Model tracking for Video-based Virtual Reality

Alberto Valinetti, Andrea Fusiello and Vittorio Murino

Dipartimento di Informatica
Università degli Studi di Verona
Strada Le Grazie, 15 - 37134 Verona, Italy
{fusiello,murino}@sci.univr.it

Abstract

This paper presents a technique for tracking complex objects (both polyhedral and smooth boundaries) in a monocular sequence. Our aim is to use this model tracking *method in an Augmented Reality context to compute the pose of a real object to be able to register it with a synthetic one. A scalar score function for an object pose is defined, based on the local image gradient along the projected model boundaries. A local search is then carried out in the configuration space of the pose to maximize the score. This technique is robust to occlusions, since the whole object contour is used, not just few control points. The proposed method is effective yet simple. No image feature extraction is necessary and no complex temporal evolution is used. Experimental results with real sequences show good performance of our technique.*

1. Introduction

In this paper we present a technique for tracking complex objects (both polyhedral and smooth boundaries) in monocular sequences where the camera or the object is moving. The model of the object is known, whereas motion and camera parameters are unknown.

Our aim is to use this *model tracking* method in an Augmented Reality context to solve the so-called *registration problem*. Augmented Reality (AR) supplements reality by allowing the user to perceive virtual objects composed with the real world. A comprehensive review of AR can be found in [1]. In order to make synthetic graphics appear in proper place (for example, as a wire-frame outline superimposed on top of a corresponding real-world object), it is necessary to know exactly where the camera (or the user) is 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 (opposed to optical approaches, where the overlay takes place by means of a see-through display), 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 world, thereby closing the loop. A known object is tracked in the images, and the camera pose (position and orientation) is computed as a result. Since the pose calculation is accurate in the image plane, the perceived image alignment error is minimized.

The problem of computing the position and orientation (*pose*) of a camera given its intrinsic parameters and 3D-2D points correspondence is known as the *absolute orientation problem* [5].

Model tracking consists basically in repeatedly solving absolute orientation problems, with the difference that correspondences are to be computed as well and that temporal coherence is exploited to alleviate the task of model to image matching, for example by using a Kalman Filter [12].

Many of the methods in literature differ in the way they match image features to model features. Lowe, in his pioneering work [9, 10], extracts features in the image (points or lines) and then relies on grouping in order to match model with image features. In our experience grouping is not a reliable technique, for it is strongly problem dependent and based on fine tuned thresholds. In [16] the model – image matching is iteratively refined following the idea of the Iterative Closest Point algorithm [2]. A drawback of all these methods is that errors introduced in the feature extraction phase cannot be eliminated subsequently.

Other methods [6] are based on the projection of model control points. These points are matched to image lines, usually with local search techniques, and pose is computed from point – line correspondence. Robust fitting must be used, because some correspondence may be wrong. These techniques are easily broken by occlusions.

The work by Worral et. al [15] first introduced an *active* model, in which the object's pose is successively updated according to forces derived by examining local peaks of directional derivatives of the image, under the control of the current estimate.

Our work is close to the one by Robert [13] and by Marchand et al. [11]. The former is a work on camera calibration that for the first time (at the best of our knowledge) introduced the idea of optimizing a criterion computed directly from the gray-level image, without extracting reference points explicitly. It first computes a rough estimate of the projection matrix. The refinement stage requires a set of 3-D reference points which should project in the image onto edge points. Then, for any given set of camera parameters, an energy value is computed from the image gradient at the projected reference points. Camera calibration consists of minimizing this energy iteratively.

In our work, starting from the ideas outlines above, a scalar evaluation score for an object pose is defined, based on the local image gradient along the projected model lines. A local search is then carried out in the configuration space of the pose to maximize the score. Since this algorithm consists of iteratively optimizing a criterion directly measured in the image, the process is somewhat analogous to an active contour [8]. Yet, this scheme is passive, in that the value at a single pose gives no indication on the movement of the model most likely to improve the score. This technique is robust to occlusion, since it uses the whole object contour, not just few control points. In summary the features of the algorithm are: i) no image feature extraction is necessary; ii) no complex temporal evolution like Kalman filter is used; iii) it copes with zoom cameras; iv) it copes with any type of object that OpenGL can render (polyhedral and smooth objects).

Our work differ from [11] in point iii) and iv), in the minimization algorithm, and in the special two steps procedure for the computation of the focal distance.

We demonstrate our model tracking algorithm in the context of a video-based virtual reality applications, where the tracker is used to compute the camera pose, which allows to overlay a synthetic object (with OpenGL) to the real image. Demonstration movies (MPEG) are available from our web site: http://www.sci.univr.it/~fusiello/demo/mdt.

The rest of the paper is organized as follows. In Section 2 the notation is introduced and the pinhole camera model is quickly reviewed. Section 3 describes our method, and Section 4 reports some results. Finally, conclusions are drawn in Section 5.

2. Notation and Basics

Let $\mathbf{w} = [x \;\; y \;\; z \;\; 1]^\top$ be the homogeneous coordinates of a 3D point W in the *model reference frame* and $\mathbf{m} = [u \;\; v \;\; 1]^\top$ the homogeneous coordinates of its projection M in the image plane (pixels). The mapping from 3-D coordinates to 2-D coordinates is the *perspective projection*, which is is given by the $3{\times}4$ matrix $\mathbf{P}$:

$$\kappa\mathbf{m} = \mathbf{P}\mathbf{w}, \tag{1}$$

where κ is an arbitrary scale factor. The camera is therefore modeled by its *perspective projection matrix* $\mathbf{P}$, which can be decomposed, using the QR factorization [4], into

$$\mathbf{P} = \mathbf{A}[\mathbf{I}|\mathbf{0}]\mathbf{G}. \tag{2}$$

The matrix $\mathbf{A}$ depends on five *intrinsic parameters* only: focal length in pixel, aspect ratio, principal point and skew factor. Camera position and orientation (the six *extrinsic parameters*) are encoded by 4×4 matrix $\mathbf{G}$ representing (in homogeneous coordinates) the rigid transformation that brings the camera reference frame onto the model reference frame.

If we collect the 11 parameters (extrinsic and intrinsic) in one vector ϕ, we can parameterize the projection operator:

$$\mathbf{m} = \Pi(\mathbf{w}; \phi). \tag{3}$$

In the absolute orientation problem, given a certain number of $(\mathbf{m}_i, \mathbf{w}_i)$ matching pairs, one want to compute the vector parameters ϕ such that Eq. 3 is satisfied for each pair. Depending on the parameterization chosen for the rotation matrix one obtains linear or non-linear equations [5, 9]. Each correspondence gives two equation, therefore six correspondences are needed at least. A line-to-line correspondence yields the same information (two equations) of a point-to-point correspondence, and the structure of algorithm remains unchanged. (see [14] for more details).

Following [13, 11], we take a different approach, which does not require feature extraction. The pose computation is casted as an optimization problem, where the cost function for an object pose is defined as the integral of the local image gradient along the projected model contours.

3. Method

This section describes our method for tracking arbitrary objects in monocular sequences. At each discrete time step t, the pose of the camera is computed, based on its pose at time $t - 1$. When dealing with dense sequences, the temporal prediction can be accomplished without complex dynamical models. We found that a simple mobile average is sufficient in most cases, and often the time window can be even reduced to be only one sample. The model is projected according to the predicted pose, and the actual camera position and orientation are computed as follows.

We set the problem as the optimization of the following score function

$$E(\phi) = \sum_i |\nabla I(\Pi(\mathbf{w}_i;\phi))| \tag{4}$$

where $\nabla I(\mathbf{x})$ is the image gray level gradient computed at point $\mathbf{x}$. Points $\mathbf{w}_i$ belongs to model edges (both model contour and colour edges). The idea is that a model edge gives rise to a gray level edge in the image, and the score function weight how much the projected points are close to an image edge, identified by gradient maxima. In other word, we want to maximize the amount of image gradient collected by the projected model edges: when the maximum is reached model edges and image edges coincides. Actually, only visible model edges are considered; let Γ_ϕ be the set of the projected visible edge points:

$$E(\phi) = \frac{1}{|\Gamma_\phi|}\sum_{\mathbf{x}\in\Gamma_\phi} ||\nabla I(\mathbf{x})|| = \frac{1}{|\Gamma_\phi|}\sum_{\mathbf{x}} ||\nabla I(\mathbf{x})\chi_{\Gamma_\phi}(\mathbf{x})|| \tag{5}$$

where χ_{Γ_ϕ} is the characteristic function of the set Γ_ϕ, also called *model edge map*. Every point with a non-zero gradient and with $\chi_{\Gamma_\phi} = 1$ gives a contribution to the score function. Normalization is necessary in order to remove bias toward longer edges.

In presence of noisy or textured images it is advisable to take into account the direction of the gradient [11] by projecting it onto the direction normal to the model edge, $\hat{\mathbf{n}}_{\Gamma_\phi}(\mathbf{x})$:

$$E(\phi) = \frac{1}{|\Gamma_\phi|}\sum_{\mathbf{x}} |(\nabla I(\mathbf{x})\cdot\hat{\mathbf{n}}_{\Gamma_\phi}(\mathbf{x}))\chi_{\Gamma_\phi}(\mathbf{x})| \tag{6}$$

In order to have a smooth and larger basin of attraction for the optimum, we consider not only the point belonging to Γ_ϕ, but also their neighbours in a given range, weighted by a Gaussian function. This can be formalized by considering a fuzzy characteristic function $\bar{\chi}_{\Gamma_\phi}$ which decreases with a Gaussian law as points are farther from the actual edge. The point on which the normal is computed is always the centre of the window – which belongs to the model edge.

The use of the gradient direction makes the cost function selective in the neighbourhood of the minimum, and it counteracts the smoothing effect which could flatten too much the cost function near the optimum.

Model projection is obtained by OpenGL. In this way we renounce to a vectorial representation of projected edges, but we gain generality and flexibility. Indeed, our method copes with any type of object, both polyhedral and with smooth boundaries, thanks to the rendering engine of OpenGL. The projection of the wire-frame model (in the case of polyhedral objects) or the boundary of the silhouette (in all the other cases), in raster format, is the model edge map χ_{Γ_ϕ}. The contour normal is computed by fitting a line to each point belonging to Γ_ϕ and its right and left neighbours on the contour.

The optimization strategy is the Hook and Jeeves algorithm [7], a direct search method which belongs to the *pattern search algorithms* class. Direct search methods relies only on the direct comparison of function values, and are particularly suited for situations where the derivatives are unavailable, like in our case. In particular pattern search algorithms are provably convergent under conventional assumptions [3].

Many model tracking methods are restrict to the case of known and fixed intrinsic parameters. Actually, having a model of the object, the problem is equivalent to camera calibration, hence all the 11 parameters can be computed, in principle. The problem is that parameters are correlated, making difficult to attribute individual contributions, and the search space is quite big, compared to R^6.

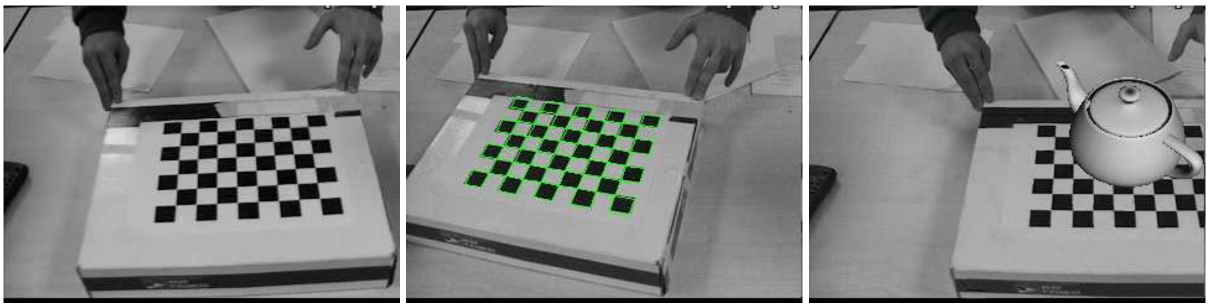
In our algorithm we carry out minimization over seven parameters, the six motion parameters and the focal length. The principal point is taken in the centre of the image, the aspect ratio is set to one and the skew to zero.

The case of varying focal length is treated separately from the estimation of the motion parameters, because in this case the same (or nearly so) score can be obtained by changing the focal length f or the distance of the object from the camera focal plane d. This is true only for orthographic images, but it is approximately true for perspective images. In practice, the minimization algorithm attributes small translations along the optical axis to a change in f and viceversa. If all seven parameters are minimized simultaneously, the values for f change randomly from one frame to another. To counteract this effect, we use a two-step procedure, where first we compute the extrinsic parameters with fixed focal, and then we change f while keeping f/d constant (extrinsic are changed accordingly). The idea is that the first minimization, even with an imprecise estimate for f, computes the right global scale f/d, hence the model edges get roughly overlaid to the image, whereas the second step adjusts the perspective skew, by changing the focal length while leaving the overall scale fixed.

4. Results

For reason of space we show here only a few results. As a model to track we used a metal calibration jig and a checkerboard pattern printed on a paper sheet. The full MPEG sequences, as well as results with other objects, are available on the web at: http://www.sci.univr.it/˜fusiello/demo/mdt.

For each video sequence two movies are produced: 1) a sequence with the model edges (Γ_ϕ) overlaid to the to the image; 2) a sequence with a synthetic teapot (the “Utah teapot”) overlaid to the image. As the camera (or the object)

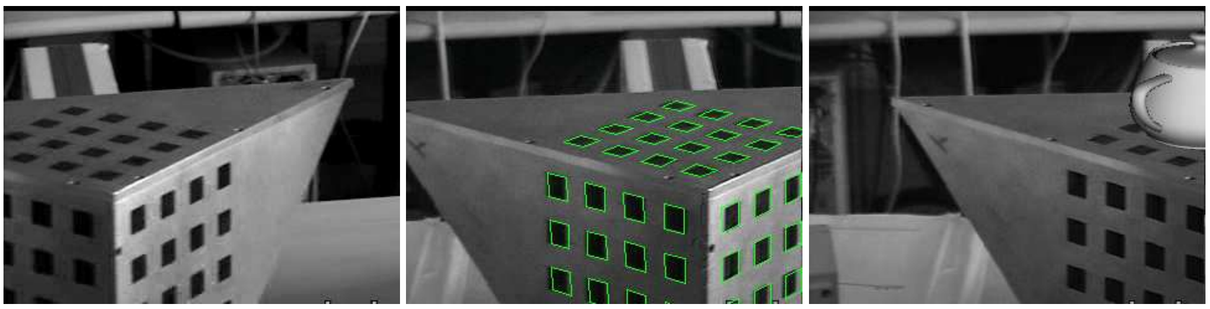


(a) First frame from the original sequence

(b) Central frame with the model overlaid

(c) Last frame with a synthetic teapot overlaid

Figure 1. Checkerboard sequence.

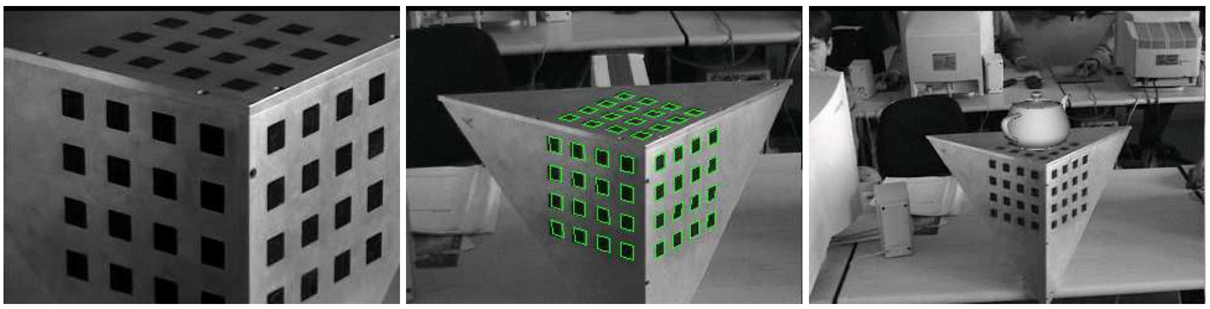


(a) First frame from the original sequence

(b) Central frame with the model overlaid

(c) Last frame with a synthetic teapot overlaid

Figure 2. Panning sequence.



(a) First frame from the original sequence

(b) Central frame with the model overlaid

(c) Last frame with a synthetic teapot overlaid

Figure 3. Zoom sequence.

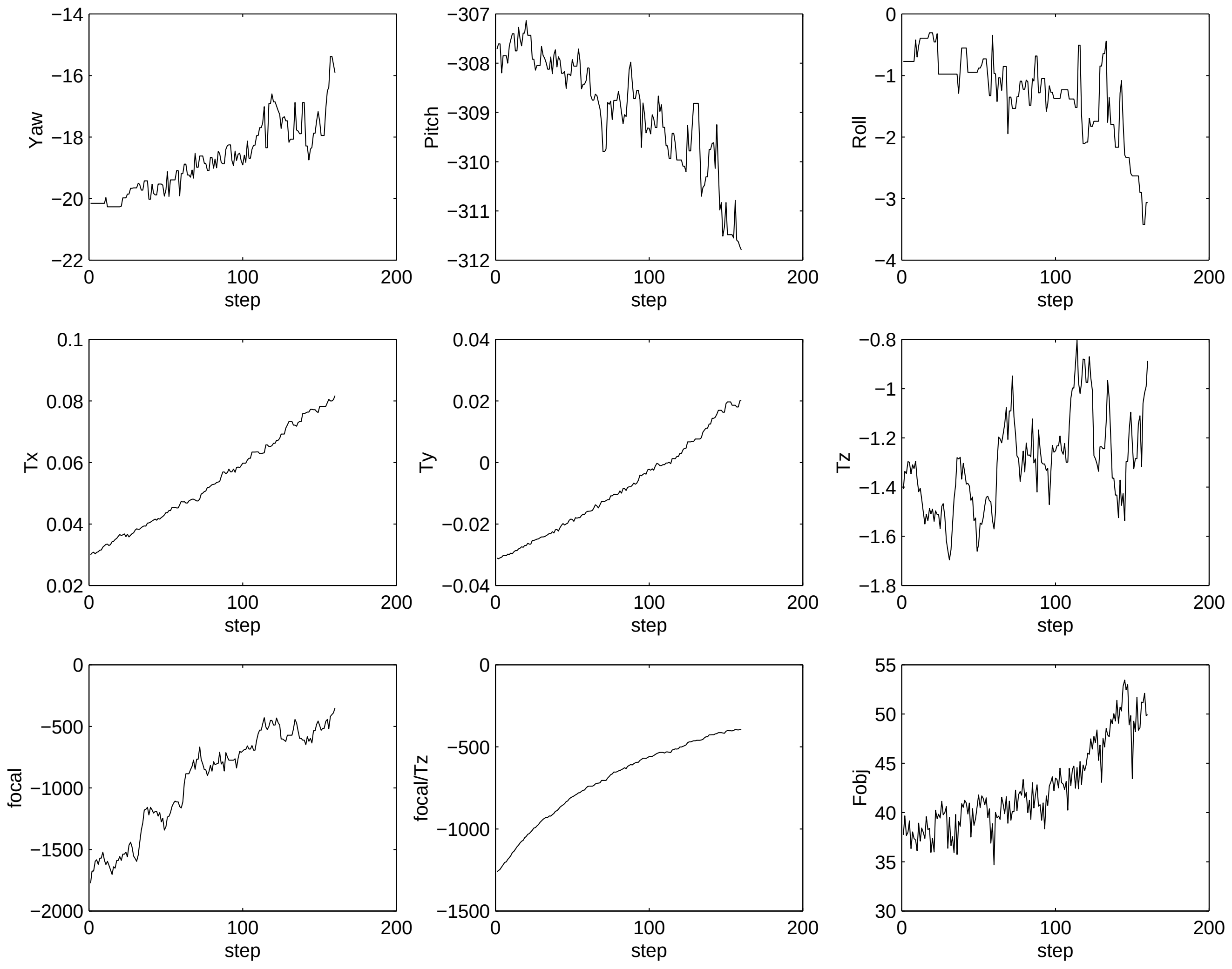


Figure 4. Zoom sequence. Values are in degrees for angles and meters for translations; focal distance is in pixels

moves, the teapot is projected in a way that it appears to stick onto the real object.

Figures 1,2 and 3 shows selected frames from the original sequence and the two synthetic ones. Three results obtained with three different sequences here: the "Checkerboard" sequence, with a planar target and unknown motion (Fig. 1), the "Panning" sequence with a calibration jig and a rotating (*panning*) camera (Fig. 2), the "Zoom" sequence with a calibration jig and a zooming stationary camera (Fig. 3).

Figure 4 shows the values of pose parameters and focal distance recovered by our tracker at each step for the "Zoom" sequence.

The translation $[T_x T y T z]$ represents the position of the origin of the object reference frame in the camera coordinate system. As a single distance d cannot be assigned to the whole object, in practice we consider the distance of the origin of the object reference frame. In this way, d coincides with the T_z component of the translation.

Rotations represent the camera pose in terms of Euler angles (RPY). In this sequence the pose is fixed, so fluctuations of the rotation angles are just noise.

The graph of T_x and T_y can be explained as a compensation of the drift of the image centre that was not allowed to vary in the minimization process. Indeed, it is well known that when the focal length changes in a real camera, not only the focal length of the pinhole model changes, but also the other intrinsic parameters. T_z is very noisy, as the focal length is. Not surprisingly, the ratio f/T_z has a regular graph. This is because f and T_z compensates each other to give anyhow the correct global scale for the object projection. This means that while the scale is computed reliably, f and T_z separately are less reliable.

5. Discussion

Even if parameter values are affected by errors, visual quality, measured as perceived misalignment in the image, is fairly good. This suggests that errors in different parameters compensate each other (like we pointed out for f and T_z) to fit the projected model to the image.

Although we are satisfied with the visual quality of the VR, we plan to study more in depth this phenomenon, and to devise a tracking technique that reduces to the minimum this correlation. In this way, tracking results could be used for robotics and visual servoing tasks, where accuracy of parameters is a concern.

Since we are now dealing with an off-line processing, we did not make any effort to keep computing time to minimum. This issue will be addressed in the next future.

5.1 Acknowledgements

The calibration jig is courtesy of A. Verri, University of Genova. Thanks to Paolo Fiorini for proofreading the manuscript.

References

[1] R. T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 1997.

[2] P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[3] L. Dixon and G. Szego. *The global optimisation problem: An introduction*, pages pages 1–15. Elsevier North-Holland, 1978.

[4] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The John Hopkins University Press, third edition, 1996.

[5] R. M. Haralick, H. Joo, R. C. N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim. Pose Eestimation from Corresponding Point Data. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1426–1446, November-December 1989.

[6] C. Harris. RAPiD - a video rate object tracker. In *British Machine Vision Conference*, pages 73–78, 1990.

[7] R. Hooke and T. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery (ACM)*, pages 212–229, 1961.

[8] M. Kass, A. Witkin, and D. Terzopulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[9] D. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[10] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, August 1992.

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

[12] J. Park, B. Jiang, and U. Neumann. Vision-based pose computation: Robust and accurate augmented reality tracking. In *International Workshop on Augmented Reality*. ACM and IEEE, 1999.

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

[14] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall, 1998.

[15] A. Worrall, G. Sullivan, and K. Baker. Pose refinment of active models using forces in 3d. In *Proceedings of the European Conference on Computer Vision*, pages 341–350, 1994.

[16] P. Wunsch and G. Hirzinger. Registration of cad-models to image by iterative inverse perspective matching. In *Proceedings of the International Conference on Pattern Recognition*, pages 77–83, 1996.