Hyperplane Approximation for Template Matching

Frédéric Jurie and Michel Dhome

Abstract—Hager and Belhumeur [6] recently proposed a general framework for object tracking in video images. It consists of low-order parametric models for the image motion of a target region. These models are used to predict movement and to track the target. The difference in intensity between the pixels belonging to the current region and the pixels of the selected target (learned during an offline stage) allows a straightforward prediction of the region position in the current image. The main aim of this article is to propose an important improvement within this framework, making the convergence faster with the same amount of online computation.

Index Terms—Visual tracking, motion estimation.

1 Introduction

In traditional tracking approaches, there are two major groups: either the tracking is performed from local correspondences (*feature-based* approaches) or from template correspondences (*template-based* approaches). *Feature-based* approaches use local features such as points, line segments, edges, or regions. Such techniques are naturally less sensitive to partial occlusions as they are based on local correspondences. If several correspondences are missing, the pose is still computable.

On the other hand, *global* or *template-based* approaches take the template as a whole. The strength of these methods lies in their ability to treat complex templates or patterns that cannot be modeled by local features. They are very robust and have been extensively used. They have also been called *sum-of-square-difference (SSD)* as they consist of minimizing the difference between a reference template and a region of the image. A $L_2 norm$ is generally used to measure the error. Historically, a brute force search was used. But, this strategy is impractical in the case of transformations more complex than 2D translations, which involve higher dimensional parameter spaces. More recent methods treat the problem as a nonlinear optimization problem using Newton type or Levenberg-Marquardt based algorithms.

Darell et al. [4] and Brunelli and Poggio [2] propose maximizing a correlation criterion between a vector characterizing the reference pattern and the image content. The processing times—significant in this case—can be reduced by working in subspaces of the initial image representation. The main limitation of these approaches is their lack of resistance with regard to occlusions. Black and Jepson [1] have overcome this limitation by reconstructing the occluded parts. They replace the quadratic norm generally used to construct the approximation of the image in the eigenspace by a robust error norm. This reconstruction involves the minimization of a nonlinear function performed using a simple gradient descent scheme. They used the same scheme to find the parametric transformation aligning the pattern on the image.

More recently, a new efficient framework has been proposed: The tracking problem is posed as being the problem of finding the best (in the least-squares sense) set of parameter values describing the motion and deformation of a target through the sequence. In this case, parameter variations are written as a linear function of difference images (difference between reference image and current image). This approach is very efficient, as motion can be easily deduced from the difference image. Cootes et al. [3] use it to dynamically estimate the parameters of a face appearance model (2D model). Only a few works use this approach with projective transformations [5], [7] because projective transformations are highly nonlinear and because of the size of the parameter space.

Hager and Belhumeur [6] have recently proposed an efficient framework for this kind of problem. The position of the target template in the first image is supposed to be known. The problem is then to estimate the position of this template in the subsequent images. The actual position of the template in the current image can be computed by comparing the gray-level values of the target template with the gray-level values of the predicted region. This computation is possible because—during an offline training stage—a relation between the variations in intensities and the variations in position has been learned.

Hager and Belhumeur [6] propose estimating this relation by using the inverse of the Jacobian image. The aim of this article is to show that this relation can be obtained using a different approach (hyperplane approximation) leading to better results without any additional computation. The gain is in the extent of the convergence area. In this article, the framework proposed by Hager and Belhumeur [6] includes models for image changes due to motion illumination and partial occlusion. The proposed formulation has been presented for motion changes only for purposes of clarity; however, it is directly applicable to illumination changes and partial occlusions.

This article comprises four sections. The first is devoted to a short presentation of the tracking framework proposed by Hager and Belhumeur. In the second section, the proposed approach is described and compared to the original one. The way to use the proposed approach in tracking applications is explained in the third section. Finally, in the last section, experiments involving real images are carried out in order to show the superiority of the proposed approach.

2 Efficient Region Tracking

We adopt almost identical notations as those proposed by Hager and Belhumeur [6] in order to make the reading easier. Bold fonts denote vectors and matrices.

Let $I(\mathbf{x}, t)$ be the brightness value at the location $\mathbf{x} = (x, y)$ in an image acquired at time t. Let the set $\mathcal{R} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ be the set of N image locations which define a *target region*. $\mathbf{I}(\mathcal{R}, t) = (I(\mathbf{x}_1, t), I(\mathbf{x}_2, t), \cdots, I(\mathbf{x}_N, t))$ is a vector of the brightness values of the target region at time t. We refer to $\mathbf{I}(\mathcal{R}, t_0)$ as *the reference template*. It is the template which is to be tracked; t_0 is the initial time $(t = 0)$.

The relative motion between the object and the camera induces changes in the position of the template in the image. We assume that these transformations can be perfectly modeled by a parametric *motion model* $\mathbf{f}(\mathbf{x}; \mu(t))$, where $\mathbf{x}$ denotes an image location and $\mu(t) = (\mu_1(t), \mu_2(t), \dots, \mu_n(t))$ denotes a set of parameters. We assume that $N > n$ and that $\mathbf{f}$ is differentiable both in $\mathbf{x}$ and μ. We call μ the motion parameter vector. The set of N image locations $(\mathbf{f}(\mathbf{x}_1; \mu(t)), \mathbf{f}(\mathbf{x}_2; \mu(t)), \dots, \mathbf{f}(\mathbf{x}_N; \mu(\mathbf{t})))$ is denoted $\mathbf{f}(\mathcal{R}; \mu(t))$. At time t_0, the position of the template is $\mu(t_0)$, also denoted μ_0^*.

With these assumptions, "tracking the object at time t" means "compute $\mu(t)$ such that $\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu(t)), t) = \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0^*), t_0)$." We write $\mu(t)$ the estimate of the ground truth value $\mu^*(t)$.

The motion parameter vector of the target region $\mu(t)$ can be estimated by minimizing the least squares function:

$$O(\mu(t)) = \|\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0^*), t_0) - \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu(t)), t)\|.$$

- *The authors are with LASMEA—CNRS UMR 6602, Université Blaise Pascal, 63177 Aubière, France.*
E-mail: {frederic.jurie, michel.dhome}@lasmea.univ-bpclermont.fr.

Manuscript received 15 Nov. 2000; revised 23 Oct. 2001; accepted 21 Nov. 2001.
Recommended for acceptance by D. Fleet.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 113140.

This very general formulation of tracking has been used by several authors. Black and Jepson [1] give a good example of how this minimization can be carried out. They proposed using an optimization algorithm (Levenberg-Marquard), which is unfortunately slow and can only tolerate very small movements of the object.

Hager and Belhumeur [6] propose a very straightforward and efficient computation of $\mu(t+\tau)$ by writing:

$$\mu(t+\tau) = \mu(t) + \mathbf{A}(t+\tau)\big(\mathbf{I}(\mathbf{f}(\mathcal{R};\mu_0^*), t_0) - \mathbf{I}(\mathbf{f}(\mathcal{R};\mu(t)), t+\tau)\big), \tag{1}$$

where $\mathbf{A}(t+\tau)$ can be obtained with small online computation and τ denotes the time between two successive images. This formulation makes real-time implementations on standard workstations possible.

If we write $\delta\mathbf{i}(t+\tau) = \mathbf{I}(\mathbf{f}(\mathcal{R};\mu_0^*, t_0) - \mathbf{I}(\mathbf{f}(\mathcal{R};\mu(t)), t+\tau)$ and $\delta\mu(t+\tau) = \mu(t+\tau) - \mu(t)$, (1) can be written as:

$$\delta\mu(t+\tau) = \mathbf{A}(t+\tau)\delta\mathbf{i}(t+\tau). \tag{2}$$

3 Jacobian Approximation (JA) versus Hyperplane Approximation (HA)

3.1 Jacobian Approximation [6]

Equation (2) shows clearly that $\mathbf{A}(t+\tau)$ plays the role of a Jacobian matrix. For this reason, we will record it as $\mathbf{A}_j(t+\tau)$. The estimate of $\mathbf{A}_j(t+\tau)$ can be obtained, as proposed by Hager and Belhumeur [6] by using the Jacobian image.

In order to simplify notations, we will denote $\mathbf{I}(\mathbf{f}(\mathcal{R};\mu(t)), t)$ by $\mathbf{I}(\mu, t)$. If the magnitude of the components of $\delta\mu$ and τ are small, it is possible to linearize the problem by expanding $\mathbf{I}(\mu+\delta\mu, t+\tau)$ in a Taylor series about μ and t,

$$\mathbf{I}(\mu+\delta\mu, t+\tau) = \mathbf{I}(\mu, t) + \mathbf{I}_\mu(\mu, t)\delta\mu + \mathbf{I}_t(\mu, t)\tau + h.o.t.,$$

where $h.o.t.$ are the high order terms of the expansion that can be neglected; $\mathbf{I}_\mu(\mu, t) = \mathbf{M}(\mu, t)$ is the Jacobian matrix of $\mathbf{I}$ with respect to μ at time t, and $\mathbf{I}_t$ is the derivative of $\mathbf{I}$ with respect to t.

By neglecting the $h.o.t.$ and with the additional approximation $\tau\mathbf{I}_t(\mu, \tau) = \mathbf{I}(\mu, t+\tau) - \mathbf{I}(\mu, t)$, assuming $\delta\mathbf{i} = \mathbf{I}(\mu+\delta\mu, t+\tau) - \mathbf{I}(\mu, t+\tau)$ the previous equation becomes

$$\delta\mathbf{i}(t) = \mathbf{M}(\mu, t)\delta\mu. \tag{3}$$

By writing

$$\mathbf{A}_j(t) = (\mathbf{M}^t(\mu, t)\mathbf{M}(\mu, t))^{-1}\mathbf{M}^t(\mu, t), \tag{4}$$

we obtain a direct expression of $\mathbf{A}_j(t)$ (where $\mathbf{M}^t$ denotes the transposition of $\mathbf{M}$).

By combining (2) and (4), we obtain:

$$\delta\mu = (\mathbf{M}^t(\mu, t)\mathbf{M}(\mu, t))^{-1}\mathbf{M}^t(\mu, t)\delta\mathbf{i} = \mathbf{A}_j(t)\delta\mathbf{i}. \tag{5}$$

The straightforward computation of $\mathbf{M}$ requires the computation of the image gradient with respect to the component of vector $\mathbf{f}$. Therefore, $\mathbf{M}$ depends on time-varying quantities and has to be completely recomputed at each new iteration. This is a computationally expensive procedure. Fortunately, it is possible to express $\mathbf{M}$ as a function of the image gradient of the reference image, allowing us to obtain $\delta\mu = (\mathbf{M}^t\mathbf{M})^{-1}\mathbf{M}^t\delta\mathbf{i}$ with only few online computations [6].

Equation (5) involves the computation of the difference in intensity $\delta\mathbf{i}$. It is possible to relate $\delta\mathbf{i}$ to the *reference template* given in the first image. If we assume that the pattern is correctly localized after the correction of the motion parameter $\delta\mu$, the image consistency assumption gives $\mathbf{I}(\mu+\delta\mu, t+\delta\tau) = \mathbf{I}(\mu_0^*, t_0)$, leading to the relation $\delta\mathbf{i} = \mathbf{I}(\mu_0^*, t_0) - \mathbf{I}(\mu, t+\tau)$.

In this case, (5) links the difference between the template in the current region and the target template with a displacement $\delta\mu$ aligning the region on the target. With these notations, the tracking consists of evaluating $\delta\mathbf{i}(t+\tau)$ and, consequently, obtaining $\delta\mu(t+\tau)$ and, finally, updating $\mu(t+\tau)$ according to the equation: $\mu(t+\tau) = \mu(t) + \delta\mu(t+\tau)$.

3.2 Hyperplane Approximation

We propose a different interpretation of the computation of matrix A. Equation (2) $\delta\mu(t+\tau) = \mathbf{A}(t+\tau)\delta i(t+\tau)$ can be seen as the modellization by n hyperplanes. In this section, matrix $\mathbf{A}$ is written $\mathbf{A}_h$ to distinguish it from $\mathbf{A}_j$. However, it plays the same role. Let us write a_{ij} the elements of matrix $\mathbf{A}_h$ (time is removed in order to simplify notations). Equation (2) can be written as:

$$(a_{11}, \dots, a_{1j}, \dots, a_{1N}, -1)(\delta i_1, \dots, \delta i_j, \dots, \delta i_N, \delta\mu_1)^t = 0$$
$$(a_{i1}, \dots, a_{ij}, \dots, a_{iN}, -1)(\delta i_1, \dots, \delta i_j, \dots, \delta i_N, \delta\mu_i)^t = 0$$
$$(a_{n1}, \dots, a_{nj}, \dots, a_{nN}, -1)(\delta i_1, \dots, \delta i_j, \dots, \delta i_N, \delta\mu_n)^t = 0.$$

Under this form, we can clearly observe that $a_{i1}, \dots, a_{ij}, \dots, a_{iN}$ are the coefficients of n hyperplanes that can be estimated by using a least-square estimate.

During the training stage, the region of interest is moved from the reference position μ_0^* (provided manually by selecting the region of interest in the first image) to $\mu_0\prime = \mu_0^* + \delta\mu_0$. Once the region is moved, the vector $\delta\mathbf{i} = \mathbf{I}(\mathcal{R}, \mu_0^*) - \mathbf{I}(\mathcal{R}, \mu_0\prime)$ is computed. This "disturbance" procedure is repeated N_p times with $N_p > N$.

Finally, we collect N_p couples $(\delta\mathbf{i}^k, \delta\mu^k), k \in [1, N_p]$, with $\delta\mathbf{i}^k = (\delta i_1^k, \dots, \delta i_N^k)^t$ and $\delta\mu^k = (\delta\mu_1^k, \dots, \delta i_n^k)^t$. It is then possible to obtain a matrix A_h such that: $\sum_{k=1}^{k=N_p}(\delta\mu^k - \mathbf{A}_h\delta\mathbf{i}^k)^2$ is minimal.

By writing $\mathbf{H} = (\delta\mathbf{i}^1, \dots, \delta\mathbf{i}^{N_p})$, $\mathbf{Y} = (\delta\mu^1, \dots, \delta\mu^{N_p})$, and assuming $N_p > N$ we obtained an overdetermined system. $\mathbf{A}_h$ can be computed from $\mathbf{Y} = \mathbf{A}_h\mathbf{H}$ by $\mathbf{A}_h = \mathbf{Y}\mathbf{H}^t(\mathbf{H}\mathbf{H}^t)^{-1}$, where $\mathbf{H}^t$ denotes the transposition of $\mathbf{H}$. A similar scheme has already been proposed by Cootes et al. [3]. They have used it to dynamically estimate the parameters of a face appearance model.

4 Advantages of Hyperplane Approximation

In the remaining part of the article, arguments μ or t will be removed when obvious from the context.

Although the same equation $\delta\mu = \mathbf{A}\delta\mathbf{i}$ is used in both cases, we will show that results obtained by using the hyperplane modelization are better than those obtained using the Jacobian image.

The reason is that in the case of Hyperplane Approximation (HA), the best linear approximation giving the motion from image differences is obtained by using the data directly (pairs of δi and $\delta\mu$ vectors produced by small disturbances).

In the case of Jacobian Approximation (JA), these data are first used to estimate, sampling point by sampling point, a linear relationship giving image differences as a function of motion parameters, and the final expected relations are computed from the first approximations.

The JA assumes that the gray levels are linear combinations of motion parameters. This assumption, which is not confirmed in real images, is not necessary in the HA case.

4.1 Simple Numerical Application

Let us suppose an image line having gray levels values corresponding to the function: $I(x) = exp(\frac{-x^2}{2})$. The region to track is $\mathcal{R} = (x_1, x_2) = (-0.1, 0.0)$. Let us suppose that the transformation is a pure translation $(\mu = tx)$, $f(x, \mu) = x + tx$. The learning stage consists of producing 1,000 random disturbances on tx, using a uniform law in the range $[-.25, +.25]$, giving pairs of δtx and δi values. In the case of hyperplane approximation, we directly obtain $\mathbf{A}_h = (13.42, -13.41)$. In the

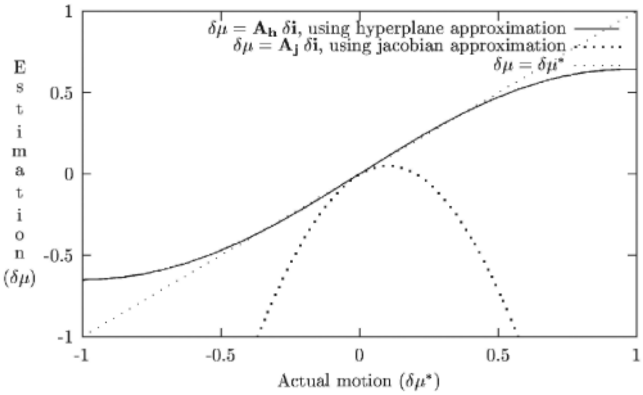

Fig. 1. The Jacobian approximation and the hyperplane approximation are compared.

case of Jacobian approximation, we first compute each value of the Jacobian matrix $\mathbf{M}$. The same pairs of $\delta\mu^k$ and $\delta\mathbf{i}^k$ vectors given by the random disturbances are used to compute $\mathbf{M}$. By writing $\mathbf{H} = (\delta\mathbf{i}^1, \ldots, \delta\mathbf{i}^{N_p})$, $\mathbf{Y} = (\delta\mu^1, \ldots, \delta\mu^{N_p})$, an estimation of $\mathbf{M}$ is given by: $\mathbf{M} = \mathbf{H}\mathbf{Y}^t(\mathbf{Y}\mathbf{Y}^t)^{-1}$ because $\mathbf{M}$ has to satisfy $\mathbf{H} = \mathbf{M}\mathbf{Y}$ (3). Therefore, exactly the same data are used in both cases (HA and JA). We finally obtain

$$\mathbf{A}_j = (\mathbf{M}^t\mathbf{M})^{-1}\mathbf{M}^t = (11.05, 1.06).$$

Comparison. The point is now to evaluate which one of the two modelizations is better. This was done by taking $\delta\mu^*$ values in $[-1, 1]$ from the previous example. Each one of these disturbances gives a $\delta\mathbf{i}$ vector. An estimate of the motion $\delta\mu$ can be obtained using the relation $\delta\mu = \mathbf{A}\delta\mathbf{i}$. If the approximation was perfect, we should obtain $\delta\mu = \delta\mu^*$. Results are presented in Fig. 1. This figure represents $\delta\mu$ as a function of $\delta\mu^*$. The hyperplane approximation appears to be much better than the Jacobian approximation.

5 Efficient Tracking with Image Hyperplanes

In the previous section, we have used an simple example to show the superiority of the hyperplane approximation (HA) over the Jacobian approximation (JA). However, the HA scheme is very inefficient, taken under its initial form. Direct computation of matrix $\mathbf{A}_h$ involves a least-square minimization, which is to be repeated for each new image. The matrix depends on the current position, orientation, etc. given by μ. The learning stage consists of computing a linear relation between a set of gray-level differences and a correction of the parameters μ. This relationship is computed around the value μ_0^*—known when the user selects an image region—and is not valid for other values of μ.

We shall see how it is possible to establish this relation for any value of μ, without recomputing the matrix.

5.1 Learning Stage

Let the region be defined as $\mathcal{R} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ the set of N point locations in a local reference called the *region reference*. The function $\mathbf{f}(\mathbf{x}; \mu)$ changes the coordinates of $\mathbf{x} = (x, y)$ in the reference region into $\mathbf{u} = (u, v) = \mathbf{f}(\mathbf{x}; \mu)$ in the reference image.

When the user defines a target region in the reference image, he defines a set of correspondences between points in the *reference region* and points in the *reference image* (for example, the corners of a rectangular region). Knowing this set of correspondences, the computation of μ_0^* such that $\mathbf{f}(\mathbf{x}; \mu_0^*)$ aligns the *reference region* on the target (defined in the *reference image*) is possible.

The learning stage consists of producing small random disturbances $\delta\mu$ around μ_0^*. We denote such disturbances as $\mu_0' = \mu_0^* + \delta\mu$. These disturbances produce the change of brightness $\delta\mathbf{i} = \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0^*)) - \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0'))$.

A set of N_p disturbances $\delta\mu$ are produced in order to obtain the linear model giving $\delta\mu = \mathbf{A}_h\delta\mathbf{i}$. During the training stage, it is possible to estimate motion $\delta\mu$ knowing $\delta\mathbf{i}$.

As shown on Fig. 2a, if $\mathbf{u}'$ are the coordinates of $\mathbf{x}$ in the reference image under the transformation μ_0', then $\mathbf{u}' = \mathbf{f}(\mathbf{x}; \mu_0')$. Let $\mathbf{x}'$ be such that $\mathbf{u} = \mathbf{f}(\mathbf{x}'; \mu_0')$. Assuming $\mathbf{f}$ is invertible, we obtain $\mathbf{x}' = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{x}; \mu_0^*); \mu_0')$.

Therefore, knowing $\delta\mathbf{i}$, we can estimate μ_0' and, finally, compute the displacement of the region expressed in the *region reference*. This displacement is only valid in a proximity of μ_0^*.

5.2 Tracking Stage

The tracking stage is illustrated in Fig. 2b. At the beginning of the tracking stage, a prediction of the parameters is known and is denoted μ'. The tracking consists of estimating μ such that

$$\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu), t) = \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0^*), t_0),$$

with the notation $I_0(\mathbf{f}(\mathcal{R}; \mu_0^*)) = \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0^*), t_0)$. Time t is removed to simplify the notations.

By computing

$$\delta\mu = \mathbf{A_h}\delta\mathbf{i} = \mathbf{A_h}[\mathbf{I}_0(\mathbf{f}(\mathcal{R}; \mu_0^*)) - \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu'))], \tag{6}$$

we obtain a disturbance $\delta\mu$ that would have produced $\delta\mathbf{i}$ if the parameters vector had been μ_0^*. In that case, a location $\mathbf{x}$ of the region is transformed into $\mathbf{x}' = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{x}; \mu'); \mu_0^*)$, with $\mu' = \mu_0^* + \delta\mu$.

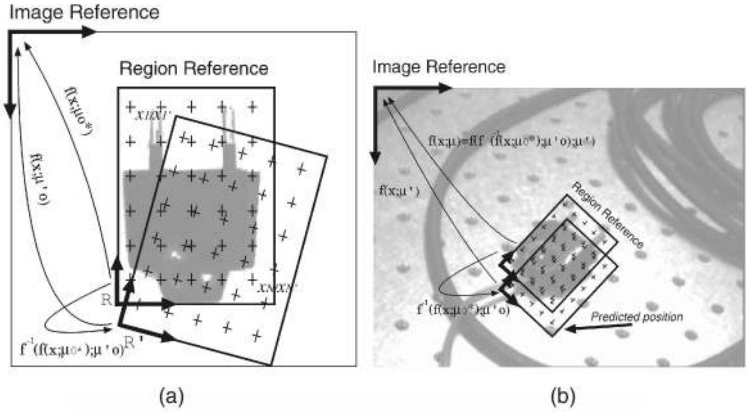

Fig. 2. From learning to tracking. (a) Learning stage and (b) tracking stage.

The actual transformation turns $\mathbf{x}$ into $\mathbf{u} : \mathbf{u} = \mathbf{f}(\mathbf{x}, \mu)$. As described in Fig. 2b, introducing $\mathbf{x}' = \mathbf{f}^{-1}(\mathbf{f}(\mathbf{x}; \mu_0^*); \mu_0')$ in the relation $\mathbf{u}' = \mathbf{f}(\mathbf{x}', \mu')$ gives:

$$\mathbf{u}' = \mathbf{f}(\mathbf{x}'; \mu') = \mathbf{f}(\mathbf{f}^{-1}(\mathbf{f}(\mathbf{x}; \mu_0^*); \mu_0'); \mu'). \tag{7}$$

This equation is fundamental for tracking: It gives the transformation aligning the region on the target at the current time, knowing a prediction μ' and a local disturbance $\delta\mu$. This local disturbance around the initial value μ_0^* is obtained by mapping the current image on the reference region and computing the difference $\delta\mathbf{i} = \mathbf{I}(\mathbf{f}(\mathcal{R}, \mu_0^*), t_0) - \mathbf{I}(\mathbf{f}(\mathcal{R}, \mu'), t)$. Equation (2) gives $\mu_0' = \mu_0^* + \mathbf{A}_h\delta\mathbf{i}$.

The main idea is therefore to correct the transformation of the region in the reference region (acting as if the parameters were μ_0^*) and to transform this correction by applying μ' to it.

5.3 Linear Motion Models

Translation, Rotation, and Scale. In the case of planar translation (tx, ty), planar rotation (θ), and scale (s), the relation between a point of coordinates (x, y) in the reference region and the corresponding point in the image coordinates (u, v) is:

$$\begin{aligned} u &= s\cos(\theta)x - s\sin(\theta) + tx \\ v &= s\sin(\theta) + s\cos(\theta) + ty. \end{aligned}$$

It can be written using matrix products which uses homegenous coordinates. Let $(x, y, 1)^t$ be the coordinates of a point in the *reference region* and $(\lambda u, \lambda v, \lambda)^t$ its coordinates in the image. Then, we have

$$\begin{pmatrix} \lambda u \\ \lambda v \\ \lambda \end{pmatrix} = \mathbf{F}(\mu) \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \; with \; \mathbf{F}(\mu) = \begin{pmatrix} s\cos(\theta) & -s\sin(\theta) & tx \\ s\sin(\theta) & s\cos(\theta) & ty \\ 0 & 0 & 1 \end{pmatrix}.$$

Homography. Using the same notations (homogenous coordinates), homographic motions can be modeled by using an eight parameters model, given by the following matrix:

$$\mathbf{F} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix}.$$

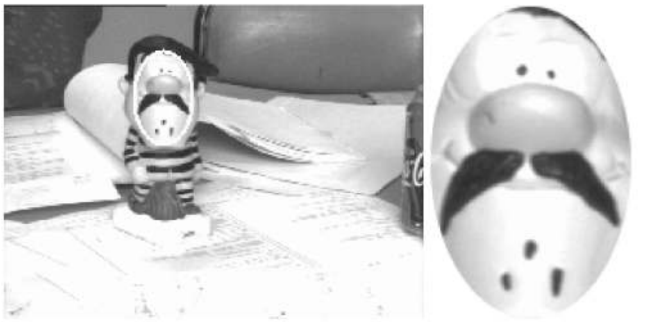

Fig. 3. Image and target region.

General Linear Motion Models. In the case of a linear motion model, we have seen above that function $\mathbf{f}$ can be written as a product of matrices:

$$\mathbf{f}(\mathbf{x}; \mu) = \mathbf{F}(\mu)x,$$

where $\mathbf{x}$ is written with homogeneous coordinates $\mathbf{x} = (sx, sy, s)$ and $\mathbf{F}$ is a 3×3 matrix.

In that case, (7) becomes:

$$\mathbf{F}(\mu) = \mathbf{F}(\mu')\mathbf{F}^{-1}(\mu_0')\mathbf{F}(\mu_0^*), \tag{8}$$

where $\mathbf{F}(\mu_0') = \mathbf{F}(\mu_0^* + \mathbf{A}_h\delta\mathbf{i})$. $\mathbf{F}(\mu')$ is the transformation obtained on the previous image. $\mathbf{F}(\mu_0^*)$ is computed from the selection of the region in the initial image. Matrix $\mathbf{A}_h$ is obtained in the learning stage. Finally, $\delta\mu$ is given by (6).

6 Experiments and Results

The following experiments show the superiority of the hyperplane approximation (HA) over the Jacobian approximation (JA). We have tested three different motion models: planar translation, planar rotation and scale (TRS), affine transformation, and homographic transformation. We only present results concerning planar TRS because the four parameters involved in this transformation can be

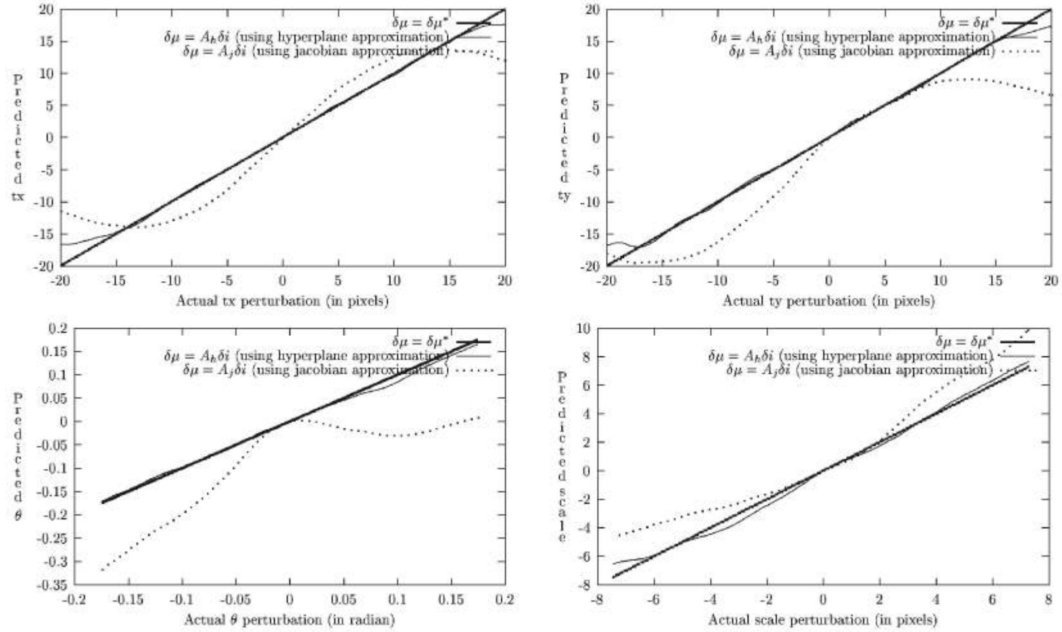

Fig. 4. Predicted motion as a function of actual motion.

intuitively perceived. This is not the case for the six parameters of the affine motion or the height parameters of the homographic motion. Furthermore, we obtain, in each case, the same kind of result.

The algorithms have been implemented on a O2 Silicon Graphics workstation. About 400 points included in an elliptic area are tracked at the frame rate (30hz). The treatments take less than 10 ms in both cases (HA and JA).

6.1 Comparison of JA and HA: Results on a Static Image

This experiment consists of selecting a region in an image, in learning the matrix $\mathbf{A}$ by both methods and, finally, in validating the matrix by moving the region and comparing the ground truth shift with the one estimated by $\delta\mu = \mathbf{A}\delta\mathbf{i}$.

Estimate of the Jacobian matrix and hyperplane estimate must be made in the same conditions, otherwise, this comparison would not have any meaning. Small disturbances around the reference position are produced using a uniform random law. Exactly the same set of disturbances is used to compute the Jacobian and the hyperplane approximation.

We conducted this experiment on several images and obtained the same kind of results. We present results obtained on a single image (refer to Fig. 3).

The graphics in Fig. 4 compares the approximations obtained with HA and JA. In both cases, the linear approximation (learning stage) has been made around the initial transformation, in a range of +/- 20 pixels for translations, +/- 10 degrees for rotation, and +/- 10 pixels for scale. The scale variation is computed as a function of the size of the region in the first image.

We have clearly observed that the approximation given by the HA is far better than the one given by the JA. For example, in the case of a 15 pixels translation, several iterations are necessary to compensate for this translation using JA; one iteration will be enough when using HA.

It means that, in tracking applications, the proposed algorithm will be able to track faster objects.

6.2 Experiments on Moving Objects

Synthetic Results. The image and target given in Fig. 3 are synthetically animated, by rotating the image around its center at a given speed. It will induce a rotation/translation displacement of the object. We measure the maximum speed that JA and HA can accept. JA can accept up to 2.2 degrees/image, while HA can accept up to 7.5 degrees/image.

Real Video Sequences. The algorithms have been implemented on a O2 Silicon Graphics workstation (having a 150Mhz R5000 processor). About 100 points included in a polygonal area are tracked at the frame rate (30Hz). The treatments take less than 10 ms. The motion is supposed to be a homography.

In that case, we also used a coarse to fine strategy; four different levels of approximation are applied in turn. Each of them have been learned distinctly, applying different levels of disturbances. Disturbance amplitudes have been set respectively to 20 percent, 10 percent, 5 percent, and 1 percent of the region size. This multiscale approach has been used to increase accuracy. Coarser scales allows us to inaccurately track large motions; finest scales allows us to accurately track small motions.

The four images presented Fig. 5 illustrate a real-time tracking sequence. The full sequence, the software, and other results are available on the Web.[1] During this sequence, the object is rotated at a speed up to 760 degrees per second (15 degrees per frame). To our knowledge, none of the previously proposed tracking algorithm can reach this speed on this kind of hardware.

[1] http://wwwlasmea/Personnel/Frederic.Jurie/tracking.html.

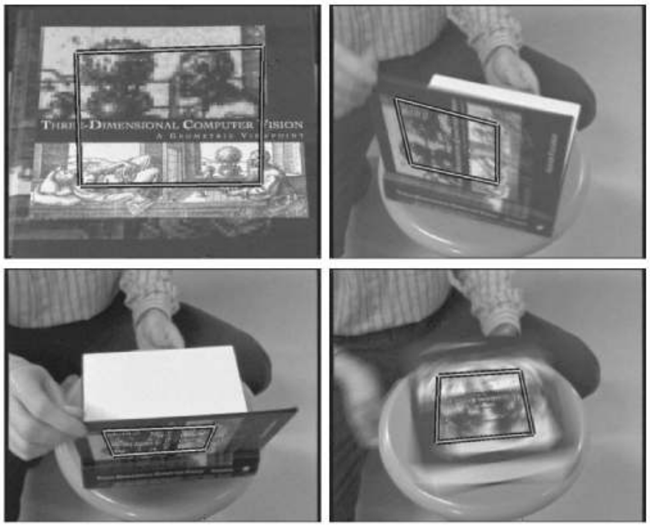

Fig. 5. Real-time tracking.

7 Conclusion

In this article, we have shown an original improvement of the tracking algorithm proposed by Hager and Belhumeur [6]. The key idea is to replace the Jacobian approximation by a hyperplane approximation. This approximation is relatively common in the computer vision community, but its direct use would involve reestimating a large system dynamically. An important contribution is showing how a precomputed approximation can be used dynamically.

With these improvements, the convergence speed is increased by 3 or 4, compared to Hager and Bellhumeur's tracker.

Despite the fact the article is focused on geometric motion, all of the previous results concerning changes in illumination, partial occlusions, or points selection remain valid and can be used directly.

We are actively looking at the problem of 3D tracking from different viewpoints by modeling objects with a set of appearances. The tracker will be able to track 3D objects under any 3D motion.

References

[1] M.J. Black and A.D. Jepson, "Eigentracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," *Int'l J. Computer Vision,* vol. 26, no. 1, pp. 63-84, Jan. 1998.

[2] R. Brunelli and T. Poggio, "Template Matching: Matched Spatial Filter and Beyond," A.I. Memo 1549, Massachusetts Inst. of Technology, Oct. 1995.

[3] T.F. Cootes, G.J. Edwards, and C.J. Taylor, "Active Appearance Models," *Proc. European Conf. Computer Vision,* pp. 484–498, 1998.

[4] T. Darrell, I.A. Essa, and A.P. Pentland, "Task-Specific Gesture Analysis in Real-Time Using Interpolated Views," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 12, pp. 1236–1242, Dec. 1996.

[5] M. Gleicher, "Projective Registration with Difference Decomposition," *Proc. CVPR '97,* pp. 331–337, 1997.

[6] G.D. Hager and P.N. Belhumeur, "Efficient Region Tracking with Parametric Models of Geometry and Illumination," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 10, pp. 1025–1039, Oct. 1998.

[7] M. La Cascia, S. Sclaroff, and V. Athitsos, "Fast, Reliable Head Tracking under Varying Illumination: An Approach Based on Registration of Textured-Mapped 3D Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 4, pp. 322–336, Apr. 2000.