A simple and efficient template matching algorithm

Frédéric Jurie, Michel Dhome

HAL Id: inria-00548281
https://hal.inria.fr/inria-00548281

Submitted on 21 Dec 2010

A simple and efficient template matching algorithm.

Frédéric Jurie and Michel Dhome

LASMEA - UMR 6602 of CNRS, Blaise-Pascal University, F-63177 Aubière - FRANCE

Abstract

We propose a general framework for object tracking in video images. It consists in low-order parametric models for the image motion of a target region. These models are used to predict the movement and to track the target. The difference of intensity between the pixels belonging to the current region and the pixels of the selected target (learnt during an off-line stage) allows a straightforward prediction of the region position in the current image.

The proposed algorithm allows to track in real time (less than 10ms) any planar textured target under homographic motions. This algorithm is very simple (a few lines of code) and very efficient (less than 10 ms on a 150Mhz hardware).

1. Introduction

In traditional tracking approaches there are two major groups, either the tracking is performed in the image (the pattern is directly tracked) or in the pose space (tracking of the object's pose).

The first approach relies on such techniques as normalized correlation or template matching. Darell *et al.* [7, 6], Brunelli *et al.* [3] propose to maximize 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 sub-spaces of the initial image representation [18, 14, 15]. The main limitation of these approaches is their lack of resistance with regard to occlusions. Black and Jepson [2] 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.

The other group of tracking techniques are those based on the computation of the object's current pose. It involves 3D models of objects, by means of 3D feature sets. These features can be points [9], line segments [5, 11], edges [10], or regions [16]. With these techniques it is possible to localize the object [12] in the current image and to predict the feature positions in the subsequent images, according to a motion model [19, 20] and an uncertainty model [13]. We specially note the work of Strom *et al.* [17] and Basu *et al.* [1]. They describe a real-time system for tracking and 3D modeling. The main idea is to select a dense set of feature points. They are matched against the incoming video to update the pose of the 3D model. A generic 3D polygonal object model is required (head or lips in their case). Pose search techniques are naturally less sensitive to occlusions, as they are based on local correspondences. If several correspondences are missing the pose is still computable.

Our work is related to the first of these two groups, as we are interested in the tracking of templates in video sequences. Recently, various successful approaches for object tracking have been proposed. They aim to track complex objects in real time, in realistic situations (occlusions, changes of illumination). They deal either on how to precisely predict the position of the template (or the relative position of the camera) by using motion models and predictive filtering, or on the way to find the actual position of the template by exploring a neighborhood of the prediction. Unlike the first problem, the second is generally computationally expensive, as it involves a search in the image or in the pose space, taking into account object occlusions if necessary. Black and Jepson [2] give a good example of how this search can be carried out. They proposed to use an optimization algorithm which simultaneously find the object's position and reconstruct occluded parts. This optimization is unfortunately slow.

We proposed to use the following framework, illustrated in Figure 1. 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. By comparing the grey level values of the target template with the grey level values of the predicted region, is is possible to obtain the actual position of the template in the current image. This computation is possible because – during an off-line training stage – a relation between the variations of intensities and the variations of position has been learnt,
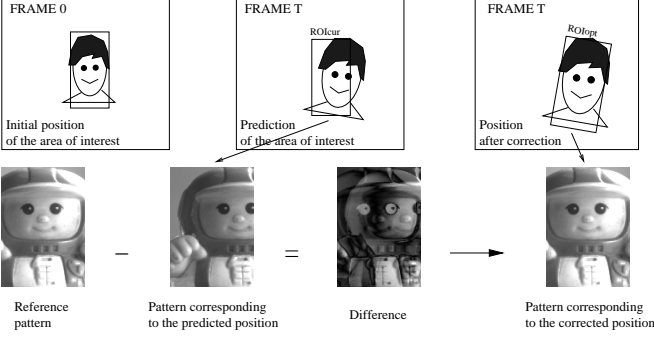
Figure 1. Illustration of the tracking principle.

using an hyperplane modelization.

This article is made of three sections. The first one - section 2 - is devoted to a short presentation of the bases of tracking framework, while the proposed approach as well as the correspondinf algorithms are given in section 3. Finally, in section 4, our approach is discussed and compared to relative works.

2. Region tracking

2.1. Formulation

Let[1] $I(\mathbf{x}, t)$ the brightness value at the location $\mathbf{x} = (x, y)$ in an image acquired at time t. Let the set $\mathcal{R} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ 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. 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 ($t = 0$). At this time, the position of the template is μ_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 $\mu(t) = (\mu_1(t), \mu_2(t), \ldots, \mu_n(t))$ 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 $(\mathbf{f}(\mathbf{x}_1; \mu(t)), \mathbf{f}(\mathbf{x}_2; \mu(t)), \ldots, \mathbf{f}(\mathbf{x}_N; \mu(\mathbf{t})))$ is denoted $\mathbf{f}(\mathcal{R}; \mu(t))$

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 estimation of the ground truth value $\mu^*(t)$.

---

[1]Bold fonts denote vectors and matrices.

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

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

This very general formulation of tracking have been used by several authors. Black and Jepson [2] give a good example of how this minimization can be carried out. They proposed to use an optimization algorithm (Levenberg-Marquard), which is unfortunately slow and can only tolerate very small movements of the object.

A very straightforward and efficient computation of $\mu(t + \tau)$ can be obtained by writing :

$$\begin{aligned} \mu(t + \tau) &= \mu(t) + \mathbf{A}(t + \tau)(\mathbf{I}(\mathbf{f}(\mathcal{R}; \mu_0^*), t_0) \\ &\quad - \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu(t)), t + \tau)), \end{aligned} \quad (1)$$

if matrix $\mathbf{A}(t + \tau)$ can be obtained with small on-line computation. τ denotes the time between 2 successive images. 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 $\delta\mu(t + \tau) = \mu(t + \tau) - \mu(t)$, equation (1) can be written:

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

2.2. Hyperplane approximation

Equation (2) can be seen as the modellization by n hyperplanes. In this section, time is suppressed in order to obtain simpler notations. Equation (2) can be rewritten :

$$\begin{cases} 0 = (a_{11}, \ldots, a_{1N}, -1)(\delta i_1, \ldots, \delta i_N, \delta\mu_1)^T \\ 0 = (a_{i1}, \ldots, a_{iN}, -1)(\delta i_1, \ldots, \delta i_N, \delta\mu_i)^T \\ 0 = (a_{n1}, \ldots, a_{nN}, -1)(\delta i_1, \ldots, \ldots, \delta i_N, \delta\mu_n)^T \end{cases}.$$

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

To learn the matrix $\mathbf{A}$, suppose that the current position of the region of interest is characterized by the vector μ. If this vector is perturbated such that $\mu' = \mu + \delta\mu$, the region of interest is moved and the vector $\delta\mathbf{i} = \mathbf{I}(\mathcal{R}, \mu) - \mathbf{I}(\mathcal{R}, \mu')$ is computed. This "perturbation" procedure is repeated N_p times, with $N_p > N$.

At the end, we have collected N_p couples $(\delta\mathbf{i}^k, \delta\mu^k)$. It is then possible to obtain a matrix $\mathbf{A}$ such $\sum_{k=1}^{k=N_p}(\mu^k - \mathbf{A}\delta\mathbf{i}^k)^2$ is minimal. By writing $\mathbf{H} = (\delta\mathbf{i}^1, \ldots, \delta\mathbf{i}^{N_p})$ and $\mathbf{Y} = (\delta\mu^1, \ldots, \delta\mu^{N_p})$, $\mathbf{A}$ can be obtained by

$$\mathbf{A} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}. \quad (3)$$

A similar scheme has already been proposed by Cootes *et. al.* [4], in a different context (they have used it to dynamically estimate the parameters of a face appearance model).

3. Efficient tracking with image hyperplanes

The Hyperplane Approximation scheme is very inefficient, taken under its initial form. The direct computation of the matrix $\mathbf{A}$ involves a least square minimization, which is to be repeated at each new image. The matrix depends on the current position, orientation, etc. given by μ. The learning stage consists in computing a linear relation between a set of grey level differences and a correction of the parameters μ. If this relationship is computed around the initial position μ_0^* – known when the user select an image region – the obtained matrix $\mathbf{A}$ is not valid for other values of μ.

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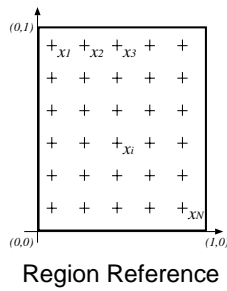

Figure 2. The region reference.

Learning stage. Let the region be defined as $\mathcal{R} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N})$ the set of N point locations in a local reference called the *region reference* (see Fig. 2). The function $\mathbf{f}(\mathbf{x}; \mu)$ changes the coordinates of $\mathbf{x} = (x, y)$ in the region reference into $\mathbf{u} = (u, v) = \mathbf{f}(\mathbf{x}; \mu)$ in the image reference.

When the user defines a target region in the reference image, he defines a set of correspondences between points of the *region reference* and points of the *image reference* (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 *region reference* on the target (defined in the *image reference*) is possible.

The learning stage consists in producing small perturbations (shifts) on μ_0^* using a function written $\mathcal{P}$. We write $\mu_0' = \mathcal{P}(\mu_0^*)$ a perturbation of μ_0^*. This perturbation produces 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 perturbations $\delta\mu = \mathcal{P}(\mu_0^*) - \mu_0^*$ are produced in order to obtain the linear model giving $\delta\mu = \mathbf{A}\delta\mathbf{i}$. Therefore, knowing $\delta\mathbf{i}$ we will be able to estimate $\mathcal{P}(\mu_0^*)$.

As shown on the figure 3, if $\mathbf{u}'$ are the coordinates of $\mathbf{x}$ in the image reference under the transformation $\mu_0' = \mathcal{P}(\mu_0^*)$ 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 $\mathcal{P}(\mu_0^*) = \mu_0'$ and finally compute the displacement of the region expressed in the *region reference*. This displacement is only valid in a neighborhood of μ_0^*.
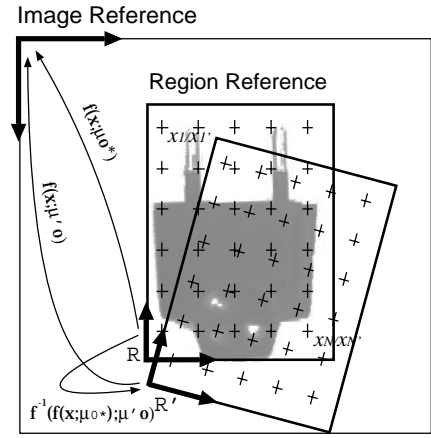

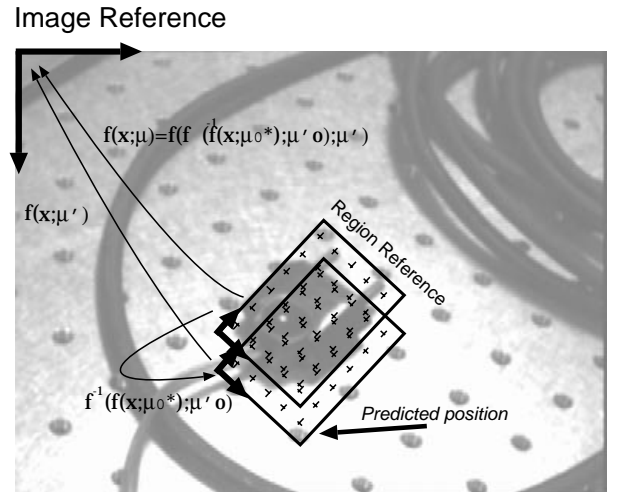**Figure 3. Learning stage.**

Figure 4. Tracking stage.

Tracking stage. The tracking stage is illustrated in Figure 4. At the beginning of the tracking stage, a prediction of the parameters is known and is denoted μ'. The tracking consists in 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 expressions.

By computing

$$\delta\mathbf{i} = \mathbf{I}_0(\mathbf{f}(\mathcal{R}; \mu_0^*)) - \mathbf{I}(\mathbf{f}(\mathcal{R}; \mu')) \tag{4}$$

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

We are lookink for the actual transformation transforming $\mathbf{x}$ in $\mathbf{u} = \mathbf{f}(\mathbf{x},\mu)$. As illustrated in Figure 4, 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 :

$$\mathbf{f}(\mathbf{x};\mu) = \mathbf{f}(\mathbf{f}^{-1}(\mathbf{f}(\mathbf{x};\mu_0^*);\mu_0');\mu') \qquad (5)$$

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 perturbation μ_0'. This local perturbation around the initial value μ_0^* is obtain by mapping the current image on the region reference 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 $\mu_0' = \mu_0^* + \mathbf{A}\delta\mathbf{i}$.

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

3.1. Motion models

Equation (5) can be simplified when affine transforms are used.

General affine motion. Let us assume that $\mathbf{f}(\mathbf{x};\mu)$ is affine in $\mathbf{x}$. Then we have $\mathbf{f}(\mathbf{x};\mu) = \mathbf{F}(\mu)x$ where $\mathbf{x}$ is written with homogeneous coordinates $\mathbf{x} = (sx,sy,s)$, and $\mathbf{F}$ is a 3×3 matrix.

In that case, equation (5) becomes :

$$\mathbf{F}(\mu) = \mathbf{F}(\mu')\mathbf{F}^{-1}(\mu_0')\mathbf{F}(\mu_0^*) \qquad (6)$$

where $\mathbf{F}(\mu_0') = \mathbf{F}(\mu_0^* + \mathbf{A}\delta\mathbf{i})$. $\mathbf{F}(\mu')$ is the transformation obtained on the previous image. $\mathbf{F}(\mu_0^*)$ is computed from the selection of the region in the initial image. The matrix $\mathbf{A}$ is obtained in the learning stage. Finally, $\delta\mathbf{i}$ is given by equation (4).

Similarity motion. In the case of planar translation, planar rotation and scale, the matrix $\mathbf{F}$ becomes

$$\mathbf{F} = \begin{pmatrix} s\mathbf{R}(\theta) & \mathbf{t} \\ 0 & 1 \end{pmatrix}$$

where $\mathbf{R}(\theta)$ is a 2×2 rotation matrix, s a scaling factor and $\mathbf{t} = (tx,ty)$ a 2D translation.

Homography. Objects views under homography (planar 3D objects) can be modeled by using a eight parameters model, given by the following matrix

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

3.2 Pseudo-codes

Learning stage. Let `F0` denotes the matrix corresponding to the initial position of the pattern in the first image of the sequence (initial transformation). The learning stage consists in shifting the region of interest slightly from the initial position and in relating the variations of intensity with the shifts. `Nbexp` shifts are randomly performed, `FRandom(ie)` denotes the perturbation (shifts) applied at iteration `ie`. Two matrices are constructed (`M` and `I`), one storing the perturbations and the other the variations of grey level values. The matrix `A` which is the best approximation of `M=A*I` is computing using a least square approximation. The function `Sample` return the grey level values of the current image, given a set of locations.

```
M=[], I=[]
Vref = Sample(Image0,F0,R)
FOR (ie=0 TO NBexp BY 1) {
      Fmod = F0 * FRandom(ie)
      Vcur = Sample(Image,Fmod,R)
      DI = Vcur-Vref
      M = [M ,Fmod]
      I = [I ,DI ]
}
A = (I^t*I)^-1*I^t*M
```

Tracking stage. Let `F0` denotes the matrix corresponding to the initial position of the pattern in the first image of the sequence, `F` the position of the pattern in the previous image, `Vref` the grey level values of the pattern in the first image, and `R` the image locations where the grey level values are taken. With these notations, the tracking pseudo-code is :

```
DO {
    Vcur = Sample(Image,F,$\cal R$)
    DI = Vcur-Vref
    inv_Fmod = inv(A*DI)
    F = F*inv_Fmod*F0
}
```

The algorithm consists only in computing the difference `DI` between the grey level values of the reference pattern with the grey level values of the current image at the previous position. This difference allows to compute the actual position of the template, corresponding to the transformation `F`.

The implementation of the tracking algorithm (as well as the learning algorithm) involves only a few lines of code. The tracking consists in a hundred of subtractions (to compute `DI`), a few hundred of multiplications (to compute `Fmod`), a matrix inversion (a 8X8 matrix in case of homography), and two matrix products.
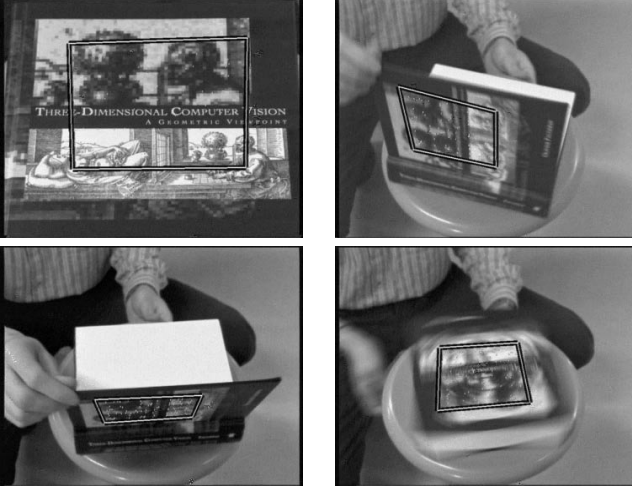
Figure 5. Real time tracking.

3.3. Experiments and Results

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 (25Hz). The treatments take less than 10 ms. The motion is supposed to be an homography.

The four images presented Fig. 5 illustrate a real time tracking sequence. During this sequence the object is rotated at a speed up to 760 degrees per sec. (15 deg. per frame). From our knowledge, none of the previously proposed tracking algorithm can reach this speed on this kind of hardware.

4. Comparison with related works

Hager *et al.* in [8] propose a similar approach and estimate the matrix $\mathbf{A}$ in equation (2) by using the inverse of an image jacobian.

Principle of Hager's approach. Equation (2) shows clearly that $\mathbf{A}(t + \tau)$ can plays the role of a jacobian matrix. For this reason, we will note it $\mathbf{A_j}(t + \tau)$. The estimation of $\mathbf{A_j}(t + \tau)$ can be obtained, as proposed by Hager *et al.* [8] by using the image jacobian.

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,

$$\begin{aligned} \mathbf{I}(\mu + \delta\mu, t + \tau) = \quad & \mathbf{I}(\mu, t) + \delta\mu \mathbf{I}_\mu(\mu, t) \\ + \quad & \tau \mathbf{I}_t(\mu, t) + h.o.t. \end{aligned} \quad (7)$$

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

By writing

$$\mathbf{A_j}(t) = (\mathbf{M}^T(\mu, t)\mathbf{M}(\mu, t))^{-1}\mathbf{M}^T(\mu, t), \quad (8)$$

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

By combining equations (2) and (8) we obtain :

$$\delta\mu = (\mathbf{M}^T(\mu, t)\mathbf{M}(\mu, t))^{-1}\mathbf{M}^T(\mu, t)\delta\mathbf{i} \quad (9)$$

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 have 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 to obtain $\delta\mu = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$ with only little on-line computations [8].

Equation (9) involves the computation of the difference of 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 that case, eq.(9) 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 in evaluating $\delta\mathbf{i}(t + \tau)$ and consequently obtaining $\delta\mu(t + \tau)$, and finally update $\mu(t + \tau)$ according to the equation : $\mu(t + \tau) = \mu(t) + \delta\mu(t + \tau)$.

Comparison This framework involves approximately the same amount of online computation than the approach proposed in this article.

However we experimentally observed that our approach has a convergence area larger than the one obtained by the Hager's one using an image jacobian approximation.

We present here a result which is representative of experiments we have performed. This result have been obtained by using the pattern shown figure 6. The motion considered for that example is a 4 parameters planar motion (planar translations, planar rotation and scaling). The pattern has been selected manually by selecting an area of interest. About 100 points are randomly selected in the area of interest. The same set of points is used for both method. During the learning stage the same shift amplitudes are used to learn the image jacobian and to learn the hyperplane approximation. They consist in translations up to +/-25% of the width of the target in horizontal translation, +/-16% of the target height in vertical translation, +/-0.2 radian in rotation and from 0.8 to 1.2 in scaling. Once the learning stage have been completed, we have compared the two approaches by moving the area of interest away from the pattern and by measuring how good is the correction given by the two approaches. The evaluation is done by computing the distance of the area of interest from the correct position after one iteration of each tracker. This distance is the sum
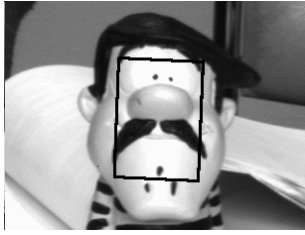
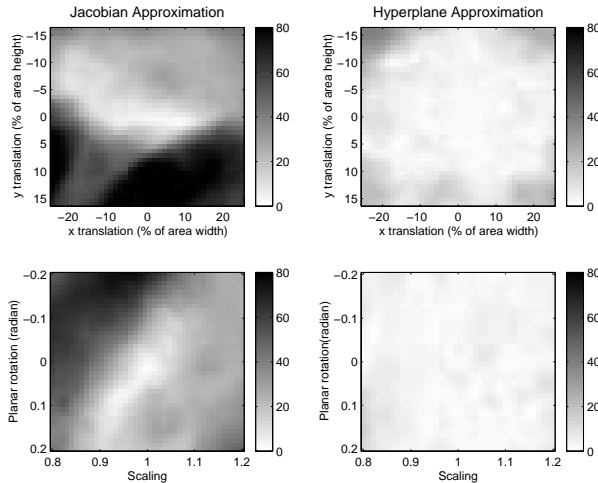**Figure 6. The pattern used for comparisons.**

Figure 7. Comparison of both approaches.

of square distances of each one of the four corners of the rectangular area of interest. A low value means the tracker can compensate for the perturbation, a high value means it cannot track correctly the pattern for such shift of the template.

By studying the Fig. 7 we can clearly notice that the tracker obtained by using the Hager's approach can tolerate only small amount of motion between two images, compare to the approach proposed in this article.

5. Conclusion

In this article, we have shown an original framework for tracking textured templates in real time. The key idea is to use an hyperplane approximation to relate the variations of intensity in an area of interest to the motion parameters.

This kind of linear approximation is relatively common in the computer vision community, but its direct use would involve to re-estimate dynamically a large system. An important contribution is to show how a precomputed approximation can be used dynamically.

As it has been explained in the article, the tracking consists only in a few hundred of subtractions and multiplications, taking less than 10ms on a 150Mhz workstation.

With these improvement, the convergence area is increased by 3 or 4, compared to the Hager *et al.*'s tracker, which is the only approach directly related to the proposed framework.

Despite the fact the article is focussed 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 tracking 3D non planar objects under different viewpoints, by modeling objects with a set of appearances.

References

[1] S. Basu and A. Pentland. A three-dimensional model of human lip motions trained from video. Technical Report 441, M.I.T. Media Laboratory Perceptual Computing Section, 1999.

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

[3] R. Brunelli and Poggio T. Template matching : Matched spatial filter and beyond. A.I. Memo 1549, M.I.T., October 1995.

[4] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. In *Proc. ECCV98*, pages 484–498, Freiburg, Germany, 1998.

[5] J.L Crowley, P. Stelmaszyl, and C. Discours. Measuring image flow by tracking edge-lines. In *Proc. IEEE CVPR Recognition*, pages 658–664, San Diego, California, 1989.

[6] T. Darrell, I.A. Essa, and Pentland A.P. Task-specific gesture analysis in real-time using interpolated views. *IEEE PAMI*, 18(12):1236–1242, 1996.

[7] T.J. Darrell and A.P. Pentland. Space-time gestures. In *CVPR93*, pages 335–340, 1993.

[8] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE PAMI*, 20(10):1025–1039, October 1998.

[9] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, Mancherster, 1988.

[10] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. ECCV96*, volume I, pages 343–356, Cambridge, UK, 1996.

[11] F. Jurie. Solution of the simultaneous pose and correspondence problem using gaussian error model. *CVIU*, pages 357–373, 1999.

[12] H. Kollnig and H.H. Nagel. 3d pose estimation by directly matching polyhedral models to gray value gradients. *IJCV*, 23(3):283–302, 1997.

[13] D.G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *IJCV*, 8(2):113–122, 1992.

[14] B. Moghaddam and A. Pentland. A subspace method for maximum likelihood taget detection. Technical report, M.I.T. Media Laboratory Perceptual Computing Section, 1995.

[15] H. Murase and S.K. Nayar. Visual learning and recognition od 3d object from appearance. *IJCV*, 18(14):5–24, 1995.

[16] F. Pla and J. Marchant. Matching feature points in image sequences through a region based method. *CVIU*, 66(3):271–285, 1997.

[17] J. Strom, T. Jebara, S. Basu, and A. Pentland. Real time tracking and modeling of faces : An ekf-based analysis by synthesis approach. Technical Report 506, M.I.T. Media Laboratory Perceptual Computing Section, 1999.

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

[19] J.J Wu, R.E. Rink, T.M. Caelli, and V.G. Gourishankar. Recovery of the 3-d location and motion of a rigid object through camera image. *IJCV*, 3:373–394, 1989.

[20] G.S. Young and R. Chellappa. 3d motion estimation using a sequence of noisy stereo images. In *IEEE CVPRn*, pages 710–716, Ann Arbor, Michigan, 1988.