3D visual tracking using a point based model

Christophe Dehais, Vincent Charvillat, Géraldine Morin

IRIT - ENSEEIHT
2, rue Charles Camichel - BP4122 - 31071 Toulouse Cedex 7
Email: {dehais,charvillat,morin}@enseeiht.fr

Abstract

This paper presents a variation around one of the state-of-the-art method for 3D visual tracking. It uses a 3D mesh model of the tracked object. Our main contribution is to substitute this facet based model by a simpler one composed of 3D points. After justifying this approach, we show that the problem can be solved in this context by linear parameters estimation steps. Those linearizations allow for a less complex and less costly implementation than the original algorithm. The proposed algorithm preserves the initial efficiency of the reference method, as shown by our experiments.

1 Introduction

Many applications require tracking rigid (or slightly non-rigid) 3D objects. Some well-known are Augmented Reality (AR) [2], visual virtual servoing [4] or face tracking [11]. In those contexts, cost constraints or some difficulties to setup the environment make vision-based solutions particularly interesting. Recently, methods using a known 3D model of the object of interest have given impressive results [5, 10]. They detect and track visual features such as edges, interest points and textured patterns.

The goals to achieve when designing such algorithms are:

- computational efficiency: applications mentionned above must track the object aproximately at acquisition rate, *i.e.* between 10 and 30 fps.
- robustness: the feature matches are often contaminated by outliers, and a tracking algorithm must be able to correctly deal with them.
- precision: two types of problems frequently arise. First, small estimation errors during tracking are propagated, leading to a large deviation from the expected solution (drift). Secondly, the jittering effect also decreases the tracking quality, something particularly problematic for AR applications.

Vacchetti et al. [11] propose a real time tracking algorithm that fulfills those requirements of robustness and precision. Assuming that a 3D mesh model is available, the authors use interest points correspondences and derive a simplified version of a bundle adjustment problem. The results show very little drift in parameters recovery as well as very good temporal consistency (low jitter). This is due to the integration of offline information in the iterative tracking process. This information comes from registered keyframes of the tracked object. Though a real time implementation of this technique has been demonstrated [11], it remains complicated to obtain for several reasons. First the optimization criterions are highly non-linear, therefore they contain local minima that require the initialization to be close enough to the solution. Moreover a costly iterative optimization method must be used. Finally there are several intermediate steps, in particular the association of extracted interest points to their respective 3D facet on the model. Those difficulties make extensions hard to envision, for example in order to explicitly take care of deformations.

The use of point based 3D models that are not complete, do not contain strong topological information, and combine appearance and geometry, is increasingly being investigated. It has been shown to be of interest in computer vision:

- for object recognition [9]
- for appearance based tracking methods [12, 8]

We also point out that this trend is also observed in computer graphics where the rendering of point based geometries is becoming efficient.

Following these ideas, we propose to benefit from the advantages of point based models and implement a new version of Vacchetti et al's algorithm [11] (named our *reference* algorithm in the remaining of the paper). In section 2 we present the modeling of the problem. The main steps of our method are detailed in section 3. Finally experiments on synthetic and real images sequences are reported in section 4. We compare in depth our algorithm to our own implementation of the reference algorithm.

2 Visual tracking using a 3D mesh model

In this section we give an overview of the reference algorithm of [11].

2.1 Problem modeling and resolution

The goal of the tracking is to recover the pose parameters of the object with respect to the camera, throughout an image sequence I_t.

Camera model. We use the classic pin-hole model for the camera, that can be represented by a 3-by-4 matrix P of the form:

$$P = K.[R|\mathbf{t}]$$

where K is the upper triangular matrix made of *intrinsic* camera parameters (focal length and sensor geometry). We suppose K to be known and constant throughout the sequence. R and $\mathbf{t}$ are respectively the rotation matrix and the translation vector defining the 3D euclidean transformation of the object frame in the camera frame. We note $P_t = K.[R_t|\mathbf{t_t}]$ the pose of the object at time t (corresponding to frame I_t). So the goal sums up as recovering six parameters (three for rotation, three for translation) per frame.

Features extraction. Several methods and features can be used in order to recover 3D motion based on 2D correspondences, such as edges, textured patterns and optical flow. Our approach is based on the extraction and matching of interest points. There are many advantages of doing this: the matching of points can be adapted to handle illumination changes and local occlusions and it offers a wider range of tractable motions compared to optical flow and template matching methods. In our case, Harris points [6] are matched between two successive images I_{t-1} and I_t.

Let p^j_{t-1} be an extracted interest point in image I_{t-1} and $p^i_t = p^{v(j)}_t$ be the matching point in image I_t (see Figure 1). Assuming p^j_{t-1} is part of the image of the tracked object, then it is the projection w.r.t. P_{t-1} of a 3D point N^j that belongs to the surface defined by the mesh model. N^j projects itself onto I_t at point $\tilde{p}^j_t$.

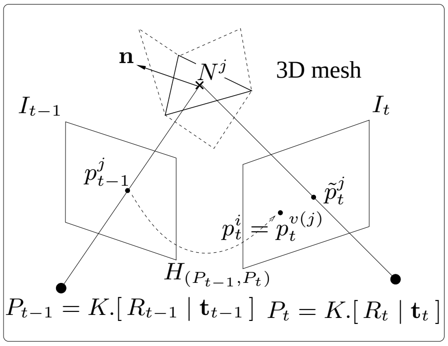

Figure 1: Point matching process and illustration of the criterion in Eq. (1)

We use a centered normalized cross-correlation criterion as a similarity measure of small patches around the points. This measure is invariant to affine illumination changes.

Resolution. Therefore we search for a matrix $P_t = K.[R_t \mid \mathbf{t}_t]$ that jointly minimizes the squared euclidean distances $\|\tilde{p}^j_t - p^{v(j)}_t\|^2$ for all established point correspondences.

We face a particular instance of a bundle adjustment problem, except that here, the knowledge of the model facets allows to avoid the explicit computation of the 3D points N^j. Indeed, p^j_{t-1} and $\tilde{p}^j_t$ are bound by an homography that only depends on pose matrices P_{t-1} and P_t and on the facet $\mathcal{F}$ to which N^j belongs [7]. Thus if $\{x \mid \mathbf{n}^T.x - d = 0\}$ is the plane containing the facet $\mathcal{F}$ and $\mathbf{n}$ its normal, then:

$$\tilde{p}^j_t = H_{(\mathcal{F},P_{t-1},P_t)}.p^j_{t-1}$$

where

$$H_{(\mathcal{F},P_{t-1},P_t)} = K.(\delta R - \delta\mathbf{t}.\mathbf{n}'^T/d').K^{-1}$$

with

$$\delta R = R_t.R^T_{t-1}\ ,\ \delta\mathbf{t} = -\delta R.\mathbf{t}_t + \mathbf{t}_{t-1}$$

$$\mathbf{n}' = R_{t-1}.\mathbf{n}\ ,\ d' = d - \mathbf{t}_{t-1}^T.(R_{t-1}.\mathbf{n})$$

Hence, we solve for each frame the following *non-linear* optimization problem whose parameters are the six parameters of P_t:

$$\widehat{P_t} = \underset{P_t}{\operatorname{argmin}} \sum_j \|H_{(\mathcal{F}(p_{t-1}^j),P_{t-1},P_t)}.p_{t-1}^j - p_t^{v(j)}\|^2 \quad (1)$$

Because localization errors on interest points appear both on p_{t-1}^j and p_t^i, in practice, the formulation is made symmetrical, by transferring p_t^i in image I_{t-1} the same way. Thus the criterion is the sum of two distances, and the estimation is done *w.r.t.* the 12 parameters of P_{t-1} and P_t.

This non-linear least squares problem is solved iteratively by the Levenberg-Marquadt method. Note that, in this formulation, only the initial pose is needed. However, this iterative scheme suffers from the accumulation of estimation errors, which prevents its use for sequences longer than a few hundreds frames. To alleviate this problem, Vacchetti et al. propose to use additional offline information provided by keyframes.

2.2 Integrating keyframes

A keyframe is made of a view of the object to be tracked and the associated camera parameters, accurately recovered, for example by hand. The interest points p_{key}^j detected in these keyframes as well as their corresponding 3D points N_{key}^j are also computed and stored.

For each matched pair of points $(p_t^j, p_{key}^{w(j)})$ between current frame I_t and a keyframe I_{key}, criterion (1) can be completed by also summing the following terms:

$$\|p_t^j - \phi(P_t, N_{key}^{w(j)})\|^2$$

where $N_{key}^{w(j)}$ is a 3D point that maps to $p_{key}^{w(j)}$ on the keyframe I_{key} and $\phi(P,\cdot)$ is the projection according to matrix P.

There are two major difficulties by then. First matching some points between the current frame and a keyframe may be hard with a wide baseline. Vacchetti et al. [11] bypass this problem by rendering a warped version of the keyframe closer to the current frame. The second issue is the choice of the keyframe. One way is to choose the one having the closest set of pose parameters *w.r.t.* the current pose P_{t-1}. A better approach is to use a similarity measure between the current frame and each available keyframe, as proposed in [11].

3 An approach using a point based model

The algorithm presented in the previous section is quite difficult to implement. We think that 3D points are sufficient to explain the 2D motion between matched points. We get rid of facets and this way gain more flexibility (see section 5 for insights). In the next paragraph we introduce the points based model we will use in the remaining of the paper.

3.1 The point based model

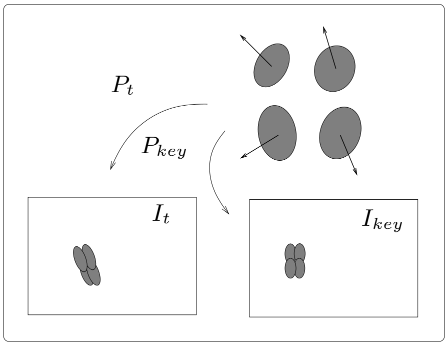

Figure 2: The model is defined by a set of points and textured patches. Two projections (in the current frame and in a keyframe) are shown.

The model we use is composed of a set of patches defined by a 3D point $M_i = (x_i, y_i, z_i)^T$, a normal $\mathbf{n}$ an its texture (see Figure 2). It is similar to the one used by Muñoz et al. [8].

3.2 A motion model well-suited for points

The rigid 3D motion between two successive images I_{t-1} and I_t is described by a three-dimensional euclidean transformation (a 4×4 matrix):

$$M = \begin{bmatrix} \delta R & \delta\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

where δR and $\delta \mathbf{t}$ are the rotation and translation of the object between the two poses P_{t-1} et P_t. Hence:

$$P_t = P_{t-1}.M$$

To obtain a 2D motion model suited for the use of 3D points, we linearize the projection of the 3D motion as proposed by Drummond [5].

The euclidean matrix M has an exponential map form written:

$$M = exp(\sum_{i=1}^{6} \alpha_i G_i)$$

where G_i matrices are the generators of 3D elementary motions (rotations and translations *w.r.t.* the axes of the object frame):

$$G_i = \begin{bmatrix} 0 & -\delta_{i,6} & \delta_{i,5} & \delta_{i,1} \\ \delta_{i,6} & 0 & -\delta_{i,4} & \delta_{i,2} \\ -\delta_{i,5} & \delta_{i,4} & 0 & \delta_{i,3} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where $\delta_{i,j}$ is the Kroenecker trigger ($\delta_{i,j} = 1$ if $i = j$ else $\delta_{i,j} = 0$) and $\alpha = (\alpha_1, \cdots, \alpha_6)$ are the corresponding parameters of the translation and rotation.

Under small-motion approximation, we have:

$$M \approx I + \alpha_i.G_i \tag{2}$$

one can compute a generating family $\{\mathbf{L}_{j,i}\}_{i=1,\dots,6}$ of 2D motion vectors at any point $p^j = (u, v, w)$ that is the projection in the image plane of a 3D point $N^j = (x, y, z, 1)^T$. We get:

$$\forall i \in [1, 6], \; \mathbf{L}_{j,i} = \begin{pmatrix} \frac{u'w - uw'}{w^2} \\ \frac{v'w - vw'}{w^2} \end{pmatrix}$$

with

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = P.G_i. \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

The $\{\mathbf{L}_{j,i}\}$ family is used to relate the 2D motion induced by a 3D motion parameterized by α. Figure 3 shows the vector field $\mathbf{L}_{j,6}$ corresponding to a rotation around z axis.

Figure 4 shows this motion model. For the computation of the family $\{\mathbf{L}_{j,i}\}$ corresponding to any points p^j_{t-1}, including those which are not stored in the model, see section 3.4.

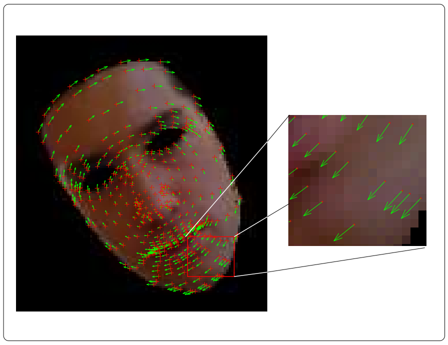

Figure 3: The vector field $\mathbf{L}_{j,6}$ corresponding to a rotation around z axis.

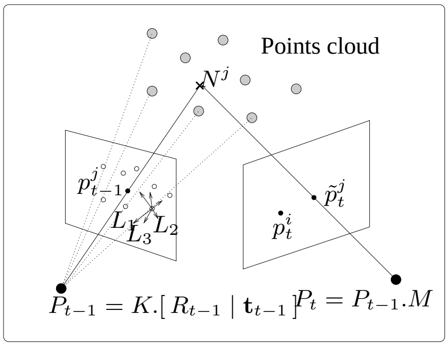

Figure 4: Projection of the generators G_i in the image. The $\{\mathbf{L}_{j,i}\}$ are being represented at the projection of a particular model point

We use this motion model to find the 3D motion from many measures of the local 2D motion (interest points correspondences). It has been previously used with edges of a wireframe 3D model [5, 11, 4], but never to our knowledge has it been applied to a point based model.

In this scheme, tracking boils down to the estimation of the inter-frame euclidean transformation M, using its parameters α. The next section show how to linearly estimate these parameters.

3.3 Resolution

Each pair of matched interest points gives a motion vector $\mathbf{d}^j$:

$$\mathbf{d}^j = p^j_{t-1} - p^{v(j)}_t$$

For a given transformation M such that $P_t = P_{t-1}.M$ and with equation (2), we have:

$$\mathbf{d}^j = \sum_{i=1}^{6} \alpha_i L_{j,i}$$

Let $\mathbf{d}^j = (d^j_x, d^j_y)^T$ be the motion vector, $\mathbf{L}_{j,i}$ the i^{th} vector of the generating family calculated at p^j_{t-1} and $L_j = (\mathbf{L}_{j,1}, \cdots, \mathbf{L}_{j,6})$ the 2×6 matrix containing the 6 vectors of $\{L_{j,i}\}$. Then the following relation holds:

$$D = L.\alpha = L.\begin{pmatrix}\alpha_1 \\ \vdots \\ \alpha_6\end{pmatrix} \qquad (3)$$

where

$$D = \begin{pmatrix} d^x_1 & d^y_1 \\ \vdots & \vdots \\ d^x_J & d^y_J \end{pmatrix} \text{ et } L = \begin{pmatrix} \underline{L_1} \\ \vdots \\ \overline{L_J} \end{pmatrix}$$

J is the number of matches.

Equation 3 linearly relates the parameters of euclidean motion and the apparent motion D calculated at the interest points locations by means of the matrix L. We can solve this problem thanks to the pseudo-inverse of L:

$$\alpha = L^+.D$$

This linear estimation scheme makes possible to derive our efficient algorithm that we introduce in the next sections.

3.4 Computation of the 2D motion model

We measure the motion vectors sparsely, at interest points locations. Those interest points are considered to be the projection of a particular 3D point on the object surface. In general, this particular 3D point does not belong to the model (see Figure 4). To apply the scheme presented above, we need to compute the family $\{\mathbf{L}_{j,i}\}$ at each interest points p_j.

We propose two approaches. The first is a back-projection of the interest points on the model surface locally reconstructed with Moving Least Squares (MLS) [1]. The computational cost of this technique is tractable when the number of rays is low (in our case as much as the number of interest points).

The second possibility consists in interpolating the vector field shown in Figure 3. In our implementation, we use a Thin Plate Spline [3] to evaluate the $\{\mathbf{L}_{j,i}\}$ families in any points. The interpolation process only relies on the visible 3D points which are filtered based on their normal and patches.

3.5 Integrating a priori knowledge

Iteratively updating the pose suffers from error accumulation that can lead to the failure of the process after few frames (see experimental results in section 4). As presented in section 2.2, we also introduce keyframes. We then proceed in two steps:

Step 1. We apply the scheme presented in section 3.3.:

- extract and match interest points p^j_t,
- build D^1 (Eq 3) from the matches $(p^j_{t-1}, p^{v(j)}_t)$,
- compute the vectors $L^1_{j,.}$ at points p^j_{t-1}, build the matrix L^1,
- estimate the update $\widehat{\alpha}^1 = L^{1+}.D^1$ of the 3D motion parameters.

Step 2. We refine the current estimation based on a keyframe.

- select a keyframe and render the model patches in the neighborhood of the current pose (using the estimation of the current pose P_t, computed as the output of Step 1, see Figure 2),
- build D^2 from the matches $(p^j_t, p^{w(j)}_{key})$,

- compute the vectors $L^2_{j,.}$ at points p^j_t, build the matrix L^2,
- refine the estimation thanks to the update $\widehat{\alpha}^2 = L^{2+}.D^2$.

We can notice that Step 2 can be repeated.

4 Experiments

4.1 Synthetic data

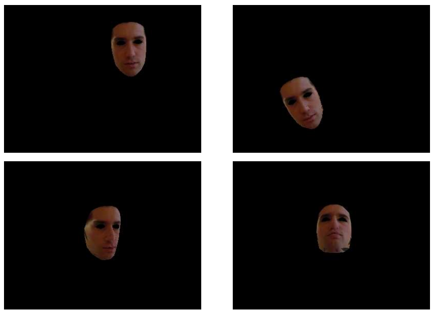

Figure 5: 4 frames from the synthesized sequence.

We use a CAD face model, made by an artist based on photographs. The model is a triangle based 3D mesh. This mesh has been manually registered to a real view of the face in order to compute texture coordinates for each point of the mesh. From those data, we produce a point based model as described in section 3.1. It is made of roughly 500 points. Then we generate an OpenGL rendered sequence of the textured face model, by varying the 6 parameters of the camera pose from frame to frame. Figure 5 shows 4 images of the generated sequence.

We take the known parameters used to generate the sequence as a ground truth. This allows to evaluate the estimated values recovered by different methods: our method without keyframe, our method with keyframes and the reference method with keyframes. All methods were implemented in Matlab.

Figure 6 show the tracking results on the 300 first images of the synthetic sequence (which consists of various rotations and translations of the model in front of a fixed pin-hole camera). To mimic the real world, we introduced a gaussian error with zero mean and a standard deviation of 2 pixels. The dotted curves show the ground truth as used to generate

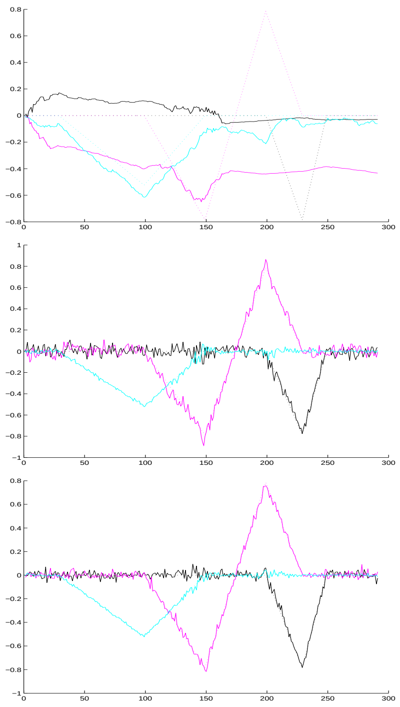

Figure 6: Evolution of the 3 rotation parameters for the 300 fi rst images of the sequence. The dotted curves shows the ground truth. Top: our method without keyframe ; middle: our method with keyframes ; bottom: the reference method, with keyframes.

the sequence and the plain curves are the recovered values for each methods.

We draw two conclusions from these results: first, both the reference method and our linear approach are greatly improved by the use of keyframes; it efficiently avoids drift effect (see the top part of Figure 6). Moreover, despite the linearization we make, our method stays stable, although logically less precise. The component of the vector of pose parameters α being of the same magnitude, we can compare the absolute median error; it is about 10^{-2} for our method against 10^{-4} for the reference method. Moreover we obtain this result for a much lower computationnal cost. Our approach, by linearizing the problem early, only requires some computations of pseudo-inverse matrices. By contrast the reference algorithm involves hundreds of evaluations of the non-linear criterion (in particular for jacobian matrices estimation).

4.2 Real data

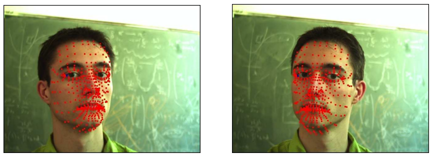

Figure 7: Two images from the tracking on a real sequence.

We also have results on real sequences acquired by a calibrated digital color camera (see Figure 7). For the tracking of such sequences to succeed, the different steps of our algorithm need to be robust to false matches. This is why we use an iterated least-squares method (M-estimation), consisting in weighting the measures from the interest points matches in the lines of the D matrices (see Eq. 3)

5 Conclusion and future work

This paper demonstrates the possibilty to use a point based 3D model for the visual 3D tracking problem. The proposed technique is at the same time simpler to implement and runs faster than a state-of-the-art tracker. In addition, it introduces consistence between the object model and the low-level computations (the 2D motion model is also based on points).

Finally, we think that using a simpler model without topology will be interesting for taking into account deformations and occlusions. A point based model could be refined all along the tracking. We will explore those aspects in future work.

References

[1] Anders Adamson and Marc Alexa. Ray tracing point set surfaces, 2003.

[2] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.

[3] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transaction on PAMI*, 11(6), 1989.

[4] Andrew I. Comport, ric Marchand, and Franois Chaumette. Robust model-based tracking for robot vision. In *IEEE Int. Conf on Intelligent Robots and Systems, IROS04*, Sendai, Japan, September 2004.

[5] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, July 2002.

[6] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Fourth Alvey Vison Conference*, Manchester, 1988.

[7] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, March 2004.

[8] Enrique Munoz, Jose M. Buenaposada, and Luis Baumela. Efficient model-based 3d tracking of deformable objects. In *Proceedings of ICCV 2005*, pages 877–882, Beijing, China, October 2005.

[9] Frederick Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3d object modeling and recognition using affine-invariant patches and multi-view spatial constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 272–277, Madison, WI, June 2003.

[10] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *International Symposium on Mixed and Augmented Reality*, Arlington, VA, November 2004.

[11] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1391–1391, 2004.

[12] Charles S. Wiles, Atsuto Maki, and Natsuko Matsuda. Hyperpatches for 3d model acquisition and tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(12):1391–1403, 2001.