Fast Model Tracking with Multiple Cameras for Augmented Reality

Alberto Sanson
Dipartimento di Informatica
Università di Verona
Strada Le Grazie, 15
37134 Verona, Italy
sanson@sci.univr.it

Umberto Castellani
Dipartimento di Informatica
Università di Verona
Strada Le Grazie, 15
37134 Verona, Italy
castella@sci.univr.it

Andrea Fusiello
Dipartimento di Informatica
Università di Verona
Strada Le Grazie, 15
37134 Verona, Italy
andrea.fusiello@univr.it

ABSTRACT

In this paper we present a technique for tracking complex models in video sequences with multiple cameras. Our method uses information derived from image gradient by comparing them with edges of the tracked object, whose 3D model is known. A score function is defined, depending on the amount of image gradient "seen" by the model edges. The sought pose parameters are obtained by maximizing this function using a non deterministic algorithm which proved to be optimal for this problem. Preliminary experiments with both synthetic and real sequences have shown small errors in pose estimations and a good behavior in augmented reality applications.

Categories and Subject Descriptors

I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Virtual reality*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*tracking, stereo*; H.5.1 [**Information Systems Applications**]: Multimedia Information Systems

General Terms

Algorithms

Keywords

Pose estimation, Registration, Exterior orientation

1. INTRODUCTION

In Augmented Reality (AR) applications [1], in order to make synthetic graphics appear in proper place, it is necessary to know exactly *pose* (position and orientation) of the camera in the real world. This is the so-called *registration problem*, which have been addressed in early systems with tracking devices (like beacons, transponders, etc.). In video-based AR, where the real image and the graphic overlay are combined using a computer, the same video camera used to capture video serves as a tracking device. Since the system captures a digitized image of the real environment, it is possible to enforce registration of the model onto the view of the real object, thereby closing the loop. Solving this problem repeatedly for each frame of a video sequence is referred to as *model tracking.* Model tracking consist in computing the correct projection of the model onto the real object (registration), which is equivalent to solving for the pose of the camera with respect to the object.

Most approaches are based on the extract-and-match paradigm [6, 3, 4, 11, 13]: image features are extracted and then matched against stored model features. A drawback of all these methods is that errors introduced in the feature extraction phase cannot be eliminated subsequently. Moreover, matching is usually strongly problem dependent and based on fine tuned thresholds; occlusions pose a serious challenge. As a result, these methods are likely to fail in cluttered and complex environment.

In this paper we present a technique for tracking models of rigid objects with multiple cameras that does not require feature extraction and matching. The main idea [9, 7, 12] consists of comparing the image gradient information with edges of the tracked model, assuming that the latter give rise to local gradient maxima. A score function is therefore defined, that measures the amount of image gradient collected by the projected model edges: Maximizing this function is tantamount to obtaining pose parameters of the camera.

Speed and accuracy of the tracking depends critically on two steps: i) the optimization procedure and ii) computation of the score function, which entails model projection (rendering). In both these respects we differ from [8, 12]. As for the first issue, after a thorough analysis of optimization algorithms (reported in [10]) we selected the *Shaker* algorithm [2], a non-deterministic method based on adaptive noise. A simple Kalman filter that tracks the pose of the camera provides the starting guess for the optimization. The computation of the score function combines the speed of rasterization (via OpenGL) with the accuracy of a vectorial representation.

Thanks to these solutions we we improve on [12] in both accuracy and speed. A further improvement in occlusion handling and accuracy is brought by the introduction of a second camera.

In principle our technique can track any rigid object, that OpenGL can render, including smooth-boundaries ones. In practice, real-time requirements pose limits the complexity of the model.

2. CONCEPTS

Let $\tilde{\mathbf{w}} = [x, y, z, 1]^\top$ be the homogeneous coordinates of a 3D point W in the *model reference frame* and $\tilde{\mathbf{m}} = [u, v, 1]^\top$ the homogeneous coordinates of its projection M in the image plane (in pixels). The pinhole camera is modeled by the *perspective projection matrix* $\mathbf{P} = [\mathbf{R}|\mathbf{t}]$. $\mathbf{A}$ depends on five *intrinsic parameters*; $\mathbf{R}$ and $\mathbf{t}$ are the *extrinsic parameters* that define the rotation and translation – respectively – bringing the camera reference frame onto the object reference frame. $\mathbf{R}$ is parameterized by a vector $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ such that $\mathbf{R} = e^{\boldsymbol{\omega}}$. The matrix exponential is computed according to the Rodriguez's formula. The modulus of $\boldsymbol{\omega}$ gives the angle of rotation, while its direction represents the axis.

The mapping from 3-D points to 2-D points is given by:

$$\tilde{\mathbf{m}} \simeq \mathbf{P}\tilde{\mathbf{w}} \tag{1}$$

where $\simeq$ means equal up to a scale factor. If we collect the 11 parameters (6 extrinsic and 5 intrinsic) in one vector $\boldsymbol{\phi}$, we can define the projection operator Π as:

$$\mathbf{m} = \Pi(\mathbf{w}; \boldsymbol{\phi}). \tag{2}$$

Given a sufficient number of $(\mathbf{m}_i, \mathbf{w}_i)$ matching pairs, one may compute the parameters vector $\boldsymbol{\phi}$ such that Eq. (2) is satisfied for all i.

In principle one could estimate all the 11 parameters, as knowing a model of the object make the problem equivalent to camera calibration [9]. In practice, if viewing conditions are close to affine, the translation along the optical axis becomes indistinguishable from a focal length change. As in a tracking application the viewing conditions cannot be guaranteed (unlike in the calibration process), we will assume that intrinsic parameters are known and constant, i.e., $\boldsymbol{\phi} = (\boldsymbol{\omega}, \mathbf{t})$.

3. TRACKING METHOD

At each discrete time step t, the 3D model (known) is projected according to the predicted pose of the camera, and the actual pose $\boldsymbol{\phi}$ of the camera is computed by optimizing an objective function $E(\boldsymbol{\phi})$. The prediction is given by a simple Kalman filter which has also the effect of smoothing out pose changes.

3.1 Objective function

Following [9, 7], pose computation is cast as an optimization problem, where the score function is defined as the integral of the local image gradient along the projected model contours. The objective function to be optimized writes:

$$E(\boldsymbol{\phi}) = \textstyle\sum_i |\nabla I(\Pi(\mathbf{w}_i; \boldsymbol{\phi}))| \tag{3}$$

where $\nabla I(\mathbf{x})$ is the image gray level gradient computed at point $\mathbf{x}$. Points $\mathbf{w}_i$ belongs to visible model-edges (both contour and color edges). Furthermore, in presence of noise or cluttered background, it is advisable to take into account the direction of the gradient [7] so as to make the cost function more selective. Let $\Gamma_{\boldsymbol{\phi}}$ be the set of the projected visible edge points and $\hat{\mathbf{n}}_{\Gamma_{\boldsymbol{\phi}}}(\mathbf{x})$ the edge direction, the objective function is defined as:

$$E(\boldsymbol{\phi}) = \frac{1}{|\Gamma_{\boldsymbol{\phi}}|} \textstyle\sum_{\mathbf{x}} |(\nabla I(\mathbf{x}) \cdot \hat{\mathbf{n}}_{\Gamma_{\boldsymbol{\phi}}}(\mathbf{x})) \chi_{\Gamma_{\boldsymbol{\phi}}}(\mathbf{x})| \tag{4}$$

where $\chi_{\Gamma_{\boldsymbol{\phi}}}$ is the characteristic function of the set $\Gamma_{\boldsymbol{\phi}}$, also called *model edge map*. In order to have a smooth and larger basin of attraction for the optimum, we consider not only points belonging to $\Gamma_{\boldsymbol{\phi}}$, but also their neighbors in a given range, weighted by a Gaussian function. This can be formalized by considering a fuzzy characteristic function $\bar{\chi}_{\Gamma_{\boldsymbol{\phi}}}$ which decreases with a Gaussian law as points are farther from the actual edge.

3.2 Numerical optimization

As the score function cannot be expressed in analytical form, we had to exclude from consideration gradient-based methods and resort to *direct-search* maximization algorithms.

After a thorough comparative analysis of several direct-search algorithms, reported in [10], the *Shaker* algorithm proposed by Caprile and Girosi [2] was selected, as it has shown best results on both synthetic and real experiments.

Given a real multivariate function $g = g(\mathbf{x})$, defined in a domain A of $\mathbb{R}^n$, let $\mathbf{x}$ be a point in A and $\mathcal{P} = \{P_1, ..., P_k\}$ a partition of set I_n formed by all natural numbers greater than 0 and smaller than $n+1$ ($I_n = \{1, ..., n\}$). For each element $P_i \in \mathcal{P}$ we define a positive number ω_i and let Ω be the set of all these numbers; such set represents the noise list. Let $rand(a, b)$ be the result of a real number random extraction in $[a, b]$ and let η be a given accuracy threshold.

```
SHAKER ALGORITHM
    $g_c := g(\mathbf{x})$;
    initialize $\Omega$ ;
    while $\forall i \in I_k :\ w_i < \eta$;
        for each $P_i \in \mathcal{P}$;
            a $n$-dimensional vector $\mathbf{v}$ is generated s.t.:
            $\mathbf{v}_j = \begin{cases} rand(-w_i, w_i) & \text{if } j \in P_i \\ 0 & \text{otherwise;} \end{cases}$
            $g_a := f(\mathbf{x} + \mathbf{v})$;
            if $g_a < g_c$ then:
              $g_c := g_a$;
              $\mathbf{x} := \mathbf{x} + \mathbf{v}$;
              $w_i := 2w_i$;
            otherwise
              $w_i := w_i/2$.
            end
    end
```

In our application, parameters are partitioned into $P_1 = \{t_x, t_y, t_z\}$ and $P_2 = \{\omega_x, \omega_y, \omega_z\}$.

3.3 Using multiple cameras

As reported in [11], general benefits derived by using multiple cameras concern the handling of occlusions and a better conditioning of the problem.

Let us consider the case of two cameras [11], and let $\boldsymbol{\phi}_c = (\boldsymbol{\omega}_c, \mathbf{t}_c)$ the (fixed) pose of the right camera with respect to the left camera, computed once for all by calibration. Let $\boldsymbol{\phi}_l = (\boldsymbol{\omega}_l, \mathbf{t}_l)$ and $\boldsymbol{\phi}_r = (\boldsymbol{\omega}_r, \mathbf{t}_r)$ be the current pose of the left and right camera, respectively. Assuming the left camera as the reference one, the total score function writes:

$$E'(\boldsymbol{\phi}) = E'(\boldsymbol{\phi}_l) = \tfrac{1}{2}(E(\boldsymbol{\phi}_l) + E(\boldsymbol{\phi}_c \circ \boldsymbol{\phi}_l)) \tag{5}$$

where $\circ$ denotes the composition of rigid tranformations.

3.4 Projection of visible edges

The objective function of our tracking algorithm is computed using the projection of visible model-edges. A raster representation – such as that readily obtained using OpenGL – is unsuited to our needs, because the normal to each projected edge in the image is required. A vectorial representation of projected edges is needed, but vectorial hidden line removal is too expensive.

The idea is to use back face culling, which is easy to perform, to obtain the vectorial representation for a superset of the visible edges. First, we compute and store in a matrix the contribution of each pixel to the cost function, using the edges (vectorial) produced by back face culling. Than we mask this matrix with the true model edge map, obtained as the wireframe rendering (raster) of the model. The resulting matrix will contain only the contribution of the visible edge-points, and the sum of its elements yields the value of the objective function.

For smooth-boundaries objects (like cylinders) we render filled polygons with flat shading. When doing the masking, only "white" pixels that have at lest one "black" neighbor are considered as "1". All the others are considered as "0", so as to mask out the interior of the objects. Only contour edges can be used in this case.

4. RESULTS

In this section some preliminary experiments are presented, aimed at validating our algorithm in both synthetic and real cases, with different objects and motion patterns. Real video sequences have been captured with a Videre color stereo camera with resolution of 640×480 pixels. The maximum tracking rate is similar for all the tested sequences and it reaches 9 fps on the average, using a Mobile AMD Athlon XP @ 2 GHz. Synthetic experiments have been carried out in order to assess the registration accuracy with respect to the ground truth. In all the experiments the background is fairly cluttered, and this adds realistic disturbance to the score function. More details on experiments can be found in [10]. Video sequences are available on the WWW[1].

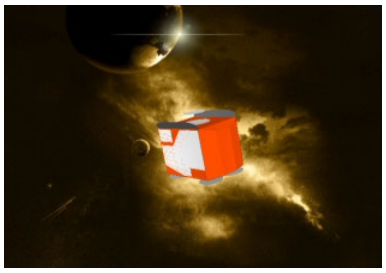

Figure 1: Sample frame from the "Satellite" synthetic sequences.

Experiment 1: Satellite
In this experiment the synthetic satellite shown in Figure 1 has been tracked. It is moving toward the observer with a screw motion (all the pose components are changing) from far ($17m$) to close ($6m$) to the observer. The size of the satellite is about $2m^3$. As can be noted in Figure 2, the error in the estimate of rotation angle is always less than 1

[1]`http://www.sci.univr.it/~fusiello/demo/mdt`.

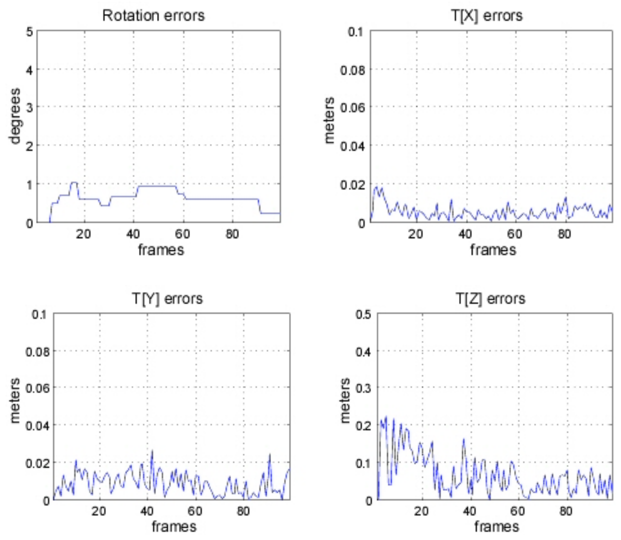

Figure 2: Satellite - pose errors.

degree and the translation errors are less than $2cm$ for t_x and t_y. As expected, the error for t_z is higher (one order of magnitude).

Experiment 2: Printed circuit board
In this experiment the system was requested to track a real printed circuit board (PCB). Although the PCB has many details, a simplified model containing only the main model edges is sufficient for tracking. Several cases have been tested: (i) mainly forward motion (figure 3.a - 3.b), (ii) mainly rotation (figure 3.c - 3.d), (iii) severe occlusion (Figure 3.e - 3.f). In all those sequences the system performed reasonably well.

Experiment 3: Spray bottle
In this experiment the system was required to track a real smooth-boundaries object: the spray bottle depicted in Figure 4.a. Eventually the video sequence was augmented with the synthetic Utah teapot.

A more complex video that simulates the interaction between real and synthetic objects is available on the WWW[1].

5. CONCLUSION

In this paper we presented a technique for tracking complex models in video sequences with multiple cameras. The problem has been modeled as the maximization of a score function which measures the fit of the projection of the model onto the actual images. In order to obtain the correct pose parameters, an efficient optimization technique has been applied. The projection of the model combines the speed of rasterization with the accuracy of a vectorial representation. The method can cope with multiple cameras, complex objects (both polyhedral and smooth boundaries), general motion and is robust to occlusions. Furthermore, it can work in (nearly) real-time.

Preliminary experiments with a small baseline stereo pair are encouraging. This work will be eventually implemented in a system called VE^3 consisting of a room with five cameras pointing inward and tracking objects in real-time. The system is supposed allow interaction with a virtual world projected onto a wall. In this context, more experiments will be conducted.

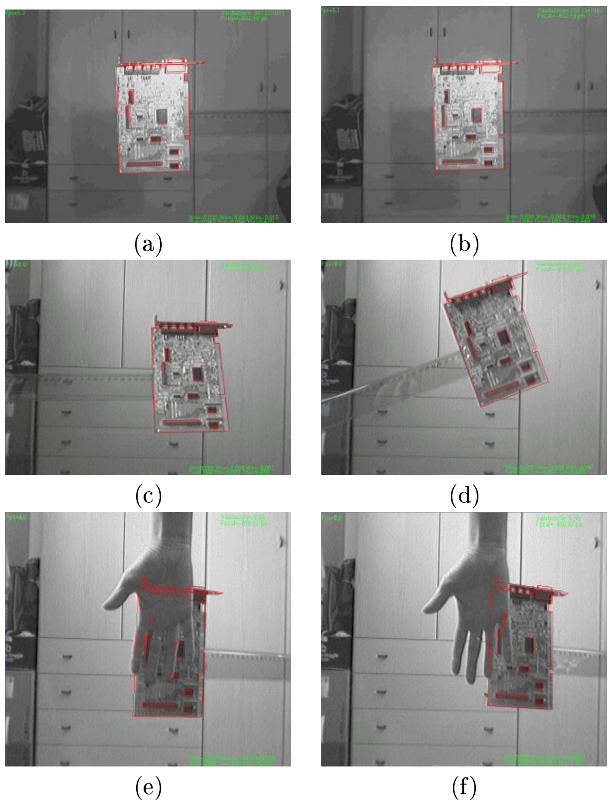
Figure 3: **Selected frames from the PCB experiment. The model outline is overlaid onto the real images.**

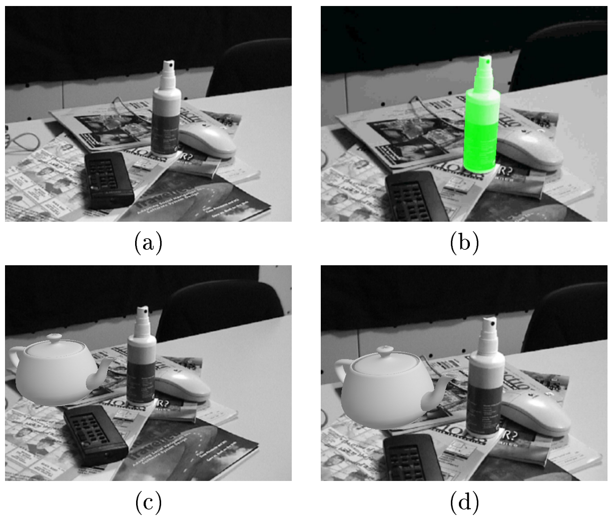
Figure 4: **Spray bottle. Sample frame from the original sequence (a) and from the tracking sequence (b), where the model is overlaid in green. Two frames augmented with the Utah teapot (c) (d).**

6. ADDITIONAL AUTHORS

Additional authors: Vittorio Murino, (University of Verona) email: `vittorio.murino@univr.it`).

7. REFERENCES

[1] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.

[2] B. Caprile and F. Girosi. A non Deterministic Minimization Algorithm. A.I. Memo No. 1254 and C.B.I.P. Paper No 58, MIT Artificial Intelligence Laboratory and Center for Biological Information Processing, Whitaker College, September 1990.

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

[4] T. Drummond and R. Cipolla. Real-time tracking of multiple articulated structures in multiple views. In *Proceedings of the European Conference on Computer Vision*, pages 20–36, 2000.

[5] R. E. Kalman. A new approach to the linear filtering and prediction problems. *Journal of Basic Engineering*, 82D:35–45, 1960.

[6] D. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, March 1987.

[7] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2-D-3-D model-based approach. *Proceedings of the IEEE International Conference on Computer Vision*, 1999.

[8] E. Marchand and F. Chaumette. Virtual visual servoing: a framework for real-time augmented reality. In H.-P. S. G. Drettakis, editor, *EUROGRAPHICS 2002 Conference Proceeding*, volume 21(3) of *Computer Graphics Forum*, Saarebrcken, Germany, September 2002.

[9] L. Robert. Camera calibration without feature extraction. *Computer Vision, Graphics, and Image Processing*, 63(2):314–325, March 1996.

[10] A. Sanson. Realtà aumentata tramite inseguimento di modelli con telecamere multiple. Tesi di laurea, Università degli Studi di Verona, Corso di Laurea in Informatica, 2004.

[11] R. Thompson, I. Reid, A. Munoz, and D. Murray. Providing synthetic views for teleoperation using visual pose tracking in multiple cameras. *IEEE Transactions on Systems, Man and Cybernetics*, 31(1):43–54, 2001.

[12] A. Valinetti, A. Fusiello, and V. Murino. Model tracking for video-based virtual reality. In *Proceedings of the 11th International Conference on Image Analysis and Processing (ICIAP 2001)*, pages 372–377, Palermo, Italy, 26-28 September 2001.

[13] A. Worrall, K. Baker, and G. Sullivan. Model based perspective inversion. In *Alvey Vision Conference*, pages 13–18, 1988.