Real Time 3D Template Matching

Frédéric Jurie, Michel Dhome

Real time 3D template matching

Frédéric Jurie and Michel Dhome
LASMEA - CNRS UMR 6602 - Université Blaise Pascal - 63177 Aubière cedex - France

Abstract

One of the most popular methods to extract useful informations from an image sequence is the template matching approach. In this well known method the tracking of a certain feature or target over time is based on the comparison of the content of each image with a sample template. In this article, we propose a 3D template matching algorithm that is able to track target corresponding to the projection of 3D surfaces. With only a few hundred of subtractions and multiplications per frame, our algorithm provides, in real time, an estimation of the 3D surface pose. The key-idea is to compute the difference between the current image content and the visual aspect of the target under the predicted spatial attitude. This difference image is converted into corrections on the 3D location parameters.

1 Introduction

Three-dimensional object tracking is a major task for numerous computer vision applications. Two major categories of approaches are generally distinguished. *Feature-based* approaches uses local features like points, line segments, edges, or regions. With these techniques it is possible to localize the object in the current image and to predict the feature positions in subsequent ones, according to a motion model and an uncertainty model. 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.

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 in 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 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 non linear optimization problem, using Newton type or Levenberg-Marquardt based algorithms.

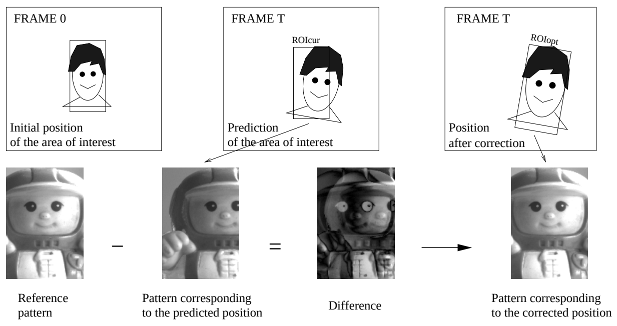
FRAME 0
Initial position of the area of interest
FRAME T
ROIcur
Prediction of the area of interest
FRAME T
ROIopt
Position after correction
Reference pattern
Pattern corresponding to the predicted position
Difference
Pattern corresponding to the corrected position

Figure 1: Principle of the difference image based approach.

Darell *et al.* [4], Brunelli *et al.* [2] 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. 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 have been proposed: the tracking problem is posed as the problem of finding the best (in least squares sense) set of parameter values describing the motion and deformation of the target through the sequence. In this case, parameter variations are written as a linear function of a difference image (the difference between the reference image and the current image). It is illustrated on Figure 1. This approach is very efficient as motion can be easily deduced from difference image. Cootes, Edwards and Taylor [3] use it to dynamically estimate the parameters of a face appearance model (2D model). Hager and Belhumeur [7] include it in a general framework for object tracking, under planar affine motions. Only a few works use this approach with projective transformations [6, 9, 8], because projective transformations are highly non-linear and because of the size of the parameter space.

This article proposes an efficient solution to the problem of SSD tracking of 3D surfaces.

This article is made of four sections. In the first one, the problem of tracking 3D objects is posed and a mathematical formulation is given. In the second one, specific problems related to 3D geometry are addressed. In the next section, some experimental results are given. At last, the proposed approach is discussed and compared to previous approaches.

2 3D template matching

2.1 3D view point dependency

The presented approach belongs to the lastly presented class of tracking methods. We want to focus this article on the major problem occurring during the tracking of 3D objects observed under perspective projection: the view point dependency. Other subjects like sensitivity to illumination have been widely treated and will not be discussed here.

The principle of the proposed technique is based on two steps. First, during an off-line stage, an *interaction matrix* is estimated. This matrix correspond to the first order approximation of the relationship linking difference image and position variation. Such a linearization have been proposed by sevral authors; the form of this matrix can vary from an author to another. Although this matrix is learned in the neighborhood of one target reference position, many authors [7, 6, 9, 8] show how it is possible to extend efficiently its validity. Second, during the tracking stage, the difference between the current image and the predicted one is computed and multiplied by the interaction matrix to obtain the correction to be applied to the target position parameters in order to align it on the current image.

At our knowledge La Cascia *et al.* [9] are the only authors using this kind of technique to track directly in 3D, the motion of a tridimensional object. In their formulation, they assume that the difference image measured in one image during the tracking stage is the consequence of a relative 3D displacement of the target. The difference image induced by a variation of the 3D localization parameters is unfortunately strongly dependent on the view point.

This phenomena is illustrated Figure 2. The upper part shows an image of a 3D textured object as well as the projection of its model edges. Three particular points of the front face of this object are materialized by squares. The two images below show effects of the same relative perturbation (1 cm in translation and 5 degrees in rotation between the textured object and its CAD model) observed from two different points of view. We can note that the perspective projection of the three materialized points do not correspond to the same textured patterns. Consequently, it is not possible to directly use the difference image to track the 3D motion of an object.

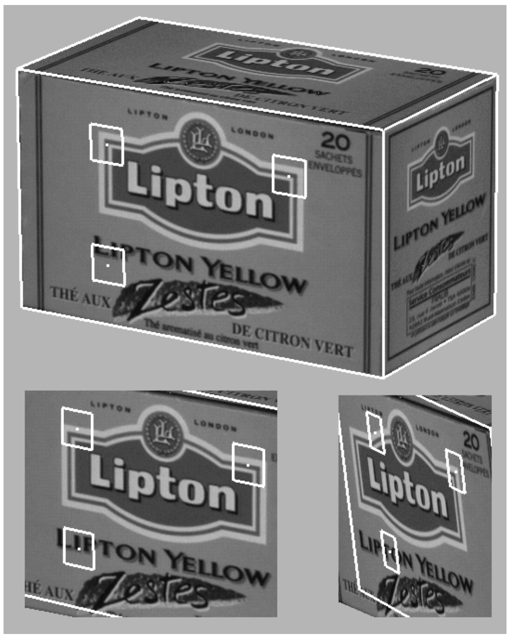

Figure 2: Difference image dependents on the view point.

The proposed approach is based on the following steps. The interaction matrix used to compensate for local perturbations of the 3D object location around a reference position is learned off-line during a learning stage. During the tracking stage, the difference image (difference between the reference pattern and the displaced one) is computed by sampling the current image at the points corresponding to perspective projection of the displaced target surface. Assuming the current different image have been met during the learning stage, the interaction matrix is only used to compute – in the neighborhood of the reference position– the 3D points of the surface where the textured signal has been sampled. Knowing a set of correspondences between 2D locations (where the current image has been sampled) and 3D points, it is possible to compute the current attitude of the target by using a pose estimation algorithm. This approach is valid even is the view point is far from the reference position.

2.2 Tracking from differences

Let us first recall the principle of tracking from differences. Let[1] $I(\mathbf{x}, t)$ the brightness value at the location $\mathbf{x} = (x, y)$ in an image acquired at time t. Let $\mathcal{R} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ the set of N image locations which define a *target region*. $\mathbf{I}(\mathcal{R}, t) = (I(\mathbf{x}_1, t), I(\mathbf{x}_2, t), \ldots, I(\mathbf{x}_N, t))$ is a vector of the brightness values of the target region. We refer to

[1] Bold fonts denote vectors and matrices.

$\mathbf{I}(\mathcal{R}, t_0)$ as the *reference template*. It is the template which is to be tracked; t_0 is the initial time ($t = 0$). These points are the projections of a set of 3D points $\mathcal{RO} = (X_1, \dots X_N)$ belonging to an object surface.

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*. In [8] we have proposed a general motion model allowing any kind of planar transformations. In the present article, we will only focus our attention on 3D motion viewed under perspective projections.

Let $\mathbf{X} = (X, Y, Z)$ the coordinates of a point in the 3D object-centered coordinate system, and $\mathbf{x} = (x, y)$ its projection in the image. The 3D rotation, translation and perspective projection can be written with the standard homogeneous transform formalism :

$$\mathbf{x} = \mathbf{PT}(t)\mathbf{R}_z(t)\mathbf{R}_y(t)\mathbf{R}_x(t)\mathbf{X}$$

where $\mathbf{T}$ is the translational matrix, $\mathbf{R_x}, \mathbf{R_y}, \mathbf{R_z}$ the three elementary rotations parametrized by the Euler angles, and $\mathbf{P}$ the perspective projection matrix depending on the focal length and the position of the principal point (intersection of the optical axis of the camera and the image plane). In that case, we assume $\mathbf{x}$ and $\mathbf{X}$ to be written with homogeneous coordinates. By writing $\mathbf{M}(\mu(t)) = \mathbf{PT}(t)\mathbf{R}_z(t)\mathbf{R}_y(t)\mathbf{R}_x(t)$ the previous equation becomes:

$$\mathbf{x} = \mathbf{M}(\mu(t))\mathbf{X}$$

where $\mu(t) = (\mu_1(t), \dots, \mu_6(t))$ is the set of parameters included in $\mathbf{M}$, depending on the relative position between the object and the camera. There are 6 parameters: 3 translational components and the 3 Euler angles. We assume $N > 6$ and we also assume that $M(\mu(t))$ is differentiable in μ. We call μ the motion parameter vector. At time t_0, the object position is known and parametrized by μ_0. The set of N image locations corresponding to the 3D points on the surface target are $\mathcal{RO}$ and their projections at time t are $\mathbf{M}(\mu(t))\mathcal{RO}$. With these notations, "tracking the object at time t" means "compute" $\mu(t)$ such that

$$\mathbf{I}(\mathbf{M}(\mu(t))\mathcal{RO}, t) = \mathbf{I}(\mathbf{M}(\mu_0)\mathcal{RO}, t_0).$$

We note $\mu(t)$ the estimation of the ground truth value $\mu^*(t)$. The ground truth value, at time t_0, is supposed to be μ_0. The motion parameter vector of the target surface $\mu(t)$ can be estimated by minimizing the following function:

$$O(\mu(t)) = \| \mathbf{I}(\mathbf{M}(\mu(t))\mathcal{RO}, t) - \mathbf{I}(\mathbf{M}(\mu_0)\mathcal{RO}, t_0) \|$$

This very general formulation of tracking have been used by several authors [1, 6, 7, 9]. Nevertheless, a very straightforward and efficient computation of the actualization of $\mu(t)$ can be obtained by writing:

$$\begin{aligned}\mu(t+\tau) &= \mu(t) + \mathbf{A}(t+\tau)[\mathbf{I}(\mathbf{M}(\mu_0)\mathcal{RO}, t_0) \\ &\quad -\mathbf{I}(\mathbf{M}(\mu(t))\mathcal{RO}, t+\tau)] \qquad (1)\end{aligned}$$

where τ denotes the time between two successive images. We will see later how the matrix $\mathbf{A}(t+\tau)$ can be obtained. If we write

$$\delta\mathbf{i}(t+\tau) = \mathbf{I}(\mathbf{M}(\mu_0)\mathcal{RO}, t_0) - \mathbf{I}(\mathbf{M}(\mu(t))\mathcal{RO}, t+\tau)$$

and

$$\delta\mu(t+\tau) = \mu(t+\tau) - \mu(t),$$

equation (1) can be written :

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

2.3 Hyper-plane approximation

Equation (2) can be seen as the equations of 6 hyper-planes. In this section, time is suppressed in order to obtain simpler notations. Equation (2) can be rewritten:

$$\left\{\begin{array}{lcl} 0 &=& (a_{11}, \dots, a_{1N}, -1)(\delta i_1, \dots, \delta i_N, \delta\mu_1)^T \\ 0 &=& \cdots \\ 0 &=& (a_{61}, \dots, a_{6N}, -1)(\delta i_1, \dots, \delta i_N, \delta\mu_6)^T \end{array}\right.$$

Under this form, we can clearly observe that $a_{i1}, \dots, a_{iN}$ are the coefficients of 6 hyper-planes that can be estimated by using a least square estimation.

To learn the matrix $\mathbf{A}$, suppose that the current position μ_0 of the region of interest in the first image is known. If this position is perturbed such that $\mu_0' = \mu_0 + \delta\mu$, the template is moved and the vector $\delta\mathbf{i} = \mathbf{I}(\mathbf{M}(\mu_0)\mathcal{RO}) - \mathbf{I}(\mathbf{M}(\mu_0')\mathcal{RO})$ can be 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 the matrix $\mathbf{A}$ such $\sum_{k=1}^{k=N_p}(\delta\mu^k - \mathbf{A}\delta\mathbf{i^k})^2$ is minimal. By writing $\mathbf{H} = (\delta\mathbf{i}^1, \dots \delta\mathbf{i}^{\mathbf{N_p}})$ and $\mathbf{Y} = (\delta\mu^1, \dots, \delta\mu^{N_p})$, $\mathbf{A}$ can be obtained by computing

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

The computation of matrix $\mathbf{A}$ is performed off-line (during a training stage).

3 Efficient 3D template matching

Matrix $\mathbf{A}$ should be recomputed for any other position μ: as explained Figure 2, in case of 3D rotation, pattern variation is depending on the relative position between object and camera. In the situation presented on Figure 3, the pattern is stretched by the rotation when viewed from position μ_0 and is shrunk from position μ. We are going to see in the following sections, how is it possible to use Equation 2 without recomputing the matrix A.

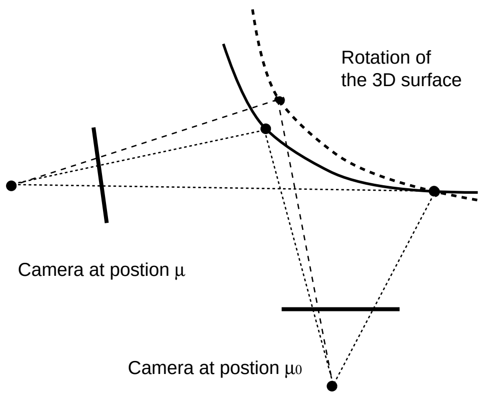

Figure 3: In case of 3D rotation, pattern variation is depending on the camera position.

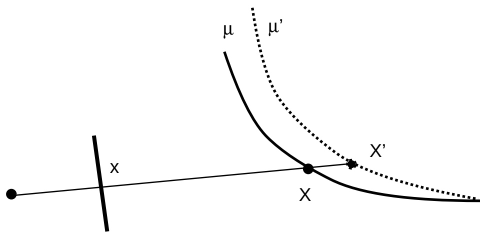

Figure 4: Geometric aspects

3.1 Geometric aspects

During the tracking, the difference $\delta\mathbf{i}$ between the reference template and the observed pattern is due to the variation $\delta\mu$ between the current estimation of the 3D object position μ and the real position μ^*. The grey level measured at the position $\mathbf{x}$ is not the projection of the 3D point $\mathbf{X}$ such $\mathbf{x} = \mathbf{M}(\mu)\mathbf{X}$ but the projection of the point $\mathbf{X}'$ with $\mathbf{x} = \mathbf{M}(\mu^*)\mathbf{X}'$. This is illustrated Figure 4. Nevertheless, if $\mathbf{X}'$ were known, one could easily compute μ^* using a localization algorithm (see for example [5]), as a set of 2D/3D correspondences would be known.

The difference image $\delta\mathbf{i}$ observed during the tracking is due to a rigid motion of the object. Under the hypothesis that the learning stage is relevant (it means that the current difference have been learned), we will show that it is easy to compute X'. In the neighborhood of the reference position used during the learning stage, the corresponding 3D motion can be estimated by using the relationship given equation (2) : $\delta\mu = \mathbf{A}\delta\mathbf{i}$. In that particular case, we have

$$\mathbf{x} = \mathbf{M}(\mu_0)\mathbf{X} = \mathbf{M}(\mu_0 + \delta\mu)\mathbf{X}'.$$

Consequently, $\mathbf{X}'$ can be estimated by computing the intersection of the line $O\mathbf{x}$ (where O is the optical center of the camera) with the object surface localized by the parameters $\mu_0 + \delta\mu$.

When the object is at the position μ the correction $\delta\mu = \mathbf{A}\delta\mathbf{i}$, which is not directly applicable, allows to compute the 3D coordinates of the point $\mathbf{X}'$ that is substituted for the point $\mathbf{X}$ (projected in $\mathbf{x}$). The tracking process therefore consists in the following steps :

1. compute the difference image $\delta\mathbf{i}$ using the predicted position μ,
2. under the hypothesis that the learning stage is relevant, compute the attitude variation $\delta\mu = \mathbf{A}\delta\mathbf{i}$ which would produce the same difference image $\delta\mathbf{i}$ in the neighborhood of the reference object attitude μ_0,
3. select 3D points $\mathbf{X}$ on the target surface and compute by ray-tracing, the coordinates of points $\mathbf{X}'$ such that $\mathbf{M}(\mu_0)\mathbf{X} = \mathbf{M}(\mu_0 + \delta\mu)\mathbf{X}'$,
4. noting that these 3D points $\mathbf{X}'$ are projected on the 2D points $\mathbf{M}(\mu)X$, compute the current object location $\mu^*$from this set of 2D/3D correspondences.

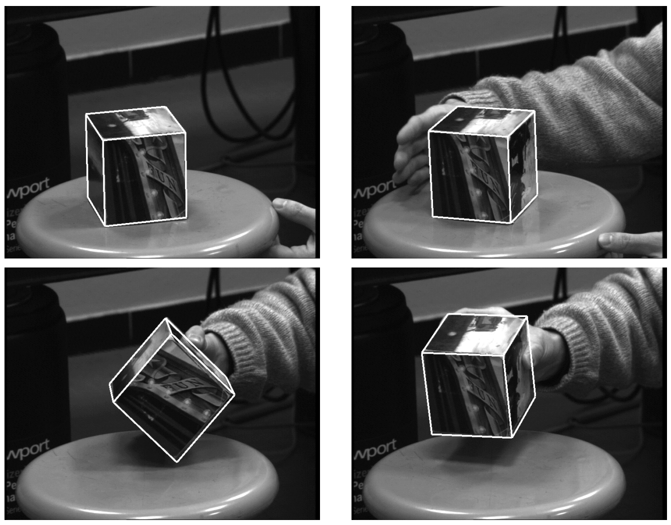

Figure 5: Four images taken from a video sequence.

It is important to note that each step of the previous algorithm can be efficiently implemented. The first and second ones corresponds only to few hundred subtractions, additions and multiplications. The third one can be made easily and rapidly by using the z-buffer of the computer visualization hardware. The last one is inexpensive if using an efficient approach as this proposed by Dementhon [5] (a single matrix multiplication). In summary, the proposed approach can be implemented in real time on a standard personal computer (SGI O^2 in our case).

4 Results

We have performed several experimentations. The one presented in this article concerns the tracking of a 3D textured cube.

The camera has been previously calibrated. The cube is modeled by 3D points belonging to different faces. For the given examples, the points belong to two faces (the front and top ones), which are supposed to be visible during the whole sequence. If one want to track 360° rotations, several interaction matrices should be learned and switched at the proper time. This task is easy because the 3D object attitude is known.

Results presented on Figure 5 have been obtained by tracking 100 points, randomly distributed on the two visible faces. A set of 500 small displacements have been performed during the learning stage, in order to compute the interaction matrix.

5 Discussions

Historically brute force search was used in template matching algorithm. This is inefficient and impractical for parameter spaces higher than 2D translations.

Several authors have recently carried out research concerning numerical methods to *efficiently* minimize the error between the transformed target and reference images. In this section we will discussed about the works of Black and Jepson [1], Gleicher [6], La Cascia *et al.* [9], and Hager and Belhumeur[7]. The comparison will be performed in two directions: the choice for an optimization method and the ability to track 3D motions.

5.1 Optimization method

The idea of tracking by minimizing the error over all the pixels within a region of interest can be seen as an optimization problem. Black and Jepson [1] minimize a nonlinear function using a simple Levenberg-Marquard scheme. One advantage of addressing the problem in its general nature is the possibility to minimize complex functions. Black and Jepson introduced in their formulation non-linear terms compensating for partial occlusions. Registration and reconstruction of occluded parts are obtained simultaneously. However their approach is unfortunately very slow and can only tolerate only very small movements of the object.

Another way, as proposed in this paper, is to linearize the function. In this case, the registration is straightforward. Two approaches have been recently proposed in the literature : Jacobian approximation and difference decomposition.

5.2 Jacobian approximation

Hager *et al.* in [7] proposed a similar approach and estimate the matrix $\mathbf{A}$ in equation (2) by using the inverse of an image Jacobian. This equation shows clearly that $\mathbf{A}(t+\tau)$ can play the role of a Jacobian matrix. 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,

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

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.

Matrix $\mathbf{A}$ can be deduced from the pseudo-inverse of $\mathbf{M}$.

We have shown in [8] that our linearization technique (hyper-plane approximation) is better than the pseudo-inverse of the Jacobian matrix; basically, the image Jacobian approximation approximates the function by a line while the hyper-plane approximation approximates it by an hyper-plane. We experimentally observed that the convergence area of our approach is larger than the one obtained by Hager's method (using Jacobian approximation).

5.3 Difference decomposition

The difference decomposition have first been proposed by Gleicher [6], and have also been used for 3D human face tracking by La Cascia *et al.* [9]. The basic idea is to decompose the difference image into a linear combination of difference templates. Difference templates are obtained by sampling the parameter space during a learning stage. For each point of the parameter space, a difference template is produced. The relation between the parameter variations and the coordinates in the template basis is then straightforward.

This method offers similarity with the eigen decomposition proposed in [1]. However instead of computing an optimal basis, the initial template basis is directly used. From our point of view, a limitation of this method is that the relation giving the parameter variations can not be learned with more example than the number of templates. As it is interesting to reduce as far as possible the number of templates, the parameter space -in spite of its size- is sampled with a very small number of samples. La Cascia *et al.* [9] argue that only four difference vectors per motion parameter are sufficient. In case of a 3D motion, 24 difference templates are used. We have experimentally observed that 24 samples in a 6-dimensional space are not enough to insure the stability of the tracker.

With the proposed hyper-plane approximation, the number of samples used to approximate the relation (2) is not restricted.

5.4 3D motions

Projective transformations have not been used very often in tracking algorithms because the search space is much larger

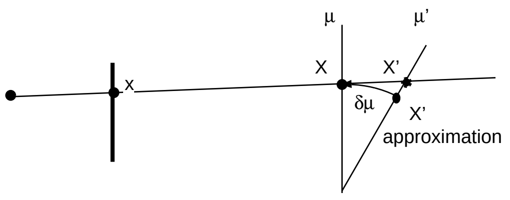

Figure 6: Back projection approximation.

and because the transformations are highly non-linear. In most cases only 2D transformations are considered.

From our knowledge La Cascia *et al.* [9] are the only authors using actually 3D motions (Gleicher only considers homographic motions). However their formulation assume that the difference image obtained by a given displacement is the same, whatever the 3D position of the object is. It is obviously not the case. Their formulation is only valid if the orientation of the object is relatively similar to the orientation used during the learning stage, because of this approximation. We have seen in the previous section that a back-projection of 2D point on the 3D moved model was necessary. However, assuming the motion is small, a new assumption can be done: the position of $\mathbf{X}'$ (as previously defined) can be approximated by $\mathbf{M}(\delta\mu)\mathbf{X}$, instead of being the back projection of $\mathbf{x} = \mathbf{M}(\mu_0)\mathbf{X}$.

In case of fronto-parallels motions (x, y translations and z rotations), this approximation is error free. In case of non fronto-parallels motions (x and y rotations, z translations), as an error is introduced, as shown Figure 6. We have experimentally observed that this approximation only suited for very small angle variations.

In that case, the actual localization can be estimated straightforwardly by the relation: $\mathbf{M}(\mu^*) = \mathbf{M}(\mu)\mathbf{M}^{-1}(\delta\mu)$.

We have made an experiment showing the limitation of this approximation. A video sequence have been processed by the method proposed in this article and by the previously mentioned approximation (used by La Cascia *et al.*). The sequence shows a textured cube which is rotated (1° per image) around the vertical axis. The former method can tolerate more than 50° of perturbation relatively to the reference position while the second one diverges after only 20° of rotation.

To take into account affine variation of luminance the sampling vector is centered and normalized before comparison with the reference template. However, some more complex techniques, as those proposed by Hager [7], can be easily integrated in the proposed scheme, to tolerate more complex illumination variations.

6 Conclusions

We have presented an original and efficient 3D tracking algorithm. Experimental results presented in that paper show our technique greatly improves the previously published approaches.

In our opinion, this article makes two contributions. First, our technique is based on the linearization of a function giving the 3D localization as a function of a difference image; we propose to use an hyper-plan model, which is original and more accurate than other similar techniques (like image Jacobian). The second contribution concerns the geometric aspect of the problem. We deal with actual 3D geometry rather than supposing that a parameter variation gives the same difference image whatever the camera position is. We have shown it greatly improves the stability of the algorithm.

References

[1] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation, January 1998.

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

[3] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. In *Proc. European Conference on Computer Vision*, pages 484–498, Freiburg, Germany, 1998.

[4] T. Darrell, I. Essa, and P. A.P. Task-specific gesture analysis in real-time using interpolated views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(12):1236–1242, 1996.

[5] D. Dementhon and L. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vison*, 15(1-2):123–141, June 1995.

[6] M. Gleicher. Projective registration with difference decomposition. In *CVPR97*, pages 331–337, 1997.

[7] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, October 1998.

[8] F. Jurie and M. Dhome. Real time template matching. In *Proc. IEEE International Conference on Computer vision*, pages 544–549, Vancouver, Canada, July 2001.

[9] 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. *PAMI*, 22(4):322–336, April 2000.