_h = (r, s, \lambda)^\top$ be respectively the Cartesian and Projective coordinates of an image pixel. They are related by: $\bar{x}_h = (r, s, \lambda)^\top \rightarrow \bar{x} = (r/\lambda, s/\lambda)^\top = (u, v)^\top$; $\lambda \neq 0$. The 2D projective linear transformation can be written as

$$f(\bar{x}_h, \bar{\mu}) = \mathbf{H}\bar{x}_h = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & 1\end{bmatrix}\begin{bmatrix} r \\ s \\ \lambda\end{bmatrix},$$

where $\bar{\mu} = (a, b, c, d, e, f, g, h)^\top$. Now $\mathbf{B}_\nabla(\bar{x}_i)$ has an extra set of columns associated to the gradient of the homogeneous coordinate[2],

$$\mathbf{B}^P_\nabla(\bar{x}_i) = \left[\begin{bmatrix} \nabla_r[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_r[\bar{b}_k](\bar{x}_i)\end{bmatrix}^\top, \begin{bmatrix} \nabla_s[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_s[\bar{b}_k](\bar{x}_i)\end{bmatrix}^\top, \begin{bmatrix} \nabla_\lambda[\bar{b}_1](\bar{x}_i) \\ \vdots \\ \nabla_\lambda[\bar{b}_k](\bar{x}_i)\end{bmatrix}^\top\right] \tag{16}$$

and matrix C is

$$\mathbf{C}^P = \begin{bmatrix} c_1 & \cdots & c_k & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & c_1 & \cdots & c_k & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & c_1 & \cdots & c_k\end{bmatrix}^\top.$$

Taking derivatives of f with respect to $\bar{x}_h$ and $\bar{\mu}$,

$$f_{\bar{x}_h}(\bar{x}_h, \bar{\mu})^{-1} = \mathbf{H}^{-1}, \tag{17}$$

$$f_{\bar{\mu}}(\bar{x}_h, \bar{\mu}) = [r\mathbf{I}_{3\times 3} \,|\, s\mathbf{I}_{3\times 3} \,|\, \lambda\mathbf{I}_{1-2}], \tag{18}$$

where $\mathbf{X}_{a-b}$ is the matrix composed with rows a to b of $\mathbf{X}$. Then, from (16–18) and (9) the factorization of $\mathbf{M}$ arises:

$$\mathbf{M}_0 = \begin{bmatrix} \mathbf{B}^P_\nabla(\bar{x}_1)(r_1\mathbf{I}_{3k\times 3k}|s_1\mathbf{I}_{3k\times 3k}|\lambda_1\mathbf{I}_{3k\times 3k}) \\ \vdots \\ \mathbf{B}^P_\nabla(\bar{x}_N)(r_N\mathbf{I}_{3k\times 3k}|s_N\mathbf{I}_{3k\times 3k}|\lambda_N\mathbf{I}_{3k\times 3k})\end{bmatrix}, \tag{19}$$

[2] $\nabla_{\bar{x}_h}\mathbf{I}(\bar{x}_h) = \left[\frac{\partial \mathbf{I}}{\partial u}, \frac{\partial \mathbf{I}}{\partial v}, -u\frac{\partial \mathbf{I}}{\partial u} - v\frac{\partial \mathbf{I}}{\partial v}\right]^\top$

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{C}^P\mathbf{H}^{-1} & 0 & 0 \\ 0 & \mathbf{C}^P\mathbf{H}^{-1} & 0 \\ 0 & 0 & \mathbf{C}^P\mathbf{H}^{-1}_{1-2} \end{bmatrix}. \quad (20)$$

Now the dimensions of $\mathbf{M}_0$ and $\mathbf{\Sigma}$ are $N \times 9k$ and $9k \times 8$ respectively.

3 Modular factored eigentracking

A modular eigenspace is a partition of the original data vector into subsets (modules) in order to compute an independent subspace model for each of them. This allows a more flexible, compact, accurate and better conditioned model of the regions of interest [9]. We will consider that all the regions are part of the same object and hence that they share the same motion parameters increment but could have different appearance. In our case we will use different subspace models for each of the eyes and the mouth.

Let $\{\mathbf{B}_1, \cdots, \mathbf{B}_r\}$ be the set of subspace basis for all modules. Then matrix $\mathbf{B}_{me}$ for modular eigentracking can be written as:

$$\mathbf{B}_{me} = \begin{bmatrix} \mathbf{B}_1 & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & \mathbf{B}_r \end{bmatrix}, \quad (21)$$

which is a block diagonal matrix representing the disjoint sets of regions which compose the image. The appearance of each region is modeled by subspace base $\mathbf{B}_i$. Therefore, the appearance parameter vector will be $\bar{c} = (\bar{c}_1^\top, \cdots, \bar{c}_r^\top)^\top$, where $\bar{c}_i$ is the parameter vector of module i. When computing $\mathbf{M}_0$, the gradients of $\mathbf{B}_{me}$ are obtained independently for each $\mathbf{B}_i$ and, as before, introduced in $\mathbf{B}_\nabla$. Finally, $g_i(\bar{\mu})$ is a function that relates the motion parameters of module i to a common reference system. The factored modular eigentracking algorithm is as follows:

- **Off-line:**
  1. Compute and store $\mathbf{M}_0$ using B_{me}.
  2. Compute and store $\mathcal{M}$.
- **Online:**
  1. For each region i do:
     a) Warp $\mathbf{I}(z, t+\tau)$ to $\mathbf{I}(f(x, g_i(\bar{\mu}_t)), t+\tau)$.
     b) $\mathcal{E}_i = [\bar{i}(f(\bar{x}, g_i(\bar{\mu}_t)), t+\tau) - \mathbf{B}_i\bar{c}_i(t)]$.
  2. The error term is now $\mathcal{E} = (\mathcal{E}_1^\top, \cdots, \mathcal{E}_r^\top)^\top$.
  3. Compute $\mathbf{\Sigma}(\bar{c}(t), \bar{\mu}_t)$.
  4. From (11) compute $\delta\bar{\mu}$ using the new $\mathcal{E}$.
  5. From (12) compute $\bar{c}(t+\tau)$ using $\delta\bar{\mu}$ and $\mathbf{B}_{me}$.
  6. Update $\bar{\mu}_{t+\tau} = \bar{\mu}_t + \delta\bar{\mu}$.
  7. Update each $\bar{c}_i$ vector from $\bar{c}(t+\tau)$.

Figure 1. Some samples from the sequence used in the first experiment.

4 Experiments

We have implemented our algorithm in C++ in a GNU/Linux environment. We used the INTEL IPL (Image Processing Library) routines for image warping and the *dgemm* BLAS routine for matrix multiplication (in the ATLAS optimized version for Pentium IV). No other special optimization has been made in the current code. The computer in which the tests were performed is a Pentium IV 2.4 GHz with 512 KBytes of cache and 512 MBytes of DDR memory. The image sequences were acquired with a Sony VL500 and a Unibrain Fire-i (the fourth experiment) firewire cameras.

In the first experiment the performance of the algorithm is tested in terms of time needed to make an iteration with different motion models (n), number of pixels (N), and subspace dimension (k). In this case we used a sequence with 595 images with both eyes and eyebrows (see Fig. 1). The time per iteration in milliseconds is shown in table 3.

In table 4 we show the frame rate achieved when the algorithm performs two Gauss-Newton iterations per frame. With the proposed algorithm we can achieve standard video rate performance with any 68×28 pixels patch whose appearance could be modeled with a subspace of dimension smaller than 40. Also, given the special structure of the gray levels of a human face, which is mainly made up of low-frequency components, it can be safely tracked with a low dimensional subspace (e.g. k=7) for which frame rates ranging from 16.7 f.p.s to 151.5 f.p.s can be achieved, depending on the number of tracked pixels (N) and on the motion model complexity (n).

	N=136 × 56			N=68 × 28		
	k=7	k=13	k=44	k=7	k=13	k=39
Projective (n=8)	29.9	41.1	98	5.6	7.7	16.9
Affine (n=6)	20.3	28.8	71.6	4.1	5.2	11.5
RTS (n=4)	14.3	21.5	57.1	3.3	4.3	8.6

Table 3. Time per iteration in milliseconds.

In the second experiment we show the performance for a projective motion model. The training sequence used for PCA (the same as the first experiment) is different from the one used for tracking. In this case the subspace dimension was 13, the size of the image patch was 68x28 and the frame rate achieved with three Gauss-Newton iterations per frame was 32 f.p.s. The difference from the 65 f.p.s. shown in table 3 for two iterations, is mainly due to the overhead

	N=136 × 56			N=68 × 28		
	k=7	k=13	k=44	k=7	k=13	k=39
Projective (n=8)	16.7	12.1	5.1	89.3	65	29.6
Affi ne (n=6)	24.6	17.4	7	122	96.1	43.5
RTS (n=4)	35	23.3	8.8	151.5	116.3	58.1

Table 4. Frames per second with two iterations per frame.

of drawing results, loading images from disk and performing the extra Gauss-Newton iteration. In the experiment the head performs moderate out of plane rotations and the tracker is able to cope with them. In Fig. 2 are shown the results of the test. The estimated position of the three regions is overlayed over the current image and on its right side are shown the rectifi ed image (top) and the reconstructed image (bottom).

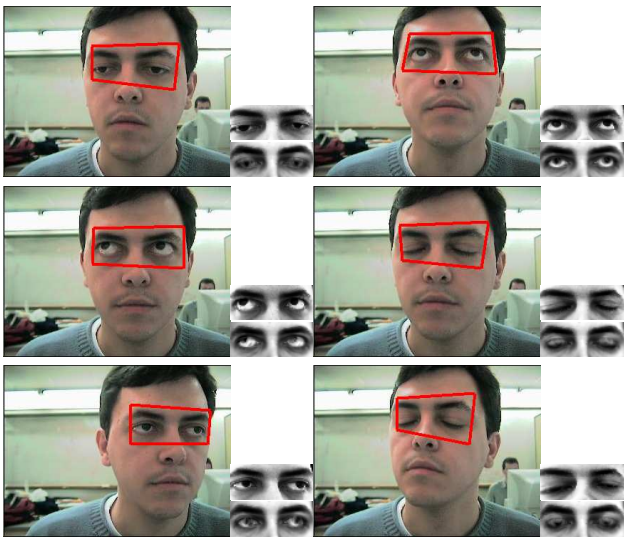

Figure 2. Projective appearance based tracking. Results for a 643 image sequence.

In the third experiment we test the performance of the tracker for an ideal situation in which the appearance model is the optimum for a given dimension, i.e. we track the same image sequence used for training the appearance subspace. We use a modular appearance model for the mouth and both eyes, a projective motion model and make two Gauss-Newton iterations per frame in the optimization procedure. As shown in Fig. 3, tracking performs quite well in terms of motion parameters and, as the illumination is the same for training and tracking, the appearance is estimated correctly in all frames. In this test the tracker is able to work at 18 f.p.s with the projective model, 26 f.p.s with the affi ne model and 34 f.p.s with the rotation-translation-scale motion model [3].

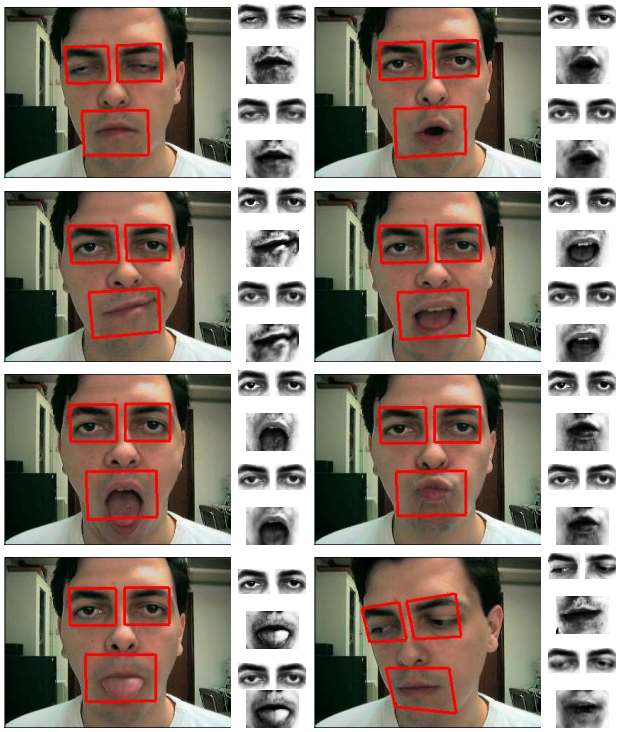

Figure 3. Projective modular appearance based tracking. Results for a 798 image sequence. This sequence was also used for training the subspace appearance model.

In the last experiment we test the performance of the tracker for a more challenging sequence. We acquired a very long sequence in order to use half of the sequence for training the appearance subspace and the other half for tracking. We use a modular appearance model for the mouth (35 by 23 pixels) and both eyes (33 by 35 pixels images each), a rotation-translation-scale motion model and make four Gauss-Newton iterations per frame in the optimization procedure. As shown in Fig. 4, tracking performs quite well in terms of motion parameters and the appearance is estimated correctly in all frames. In this test the tracker is able to work at 13 f.p.s with the rotation-translation-scale motion model [4].

5 Discussion

In this paper we are dealing with the problem of incremental image alignment for tracking. A traditional solu-

[3] Frame rates for this experiment include the time needed for image decoding and showing results.

[4] Frame rates for this experiment include the time needed for image decoding and showing results.

tion is the well known Lucas and Kanade algorithm [18]. It is based on minimizing the first order approximation to the difference between the template and the rectified images This approach is quite demanding in terms of computational resources as the Jacobian matrix,

$$\mathbf{M} = \left. \frac{\partial I(f(\bar{x}, \bar{\mu}), t + \delta t)}{\partial \bar{\mu}} \right|_{\bar{\mu}=\bar{\mu}_t},$$

has to be recomputed on each frame in the sequence. The output of the algorithm are the increment of motion parameters, $\delta\bar{\mu}$, such that $\bar{\mu}_{t+\delta t} = \bar{\mu}_t + \delta\bar{\mu}$. This is an additive approach in contrast to a compositional one in which $f(\bar{x}, \bar{\mu}_{t+\delta t}) = f(f(\bar{x}, \delta\bar{\mu}_c), \bar{\mu}_t)$ [2].

The work of Hager and Belhumeur[12] reduce the online computational cost by computing a factorization the Jacobian matrix, $\mathbf{M}$, into $\mathbf{M}_0(\bar{x})\Sigma(\bar{\mu})$. This reduces the online cost of the algorithm to the computation of the inverse of Σ (see [12] for details). On the other hand, the inverse compositional approach of Baker and Matthews[2] achieves the same goal of reducing the online computation by changing the role of the template and rectified images. The Jacobian of the template image with respect to the motion parameters is pre-computed and the online computation is also small (see [2] for details).

The Jacobian factorization idea, although first introduced in the context of rigid tracking by Hager and Belhumeur[12], has been reused by us in a new development in the context of appearance-based tracking. Our approach consists of a linear appearance and a motion model, like in Black and Jepson's eigentracking [5]. It differs from Active Appearance Models [1] in that we have no shape model. The main difference between the original eigentracking [5] and our approach is that in the former efficiency issues were not considered (e.g. image gradients have to be recomputed for each frame and for each level in the pyramid).

Currently there are two ways of efficiently performing incremental image alignment, the Jacobian factorization and the inverse compositional approach. The main contribution of this paper is extending the Jacobian factorization approach to deal with appearance changes. Another contribution we have made, is the use of the Jacobian factorization for tracking with a projective motion model. This was not solved in [12] and recently it has been claimed that it could not be solved with such approach [1, 2].

6 Conclusions

Efficiency is the key for appearance based methods to be useful in tracking applications. In this paper we have presented an efficient procedure for tracking using a linear subspace model of target appearance. Efficiency is gained by precomputing the set of motion templates which arise in a factorization of the image Jacobian used in the minimization of the tracking error function. We have also shown how to make this factorization for some usual motion models: rotation, translation and scale, affine and projective.

In the experiments conducted we have shown that standard video rate performance can be easily achieved for tracking a human face or any other image patch with moderate size and low frequency texture.

There are still some important open issues on which we are currently working, namely, how to efficiently deal with illumination changes and target occlusions.

Acknowledgment

The authors gratefully acknowledge funding from the Spanish Ministry of Science and Technology under grant number TIC2002-000591. Enrique Muñoz was funded by a FPU grant from the Spanish Ministry of Education.

References

[1] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Proc. of International Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–1090–I–1097. IEEE, 2001.

[2] S. Baker and I. Matthews. Lukas-kanade 20 years on: A unifying framework. *Int. Journal of Computer Vision*, 56(3):221–255, 2004.

[3] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):711–720, July 1997.

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

[5] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

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

[7] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):11–20, 1994.

[8] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. In *Proc. European Conference on Computer Vision*. Springer-Verlag, 1998.

[9] F. de la Torre and M. Black. Robust parameterized component analysis. In *Proc. European Conference on Computer Vision, LNCS 2353*, pages 653–669. Springer-Verlag, 2002.

[10] B. Frey, A. Colmenarez, and T. Huang. Mixtures of local linear subspaces for face recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 32–37, June 1998.

[11] Z. Ghahramani and G. Hinton. The em algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, 1997.

[12] G. D. Hager and P. N. Belhumeur. Effi cient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analisys and Machine Intelligence*, 20(10):1025–1039, 1998.

[13] M. Isard and A. Blake. Condensation–conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):2–28, 1998.

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

[15] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):996–100, 2002.

[16] A. Kannan, N. Jojic, and B. Frey. Fast transformation-invariant factor analysis. In *Advances in Neural Information Processing Systems 2002*, volume 15. MIT-Press, 2003.

[17] M. La Cascia, S. Sclaroff, and V. V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on robust registration of texture-mapped 3d models. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 22(4):322–336, April 2000.

[18] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of Imaging Understanding Workshop*, pages 121–130, 1981.

[19] S. Nayar, H. Murase, and S. Nene. Parametric appearance representation. In S. Nayar and T. Poggio, editors, *Early visual learning*, pages 131–160. Oxford University Press, 1996.

[20] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2001.

[21] A. Shashua, A. Levin, and S. Avidan. Manifold pursuit: a new approach to appearance based recognition. In *Proc. of International Conference on Pattern Recognition, ICPR2002*, volume III, pages 590–594, Quebec, Canada, August 2002. IEEE.

[22] D. Skokaj, H. Bischof, and A. Leonardis. A robust pca algorithm for building representations from panoramic images. In *Proc. European Conference on Computer Vision*, volume LNCS 2353, pages 761–775. Springer-Verlag, 2002.

[23] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[24] L. Torresani, D. Yang, G. Alexander, and C. Bregler. Tracking and modelling non-rigid objects with rank constraints. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition*. IEEE, 2002.

[25] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neurosience*, 3(1), 1991.

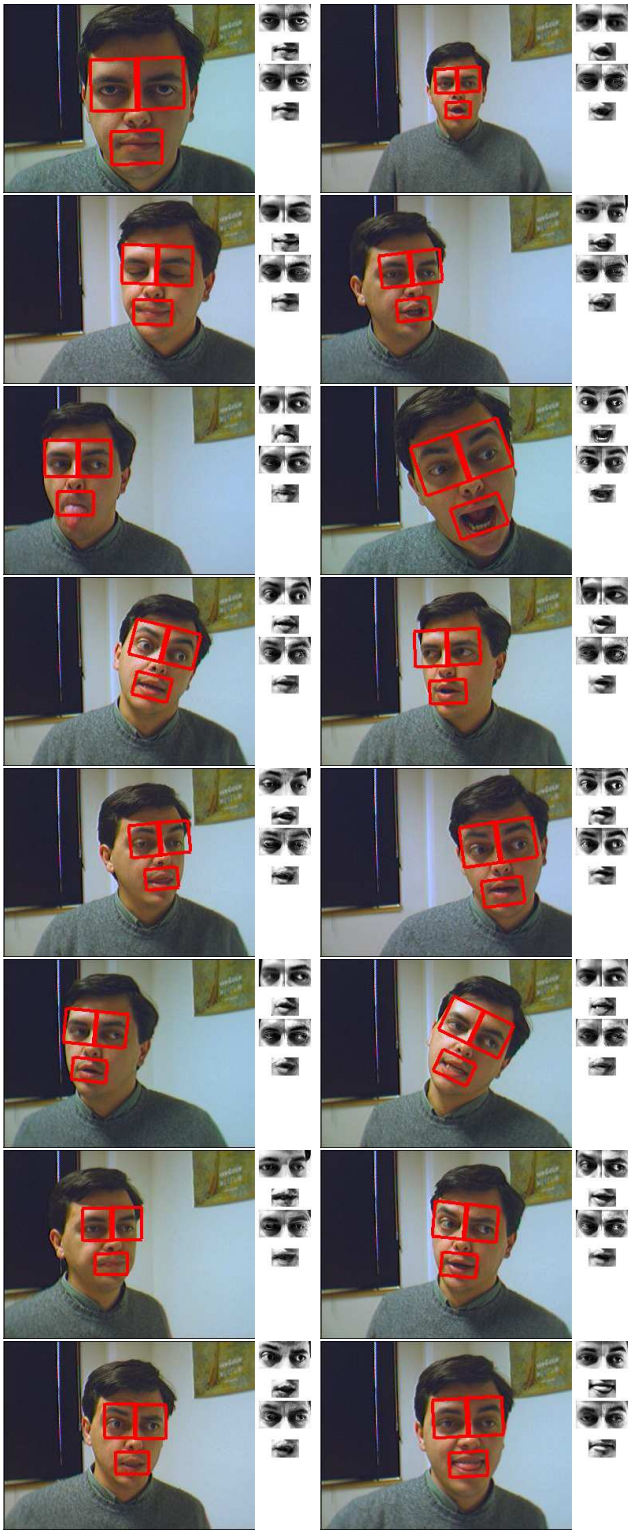

Figure 4. Rotation-translaton-scale modular appearance based tracking. Results for a 4787 image sequence. Half of the sequence was used in training (2720 images) and the other half for tracking (2067 images).