Real-time markerless tracking for augmented reality: the virtual visual servoing framework

Andrew Comport, Eric Marchand, Muriel Pressigout, François Chaumette

Real-Time Markerless Tracking for Augmented Reality: The Virtual Visual Servoing Framework

Andrew I. Comport, *Member*, *IEEE*, Eric Marchand, *Member*, *IEEE*,
Muriel Pressigout, and François Chaumette, *Member*, *IEEE*

Abstract—Tracking is a very important research subject in a real-time augmented reality context. The main requirements for trackers are high accuracy and little latency at a reasonable cost. In order to address these issues, a real-time, robust, and efficient 3D model-based tracking algorithm is proposed for a "video see through" monocular vision system. The tracking of objects in the scene amounts to calculating the pose between the camera and the objects. Virtual objects can then be projected into the scene using the pose. Here, nonlinear pose estimation is formulated by means of a virtual visual servoing approach. In this context, the derivation of point-to-curves interaction matrices are given for different 3D geometrical primitives including straight lines, circles, cylinders, and spheres. A local moving edges tracker is used in order to provide real-time tracking of points normal to the object contours. Robustness is obtained by integrating an M-estimator into the visual control law via an iteratively reweighted least squares implementation. This approach is then extended to address the 3D model-free augmented reality problem. The method presented in this paper has been validated on several complex image sequences including outdoor environments. Results show the method to be robust to occlusion, changes in illumination, and mistracking.

Index Terms—Augmented reality, virtual visual servoing, robust estimators, real-time, model-based tracking, model-free tracking.

1 Introduction

This paper presents a vision-based markerless tracker for Augmented Reality (AR). Many different types of sensors have been used to achieve this, including GPS, gyroscopes, cameras, hybrid vision, accelerometers and others, as reported in [1], [2], [17], [41]. Although the implementation presented here is not restricted to a particular display technology, for this work we use a monocular camera. This study will focus on the tracking techniques that allow alignment in real-time of real and virtual worlds using images acquired by a moving camera.

In this paper, a markerless 3D model-based algorithm is first used for the tracking of objects in monocular image sequences. The main advantage of a model-based method is that the knowledge about the scene (the implicit 3D information) allows improvement of robustness and performance by being able to predict hidden movement of the object and acts to reduce the effects of outlier data introduced in the tracking process. However, since such 3D information is not easily available in certain circumstances, it is sometimes necessary to achieve the pose computation with less constraining knowledge on the viewed scene. We will show that one can take advantage of the geometrical properties of the scene and of the vision system to compensate for the lack of a complete 3D model.

1.1 Real-Time Markerless Tracking

When dealing with 3D camera localization or pose computation, most of the approaches proposed in the literature rely on a 3D registration issue. The most common methods, suitable for AR applications, to compute the pose rely on indoor [3], [26], [39], [53], [55] and outdoor [43] fiducial markers. In the related computer vision literature, geometric features considered for the estimation are often points [18], [9], segments [11], straight lines [29], contours or points on the contours [34], [38], [12], conics [47], [8], cylindrical objects [10], or a combination of these different features [39]. Another important issue is the registration problem. *Purely geometric* (eg, [15], [22], [11]), or *numerical and iterative* [9] approaches may be considered. *Linear approaches* use a least-squares method to estimate the pose [16], [14], [32]. *Full-scale, nonlinear optimization techniques* (e.g., [35], [36], [12]) consist of minimizing the error between the observation and the forward-projection of the model (this process will be described in detail in Section 2.1). In this case, minimization is handled using numerical iterative algorithms such as Newton-Raphson or Levenberg-Marquardt [35]. The main advantage of these approaches is their accuracy. The main drawback is that they may be subject to local minimum and, worse, divergence if initialization is not properly done. These 2D-3D registration techniques rely on the use of a 3D model of the tracked objects. When such 3D models are not available, other approaches are required. If the full sequence is a priori available (such as for postproduction applications), the bundles adjustment technique is considered. This approach consists in the joint estimation of both camera localization and scene structure using a nonlinear minimization technique. Commercial systems such as Boujou from 2D3 or MatchMover from Realviz exploit these very efficient techniques, but they are not suitable for online real-time. Indeed, they are targeted for offline AR applications such as postproduction (since a large number of image are required,

• *The authors are with IRISA-INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France. E-mail: Andrew.Comport@sophia.inria.fr, {Eric.Marchand, Muriel.Pressigout, Francois.Chaumette}@irisa.fr.*

Manuscript received 20 Dec. 2004; revised 9 Nov. 2005; accepted 19 Dec. 2005; published online 10 May 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0260-1204.

1077-2626/06/$20.00 © 2006 IEEE Published by the IEEE Computer Society

the computational cost is prohibitive). Other approaches allow computing both camera displacement and the structure of the scene using on-the-fly techniques based on SLAM approaches (simultaneous location and mapping) [7] or real-time structure from motion [44]. In the case of online AR, it is not always necessary to estimate the pose itself. One can instead compute the relative camera motion [5] via planar homography estimation [51], [50], [46] or optical-flow-based techniques [43]. These methods have been shown to work in real-time and in outdoor environments. However, they are usually restricted to planar surfaces, which may be problematic in complex environments. Another class of AR approaches, referred to as calibration-free approaches, uses affine or projective spaces for the integration of virtual objects [30], [48].

1.2 Robust Tracking

Whatever the method chosen, the solution must deal with the problem of robustness so as to account for noise in real video acquisition, occlusion phenomena, changes in illumination, mistracking, and for any perturbation that may be found in the video. Using a robust low-level feature extraction is certainly useful but usually not sufficient since it is not possible to model all the possible perturbations. It is more efficient to assume that the data is corrupt and use a statistically robust camera pose (or camera displacement) estimation process to address this problem.

In related computer vision and statistics literature, many different approaches exist to treat these external sources of error. Among the robust outlier rejection algorithms, methods in computer vision have included the Hough Transform and RANSAC [15]. These approaches treat the standard deviation of the inlier data (scale) as a constant to be tuned. On the other hand, the statistical methods such as Least Median Square (LMedS) and M-estimators [23] have been developed which treat scale as a parameter to be estimated. Most of these approaches have been used in pose estimation or in camera displacement estimation (see, for example, [15], [18], [29], [50], [54]). The reader is referred to [52] for a review of different robust techniques applied to computer vision.

1.3 Outline of the Paper and Contributions

In this paper, pose computation and camera displacement estimation are formulated in terms of a full scale nonlinear optimization: Virtual Visual Servoing (VVS). This way, the pose computation problem is considered as similar to 2D visual servoing as initially proposed in [53], [39]. Essentially, 2D visual servoing [24], [13], [20] consists of specifying a task (mainly, positioning or target tracking tasks) as the regulation in the image of a set of visual features. A closed-loop control law that minimizes the error between the current and desired position of these visual features can then be implemented, which automatically determines the motion the camera has to realize. This framework is used to create an image-feature-based system which is capable of treating complex scenes in real-time. Advantages of the virtual visual servoing formulation are demonstrated by considering a wide range of performance factors. Notably the accuracy, efficiency, stability, and robustness issues have been addressed and demonstrated to perform in complex scenes. In particular, the interaction matrices that link the virtual camera velocity to the variation of a distance in the image are determined. A robust control law that integrates an M-estimator is considered to improve robustness. The resulting pose computation algorithm is thus able to deal efficiently with incorrectly tracked features that usually contribute to a compound effect which degrades the system until failure. Let us note that in other contexts, techniques with a formulation similar to visual servoing have been presented. This is the case for the ”analysis through image synthesis” techniques [28], [42] derived in the context of structure and motion for video coding.

In the remainder of the paper, Section 2.1 presents the principle of the approach. In Section 2.2, the details of the robust visual servoing control law are shown and a stability analysis is presented. In Section 2.3, the computation of the confidence in the local features extraction is introduced. Section 3 deals with the chosen visual features considered in the 3D model-based tracking process. First, the analytical formulation of the interaction matrices for various features is derived, and then the algorithm used for tracking local features is presented. In Section 4, the approach is extended to address the 3D model-free AR problem. In Section 5, several experimental results are presented.

2 3D Model-Based Tracking Using Robust Virtual Visual Servoing

2.1 Overview and Motivations

As already stated, the fundamental principle of the proposed approach is to define the pose computation problem as the dual problem of 2D visual servoing [13], [24]. In visual servoing, the goal is to move a camera in order to observe an object at a given position in the image. This is achieved by minimizing the error between a desired state of the image features $\mathbf{s}^*$ and the current state $\mathbf{s}$. If the vector of visual features is well chosen, there is only one final position of the camera that allows this minimization to be achieved. An explanation will now be given as to why the pose computation problem is very similar.

To illustrate the principle, consider the case of an object with various 3D features $\mathbf{P}$ (for instance, ${}^o\mathbf{P}$ are the 3D coordinates of object points in the object frame). A virtual camera is defined whose position and orientation in the object frame are defined by $\mathbf{r}$. The approach consists in estimating the real pose by minimizing the error Δ between the set of observed data $\mathbf{s}^*$ (usually the position of a set of features in the image) and the position $\mathbf{s}$ of the same features computed by forward-projection according to the current pose,

$$\Delta = \sum_{i=1}^{N} \left(s_i(\mathbf{r}) - s_i^*\right)^2 = \sum_{i=1}^{N} \left(pr_\xi(\mathbf{r}, {}^o\mathbf{P}_i) - s_i^*\right)^2, \qquad (1)$$

where $pr_\xi(\mathbf{r}, {}^\mathbf{o}\mathbf{P})$ is the projection model according to the intrinsic parameters ξ and camera pose $\mathbf{r}$ and where N is the number of considered features. It is supposed here that intrinsic parameters ξ are available, but it is possible, using the same approach, to also estimate these parameters.

In this formulation of the problem, a virtual camera (initially at $\mathbf{r}_i$) is moved using a visual servoing control law

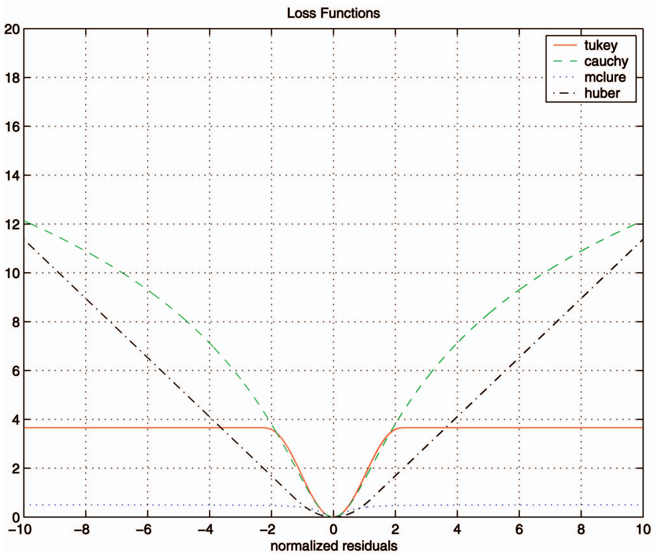


Fig. 1. Robust function $\rho(u)$ for different M-estimators.

in order to minimize this error Δ. At convergence, the virtual camera reaches the pose $\mathbf{r_d}$ which minimizes this error. $\mathbf{r_d}$ is the real camera pose we are looking for.

An important assumption is to consider that $\mathbf{s}^*$ is computed (from the image) with sufficient precision. In visual servoing, the control law that performs the minimization of Δ is usually handled using a least squares approach [13], [24]. However, when outliers are present in the measures, a robust estimation is required. M-estimators can be considered as a more general form of maximum likelihood estimators [23]. They are more general because they permit the use of different minimization functions not necessarily corresponding to normally distributed data. Many functions have been proposed in the literature which allow uncertain measures to be less likely considered and in some cases completely rejected. In other words, the objective function is modified to reduce the sensitivity to outliers. The robust optimization problem is then given by

$$\Delta_{\mathcal{R}} = \sum_{i=1}^{N} \rho\Big(s_i(\mathbf{r}) - s_i^*\Big), \tag{2}$$

where $\rho(u)$ is a robust function [23] that grows subquadratically and is monotonically nondecreasing with increasing $|u|$ (see Fig. 1). Iteratively Reweighted Least Squares (IRLS) is a common method of applying the M-estimator. It converts the M-estimation problem into an equivalent weighted least-squares problem.

To embed robust minimization into visual servoing, a modification of classical control laws is required to allow outlier rejection.

2.2 Robust Control Law

The objective of the control scheme is to minimize the objective function given in (2). This new objective is incorporated into the control law in the form of a weight which is given to specify a confidence in each feature location. Thus, the error to be regulated to zero is defined as

$$\mathbf{e} = \mathbf{D}(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*), \tag{3}$$

where $\mathbf{D}$ is a diagonal weighting matrix given by

$$\mathbf{D} = \mathrm{diag}(w_1, \ldots, w_k).$$

Each w_i reflects the confidence in the ith feature. The computation of weights w_i is described in Section 2.3. If $\mathbf{D}$ was constant, the derivative of (3) would be given by

$$\dot{\mathbf{e}} = \frac{\partial \mathbf{e}}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \mathbf{r}} \frac{\mathbf{dr}}{\mathrm{d}t} = \mathbf{D}\mathbf{L_s}\mathbf{v}, \tag{4}$$

where $\mathbf{v}$ is the camera velocity screw and $\mathbf{L_s}$ is called the interaction matrix related to $\mathbf{s}$. This matrix depends on the value of the image features $\mathbf{s}$ and their corresponding depth Z in the scene (which is available here). If an exponential decrease of the error $\mathbf{e}$ is specified

$$\dot{\mathbf{e}} = -\lambda \mathbf{e}, \tag{5}$$

where λ is a positive scalar, the following control law is obtained:

$$\mathbf{v} = -\lambda (\widehat{\mathbf{D}\mathbf{L_s}})^{+} \mathbf{D}\big(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*\big), \tag{6}$$

where $(\widehat{\mathbf{D}\mathbf{L_s}})^{+}$ is the pseudoinverse[1] of $\widehat{\mathbf{D}\mathbf{L_s}}$, and where $\widehat{\mathbf{D}\mathbf{L_s}}$ is a model of $\mathbf{D}\mathbf{L_s}$.

Different choices are possible for $\widehat{\mathbf{D}\mathbf{L_s}}$:

- the first case is to use the current value of the weight, the feature, and the depth at each iteration:

  $$(\widehat{\mathbf{D}\mathbf{L_s}})^{+} = \big[\mathbf{D}\mathbf{L}_\mathrm{s}(\mathbf{s}, Z)\big]^{+}. \tag{7}$$

  This choice allows the system to follow, as closely as possible, the intended behavior ($\dot{\mathbf{e}} = -\lambda \mathbf{e}$).
- In the second case, a constant interaction matrix is considered using the initial depth $\mathbf{Z_i}$, the initial value of the features $\mathbf{s}_i$, and the first value of the weighting matrix $\widehat{\mathbf{D}} = \mathbf{I}$:

  $$(\widehat{\mathbf{D}\mathbf{L_s}})^{+} = \mathbf{L_s^{+}}(\mathbf{s_i}, \mathbf{Z_i}). \tag{8}$$

  This choice leads to a simpler control law:

  $$\mathbf{v} = -\lambda \widehat{\mathbf{L_s}}^{+} \mathbf{e} = -\lambda \mathbf{L_s^{+}}(\mathbf{s_i}, \mathbf{Z_i}) \mathbf{D}(\mathbf{s} - \mathbf{s}^*). \tag{9}$$

  Note also that, even if (8) is constant, the evolution of the weights during the realization of the control law is taken into account through the computation of $\mathbf{e}$, as in (9). Furthermore, the weights $\mathbf{w_i}(0)$ could be computed instead of choosing them to be equal to 1. However, these initial weights may be equally incorrect.

In both cases, only the local stability can be demonstrated [39] as long as a sufficient number of features will not be rejected (so that $\mathbf{D}\mathbf{L_S}$ is always of full rank). This means that the convergence may not be obtained if the error $\mathbf{s} - \mathbf{s}^*$ is too large. However, in tracking applications, $\mathbf{s}$ and $\mathbf{r}$ are obtained from the previous image; thus, the motion between two successive images acquired at video rate is sufficiently small to ensure the convergence. In practice, it has been observed that the convergence is obtained, in

[1] In our case, since the number of rows is greater than the number of columns, the pseudoinverse of a matrix $\mathbf{A}$ is defined by $\mathbf{A}^{+} = (\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}$, where $\mathbf{A}^{\top}$ is the transpose of $\mathbf{A}$.

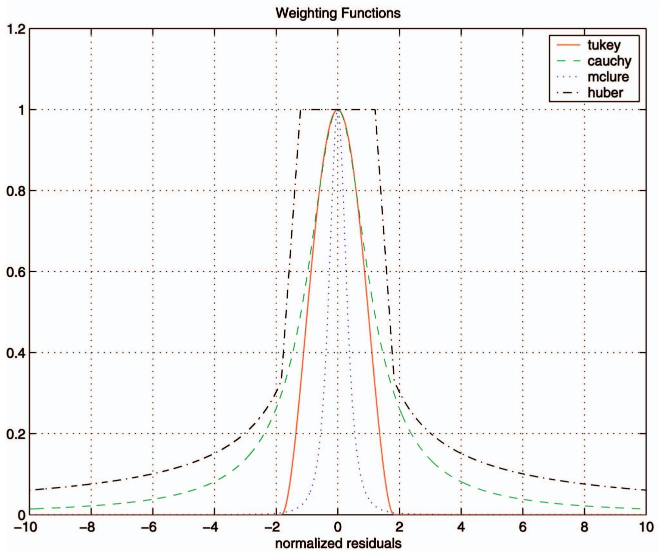


Fig. 2. Weights evolution for different M-estimators.

general, when the camera displacement has an orientation error less that 30 degrees on each axis. Thus, potential problems only appear for the very first image where the initial value for $\mathbf{r}$ may be too coarse. In the current algorithm, the initialization for the very first image is done manually. (An automatic initialization is possible [6], [31] but is out of the scope of this paper.)

Finally, Rodrigues' formula is then used to map the velocity vector $\mathbf{v}$ to its corresponding instantaneous displacement, allowing the pose to be updated. To apply the update to the displacement between the object and camera, the exponential map (see, for example, [37]) is applied using homogeneous matrices, resulting in

$$ {}^{c}\mathbf{M}_{o}^{k+1} = {}^{c}\mathbf{M}_{o}^{k} e^{\mathbf{v}}, \tag{10} $$

where k denotes the number of iterations of the minimization process.

2.3 Computing Confidence

The weights w_i, which represent the different elements of the $\mathbf{D}$ matrix and reflect the confidence of each feature, are given by [23]:

$$ w_i = \frac{\psi(\delta_i/\sigma)}{\delta_i/\sigma}, \tag{11} $$

where $\psi(u) = \frac{\partial \rho(u)}{\partial u}$ is the influence function and δ_i is the normalized residue given by $\delta_i = \Delta_i - Med(\Delta)$ (where $Med(\Delta)$ is the median operator).

Of the various loss and corresponding influence functions that exist in the literature, Tukey's hard redescending function has been chosen. Tukey's function completely rejects outliers and gives them a zero weight. This is of interest in tracking applications so that a detected outlier has no effect on the virtual camera motion. This influence function is given by (see Fig. 2)

$$ \psi(u) = \begin{cases} u(C^2 - u^2)^2, & \text{if } |u| \leq C \\ 0, & \text{else}, \end{cases} \tag{12} $$

where the proportionality factor for Tukey's function is $C = 4.6851$ and represents 95 percent efficiency in the case of Gaussian noise.

In order to obtain a robust objective function, a value describing the certainty of the measures is required. The scale σ is the standard deviation of the inlier data and is an important value for the efficiency of the method. In nonlinear regression for pose computation, this estimate of the scale can vary dramatically during convergence. Scale may be manually chosen as a tuning variable. In our case, we chose to estimate it online. One robust statistic used to estimate scale is the median absolute deviation (MAD), given by

$$ \widehat{\sigma} = \frac{1}{\Phi^{-1}(0.75)} Med_i(|\delta_i - Med_j(\delta_j)|). \tag{13} $$

where $\Phi()$ is the cumulative normal distribution function and $\frac{1}{\Phi^{-1}(0.75)} = 1.48$ represents one standard deviation of the normal distribution. To date, a convergence proof for nonlinear regression using the MAD only exists if it is calculated once as an ancillary scale estimate due to the median's lack of asymptotic properties [21]. However, although convergence has yet to be proved, experimental results show that recomputing the MAD at each iteration gives better results (see Section 5). Let it be noted that more recent and more efficient algorithms have been proposed to estimate scale, but they are not always suitable for real time issue [54].

3 Visual Features for Model-Based Tracking

3.1 Interaction Matrices

Any kind of geometrical feature can be considered within the proposed control law as soon as it is possible to compute its corresponding interaction matrix $\mathbf{L_s}$. In [13], a general framework to compute $\mathbf{L_s}$ has been proposed. It is then possible to compute the pose from a large set of image features (points, lines, circles, distances, etc.) within the same framework. It is also easy to show that combining different features can be achieved by adding features to vector $\mathbf{s}$ and by "stacking" each feature's corresponding interaction matrix into a large interaction matrix of size $nd \times 6$ where n corresponds to the number of features and d their dimension:

$$ \dot{\mathbf{s}} = \begin{bmatrix} \dot{\mathbf{s}}_1 \\ \vdots \\ \dot{\mathbf{s}}_n \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{\mathbf{s}_1} \\ \vdots \\ \mathbf{L}_{\mathbf{s}_n} \end{bmatrix} \mathbf{v} = \mathbf{L_s v}. \tag{14} $$

The redundancy yields a more accurate result with the computation of the pseudoinverse of $\mathbf{L_s}$ as given in (6). Furthermore, if the number or the nature of visual features is modified over time, the interaction matrix $\mathbf{L}$ and the vector error $\mathbf{s}$ are easily modified, consequently. In [39], classical geometrical features (point, straight line, circle, and cylinder) have been considered.

In this paper, the visual features $\mathbf{s}$ are composed of a set of distances between local point features obtained from a fast image processing step and the contours of a more global 3D model. In this case, the desired value $\mathbf{s}^*$ is zero. The assumption is made that the contours of the object in

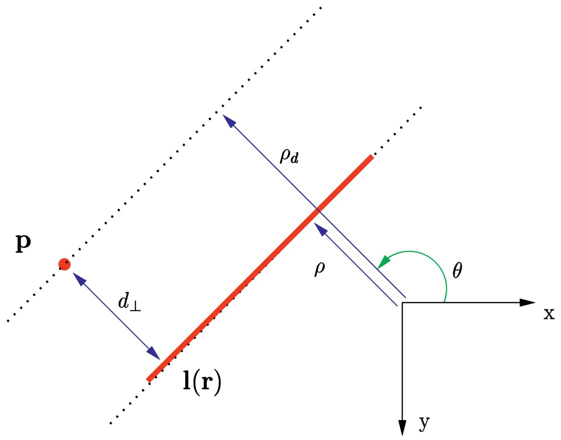


Fig. 3. Distance of a point to a straight line.

the image can be described as piecewise linear segments or portions of ellipses. All distances are then treated according to their corresponding segment or ellipse.

3.1.1 *Case of a Distance to a Line*

The derivation of the interaction matrix that links the variation of the distance between a fixed point and a moving straight line to the virtual camera motion is now given. In Fig. 3, $\mathbf{p}$ is the tracked point and $\mathbf{l}(\mathbf{r})$ is the current line feature position.

The position of the straight line is given by its polar coordinates representation,

$$x\cos\theta + y\sin\theta = \rho, \forall(x,y) \in \mathbf{l}(\mathbf{r}). \tag{15}$$

The distance between point $\mathbf{p}$ and line $\mathbf{l}(\mathbf{r})$ can be characterized by the distance $d_\perp$ perpendicular to the line. In other words, the distance parallel to the segment does not hold any useful information unless a correspondence exists between a point on the line and $\mathbf{p}$ (which is not the case). Thus, the distance feature from a line is given by

$$d_l = d_\perp(\mathbf{p}, \mathbf{l}(\mathbf{r})) = \rho(\mathbf{l}(\mathbf{r})) - \rho_d, \tag{16}$$

where

$$\rho_d = x_d\cos\theta + y_d\sin\theta, \tag{17}$$

with x_d and y_d being the coordinates of the tracked point. Thus,

$$\dot{d}_l = \dot{\rho} - \dot{\rho_d} = \dot{\rho} + \alpha\dot{\theta}, \tag{18}$$

where $\alpha = x_d\sin\theta - y_d\cos\theta$. Deduction from (18) gives $\mathbf{L}_{d_l} = \mathbf{L}_\rho + \alpha\mathbf{L}_\theta$. The interaction matrix related to d_l can be thus derived from the interaction matrix related to a straight line given by (see [13] for its complete derivation)

$$\begin{array}{l} \mathbf{L}_\theta = [\lambda_\theta\cos\theta \quad \lambda_\theta\sin\theta \quad -\lambda_\theta\rho \quad \rho\cos\theta \quad -\rho\sin\theta \quad -1] \\ \mathbf{L}_\rho = [\lambda_\rho\cos\theta \quad \lambda\rho\sin\theta \quad -\lambda_\rho\rho \quad (1+\rho^2)\sin\theta \quad -(1+\rho^2)\cos\theta \quad 0], \end{array} \tag{19}$$

where

$$\begin{aligned} \lambda_\theta &= (A_2\sin\theta - B_2\cos\theta)/D_2, \\ \lambda_\rho &= (A_2\rho\cos\theta + B_2\rho\sin\theta + C_2)/D_2, \end{aligned}$$

and

$$A_2X + B_2Y + C_2Z + D_2 = 0$$

is the equation of a 3D plane to which the line belongs.

From (18) and (19), the following is obtained:

$$\mathbf{L}_{d_l} = \begin{bmatrix} \lambda_{d_l}\cos\theta \\ \lambda_{d_l}\sin\theta \\ -\lambda_{d_l}\rho \\ (1+\rho^2)\sin\theta - \alpha\rho\cos\theta \\ -(1+\rho^2)\cos\theta - \alpha\rho\sin\theta \\ -\alpha \end{bmatrix}^\top, \tag{20}$$

where $\lambda_{d_l} = \lambda_\rho + \alpha\lambda_\theta$.

Let it be noted that the case of a distance between a point and the projection of a cylinder is very similar to this case. Indeed, if the considered 3D object is a cylinder, its projection in the image can be represented by two straight lines (in all nondegenerated cases) and parameters A_2/D_2, B_2/D_2, and C_2/D_2 can have the same value for both lines. More precisely, we have

$$\begin{cases} A_2/D_2 = -X_O/(X_O^2 + Y_O^2 + Z_O^2 - R^2) \\ B_2/D_2 = -Y_O/(X_O^2 + Y_O^2 + Z_O^2 - R^2) \\ C_2/D_2 = -Z_O/(X_O^2 + Y_O^2 + Z_O^2 - R^2), \end{cases}$$

where R is the radius of the cylinder and where X_O, Y_O, and Z_O are the coordinates of the point of the axis of the cylinder the nearest to the camera's optical center.

3.1.2 *Case of a Distance to an Ellipse*

Here, the derivation of the interaction matrix is given which relates the distance between a fixed point $\mathbf{p}$ and an ellipse that results from the projection in the image plane of a moving circle or a moving sphere. If the ellipse is parameterized by its center of gravity and by its moments of order 2 (that is, $(x_g, y_g, \mu_{02}, \mu_{20}, \mu_{11})$), the distance d_e between a point $\mathbf{p}(x, y)$ and an ellipse is defined from the ellipse equation:

$$\begin{aligned} d_e = {} & \mu_{02}x^2 + \mu_{20}y^2 - 2\mu_{11}xy + 2(\mu_{11}y_g - \mu_{02}x_g)x \\ & + 2(\mu_{11}x_g - \mu_{20}y_g)y + \mu_{02}x_g^2 + \mu_{20}y_g^2 \\ & - 2\mu_{11}x_gy_g + \mu_{11}^2 - \mu_{20}\mu_{02}. \end{aligned} \tag{21}$$

The variation of the distance due to the variation of the ellipse parameters are thus given by

$$\begin{aligned} \dot{d}_e = {} & \underbrace{\begin{bmatrix} 2(\mu_{11}(y - y_g) + \mu_{02}(x_g - x)) \\ 2(\mu_{20}(y_g - y) + \mu_{11}(x - x_g)) \\ ((y - y_g)^2 - \mu_{02}) \\ 2(y_g(x + x_g) + x_gy + \mu_{11}) \\ ((x - x_g)^2 - \mu_{20}) \end{bmatrix}^\top}_{\mathbf{L}_{d_e}} \begin{bmatrix} \dot{x_g} \\ \dot{y_g} \\ \dot{\mu_{20}} \\ \dot{\mu_{11}} \\ \dot{\mu_{02}} \end{bmatrix} \\ = {} & \mathbf{L_{d_e}}\mathbf{L_c}\mathbf{v}, \end{aligned} \tag{22}$$

where $\mathbf{L_c}$ is the interaction matrix related to an ellipse and is given in [13].

3.2 Visual Features Low-Level Tracking

When dealing with image processing, the normal displacements are evaluated along the projection of the object model contours using the spatiotemporal Moving

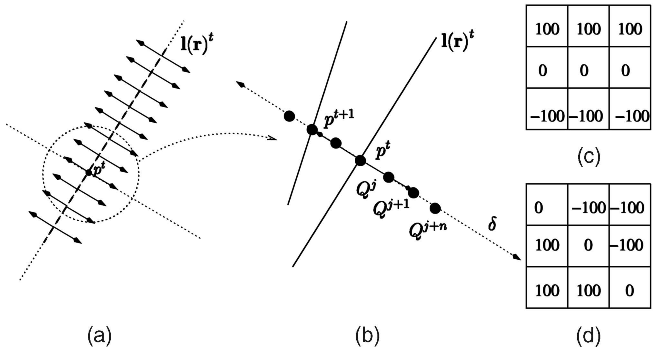


Fig. 4. Determining a point's position in the next image using the ME algorithm: (a) calculating the normal at sample points, (b) sampling along the normal ((c) and (d)) 2 out of 180 3×3 predetermined masks (in practice, 7×7 masks are used), (c) 180 degrees, and (d) 45 degrees.

Edges algorithm (ME) [4]. One of the advantages of the ME method is that it does not require any prior edge extraction. Only point coordinates and image intensities are manipulated. The ME algorithm can be implemented with convolution efficiency and leads to real-time computation [4], [38]. The process consists of searching for the "correspondent" p^{t+1} in image I^{t+1} of each point p^t. A 1D search interval $\{Q_j, j \in [-J, J]\}$ is determined in the direction δ of the normal to the contour (see Figs. 4a and 4b.) For each point p^t and for each position Q_j lying in the direction δ, a criterion ζ_j is computed. This criterion is nothing but the convolution values computed at Q_j using a *predetermined mask* M_δ function of the orientation of the contour (see Figs. 4c and 4d).

The new position p^{t+1} is given by

$$Q^{j^*} = \arg \max_{j \in [-J,J]} \zeta_j \text{ with } \zeta_j = \mid I^t_{\nu(p^t)} * M_\delta + I^{t+1}_{\nu(Q_j)} * M_\delta \mid,$$

where $\nu(.)$ is the neighborhood of the considered pixel.

This low level search produces a list of k points, which are used to calculate distances from corresponding projected contours.

3.3 Uncertainty Propagation

The local ME method described in Section 3.2 determines points along the normal of a contour using a maximum likelihood approach. The decision as to whether or not a spatiotemporal edge exists is made by thresholding the local likelihood value. ζ_{j^*} is chosen to be an edge, providing that it is greater than a threshold λ. This threshold is usually chosen manually and it depends on both the contrast of the contours in the image as well as the size of the mask being applied. A method is presented here to propagate the local likelihood of the points to the global likelihood of the pose. Assuming that the local measure of uncertainty $\zeta_{j^*} \in [0, 1]$ is independent of the global measure of uncertainty w_i, the weights are given by

$$w_{p_i} = w_i \; \zeta_{j^*}, \tag{23}$$

where w_{p_i} is the propagated weight. Matrix $\mathbf{D}$ is then given by

$$\mathbf{D} = \begin{bmatrix} w_{p_1} & & 0 \\ & \ddots & \\ 0 & & w_{p_n} \end{bmatrix}.$$

This has the effect of giving the most certainty to strong contours in terms of the local likelihood and, among those correspondences, the M-estimator converges upon those which conform globally to the 3D shape of the object. Effectively, the robust estimator chooses which correspondences should be considered instead of a manually chosen threshold. This is advantageous when different scenes are considered along with different sized masks.

4 Toward Model-Free Augmented Reality

The previous approach requires a 3D model of the tracked object. Since such 3D knowledge is not always easily available, it is also possible to overcome the pose computation considering less constraining knowledge about the viewed scene. In this section, the proposed method copes with this issue by using, at most, the 2D information extracted from the images and the geometrical constraints inherent in a moving vision system. The objective is therefore to estimate the camera displacement between the capture of two images instead of the camera pose. This can be accurately achieved by minimizing a distance in the image defined using the strong constraints linking two images of the same scene.

4.1 Computing Displacement

As already stated, the fundamental principle of the proposed approach is to define a nonlinear minimization as the dual problem of 2D visual servoing [24]. Displacement computation is a very similar issue. When the 3D model is no longer available, the method presented in the previous section cannot be considered and the projection of the 3D model that was used in (1) has to be replaced by another formulation.

Whereas for pose estimation the goal is to minimize the error between the features observed in the image and their forward projection in the image plane, for camera motion estimation the idea is to minimize the error between the position of the observed features in the second image ($\mathbf{s_2}$) and the position of the corresponding features in the first image transferred in the second one through a camera displacement. Equation (1) is then replaced by

$$\Delta = \sum_{i=1}^{N} \left(\mathbf{s}_{2_i} - {}^2tr_2(\mathbf{s}_{1_i}) \right)^2 \tag{24}$$

where N is the number of considered features. In this formulation of the problem, from the current pose corresponding to the first image, a virtual camera is moved to reach the unknown pose corresponding to the second image by a visual servoing control law minimizing this error Δ. At convergence, the virtual camera has realized the displacement $\widehat{{}^2\mathbf{M}_1}$ which minimizes this error (${}^2\mathbf{M}_1$ will be the real camera displacement).

In the more realistic case where image measurement errors occur in both images, it is better to minimize the errors in both images and not only in one. We then have to consider the forward (${}^2tr_1$) and backward (${}^1tr_2$) transformation. The criterion to be minimized is then

$$\Delta = \sum_{i=1}^{N} {}^2\mathbf{e_1}_i^2 + {}^1\mathbf{e_2}_i^2, \tag{25}$$

where ${}^2\mathbf{e}_{1_i} = \mathbf{s}_{2_i} - {}^2tr_1(\mathbf{s}_{1_i})$ and ${}^1\mathbf{e}_{2_i} = \mathbf{s}_{\mathbf{1}_\mathbf{i}} -{}^1 tr_2(\mathbf{s}_{2_i})$. Minimizing this criterion is equivalent to minimizing the error vector,

$$\mathbf{e} = ({}^2\mathbf{e}_{1_1}, {}^1\mathbf{e}_{2_1}, \ldots, {}^2\mathbf{e}_{1_i}, {}^1\mathbf{e}_{2_i}, \ldots, {}^2\mathbf{e}_{1_N}, {}^1\mathbf{e}_{2_N}), \tag{26}$$

by the following control law:

$${}^\mathbf{c}\mathbf{v} = -\lambda \widehat{\mathbf{L}}_\mathbf{e}^+ \mathbf{e}, \tag{27}$$

where ${}^\mathbf{c}\mathbf{v}$ is the velocity of the virtual camera (expressed in its current frame), and where $\mathbf{L_e}$ is the interaction matrix related to the error vector, such as

$$\widehat{\mathbf{L}}_\mathbf{e} = \left(\cdots, \widehat{\mathbf{L}}({}^2\mathbf{e}_{1_i}), -\widehat{\mathbf{L}}({}^1\mathbf{e}_2){}^1\widehat{\mathbb{T}}_c, \cdots\right). \tag{28}$$

$\mathbf{L}({}^k\mathbf{e}_{l_i})$ is the interaction matrix that links the variation of the error ${}^k\mathbf{e}_{l_i}$ to the virtual camera velocity such that ${}^k\dot{\mathbf{e}}_{l_i} = \mathbf{L}({}^k\mathbf{e}_{l_i}){}^k\mathbf{v}$. ${}^1\mathbb{T}_c$ is the velocity transformation matrix from camera 1 frame to current camera frame, given by the following 6×6 matrix:

$${}^1\mathbb{T}_c = \begin{bmatrix} {}^1\mathbf{R}_c & [{}^1\mathbf{t}_c]_\times {}^1\mathbf{R}_c \\ \mathbf{0}_{3\times3} & {}^1\mathbf{R}_c \end{bmatrix},$$

where$[\mathbf{t}]_\times$ is the skew matrix related to the vector $\mathbf{t}$.

As in the previous section, if data are corrupted by noise, the statistical techniques of robust M-estimation [23] described in Section 2.2 can be introduced within the minimization process. Equation (27) is then replaced by

$${}^\mathbf{c}\mathbf{v} = -\lambda(\widehat{\mathbf{D}}\widehat{\mathbf{L_e}})^+ \widehat{\mathbf{D}}\mathbf{e}, \tag{29}$$

where $\widehat{\mathbf{D}}$ is computed as in Section 2.3.

4.2 Points Transfer

Let us note that in the general case (that is, a nonplanar scene viewed by a camera which rotates and translates), the point transfer can be achieved, using multiple images, considering the epipolar geometry and the essential or fundamental matrices (see, for example, [19]). This section will be restricted to the less general case where point transfer can be achieved using an homography. Indeed, some particular cases (planar scene, pure rotational camera motion) lead the 2D transformation between two images to be a homography. In that case, a point ${}^1\mathbf{p}$ expressed in homogeneous coordinates ${}^1\mathbf{p} = ({}^1x, {}^1y, {}^1w)$, is transferred in image 2 as a point ${}^2\mathbf{p}$, considering the following relation:

$${}^2\mathbf{p} = {}^2tr_1({}^1\mathbf{p}) = \alpha {}^2\mathbf{H}_1 {}^1\mathbf{p}, \tag{30}$$

where ${}^2\mathbf{H}_1$ is a homography (defined up to scale factor α) that defines the transformation between the images acquired by the camera at pose 1 and 2. Once a camera displacement is generated, the homography ${}^2\mathbf{H}_1$ is given by [19]:

$${}^2\mathbf{H}_1 = \left({}^2\mathbf{R}_1 + \frac{{}^2\mathbf{t}_1}{{}^1d} {}^1\mathbf{n}^\top\right), \tag{31}$$

where ${}^1\mathbf{n}$ and ${}^1d$ are the normal and distance to the origin of the reference plane expressed in camera 1 frame (these parameters are assumed to be known in the very first image and updated using the estimated camera displacement; see Section 4.3. Using ${}^2\mathbf{p} = ({}^2x, {}^2y, {}^2w) = {}^2\mathbf{H}_1 {}^1\mathbf{p}$, we finally get $(x_i, y_i) = ({}^2x/{}^2w, {}^2y/{}^2w)$ that is used for the next iteration of the control law. The backward transformation is obtained in a similar way using the inverse homography ${}^1\mathbf{H}_2 = {}^2\mathbf{H}_1^{-1}$.

The interaction matrix $\mathbf{L}({}^\mathbf{1}\mathbf{e_2})$ is the interaction matrix that links the variation of the point position to the camera motion [13], [24]:

$$\mathbf{L}({}^2\mathbf{e}_{1_i}) = \begin{pmatrix} -1/Z_i & 0 & x_i/Z_i & x_iy_i & -(1+x_i^2) & y_i \\ 0 & -1/Z_i & y_i/Z_i & (1+y_i^2) & -x_iy_i & -x_i \end{pmatrix}. \tag{32}$$

The depth information Z_i is computed at each iteration from the coordinates (x_i, y_i) and from the equation of the reference plane updated from the current camera displacement:

$$1/Z_i = \frac{{}^1d - {}^2\mathbf{t}_1 {}^1n}{{}^2\mathbf{R}_1 {}^1\mathbf{n}\ {}^2\mathbf{p}}.$$

4.3 Application to Augmented Reality

For augmented reality applications, the pose between the camera and the world coordinate system is required. If an initial pose ${}^1\widehat{\mathrm{M}}_W$ is known, computing the current pose from the estimated displacement is straightforward:

$${}^n\widehat{\mathrm{M}}_W = {}^n\widehat{\mathrm{M}}_{n-1} {}^{n-1}\widehat{\mathrm{M}}_W. \tag{33}$$

Computing ${}^1\widehat{\mathrm{M}}_W$ requires the introduction of 3D information. Therefore, it has been decided to estimate this first pose from the image of a rectangle in the first image following the approach presented in [51]. The only 3D information required is a rectangle in the very first image and the length of one of its sides.

Drift is inherent to this kind of approach since estimated camera displacements are successively integrated. To limit the drift, it is possible to compute the motion no longer between two successive frames as in (33), but between the current frame and a reference frame (say, frame R_0):

$${}^n\mathrm{M}_W = {}^n\mathrm{M}_{R_O} {}^{R_O}\mathrm{M}_W.$$

To avoid convergence problems due to a too large displacement between R_0 and R_n, it is still possible to consider ${}^{n-1}\mathrm{M}_{R_0}$ to initialize the estimation of ${}^n\mathrm{M}_{R_0}$. Using such a method, there is no drift since there is no integration of the motion (although errors due the low-level tracking process may remain). Obviously, such an approach can be considered only if all the tracked points remain in the camera's field of view. This is not the case in the sequence presented in the next section, for example. In that case, when the number of tracked points is no longer sufficient (say, 50 percent of the initial number of points), a new reference image is chosen and new points are extracted.

$${}^n\mathrm{M}_W = {}^n\mathrm{M}_{R_k} {}^{R_k}\mathrm{M}_{R_{k-1}} \ldots {}^{R_1}\mathrm{M}_{R_O} {}^{R_O}\mathrm{M}_W.$$

This equation is similar to (33), but the number of reference images is much smaller than the number of acquired images, which limits drastically potential drift.

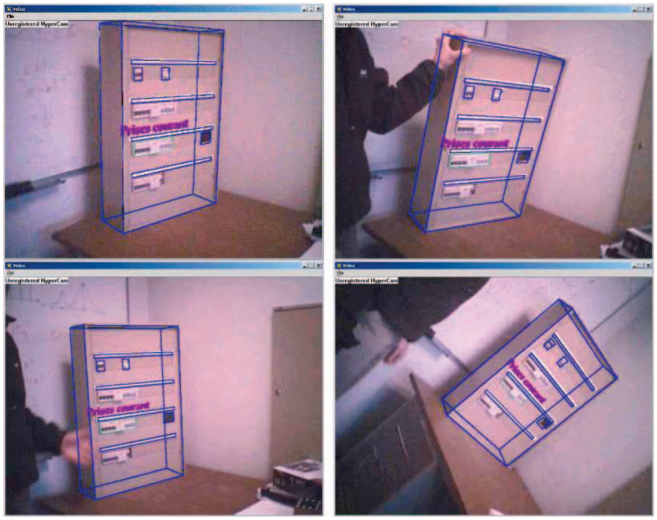


Fig. 5. Tracking of an electrical panel: The sequence features partial occlusions and large motion of both the camera and of the object.

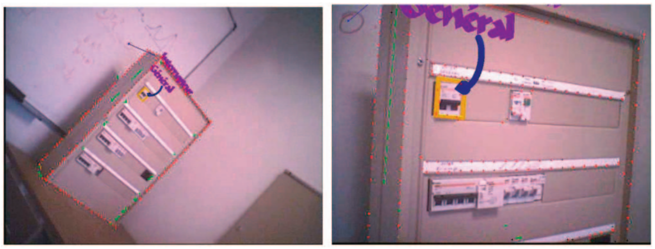


Fig. 6. Tracking of an electrical panel: illustration of the dynamic modification of the model (some parts are added or removed according to the relative camera/object position).

5 Experimental Results

In this section, we report various AR experiments carried out using the proposed virtual visual servoing approach. Dealing with 3D model-based tracking, two software environments have been developed: Marker (Marker-based Augmented Reality KERnel) and Markerless (Markerless-based Augmented Reality KERnel). The former is used for AR applications that uses fiducial markers (results are given in [39]), while the latter is dedicated to markerless tracking (see results in Section 5.1). These software environments have been implemented in C++ under both Linux and Windows XP and are now part of the D'Fusion software proposed by the Total Immersion company. In all the reported experiments about 3D model-based AR, the tracking and 3D localization are achieved at 50 Hz on a simple PC (3.0 Ghz) running Windows XP. Tracking is achieved at least at 15 Hz on a PC (3.0 Ghz) running Linux when dealing with model-free AR experiments (see results in Section 5.2).

5.1 Markerless Augmented Reality

The goal in this paper is not to build a complete AR-based scenario but just to illustrate the fact that the proposed tracker is robust enough to provide the necessary information to build such a system. As stated previously, one of the advantages of this algorithm is its robustness with regard to partial occlusions, to lighting variations, to important and fast camera motion, etc. This is achieved due to both a robust low-level image processing algorithm that provides a real-time and reliable feature matching algorithm and the use of robust M-estimators in the 2D-3D registration process. The following results try to illustrate these points, among others, with both quantitative and qualitative evaluation.

5.1.1 Qualitative Evaluation

In the first example (see Figs. 5 and 6), the markerless tracking algorithm is used to track a domestic or industrial object that can be subject to a maintenance process. An electrical panel is considered and is subject to large occlusions and large camera self-motion. Forward projection of the model appears in blue and the image is augmented with various textual information. The complete video is available online (see Section 5.3).

The model of the object is represented as a hierarchical scene graph (encoded in VRML) that allows online modification of the model used in the tracking process. Since the relative position between the camera and the tracked object is subject to large movements, it is indeed necessary to select automatically online the feature to use. Indeed, when the camera moves away from the object, some details become too small to be useful for tracking (or may even induce a mistracking). In Fig. 6, the white bands (modeled by two lines in the CAD) are used when the camera is close to the object (see the red points on the right image and on the video), while when the camera moves away from the object (left image), these two lines are too close to each another to provide reliable information. Finally, in order to maintain tracking at video rate, the number of tracked points is restricted to be lower than 400 ($\mathbf{L_s}$ is then a 400×6 matrix). This is achieved by constantly modifying the sampling density of points along the lines of the model.

To demonstrate the robustness of the algorithm, a small nonconvex object was placed in a highly textured environment (an electronic board) as shown in Fig. 7. Multiple temporary and partial occlusions were also made by a hand, and self-occlusions of the object itself were imposed during the tracking. Despite these difficulties, tracking was correctly achieved and once again images were acquired and processed at video rate.

In Fig. 8, a chair is tracked and a virtual Lego-man is augmented in the view. The model of the chair is very rough. Each leg of the chair is only modeled by a simple line in the VRML model (and the back of the chair by a simple polygon). The motion of the chair in the video stream was very fast.

Other experiments have been carried out. In Fig. 9, the tracker has been used to track a castle ruin (the tracked object and the lines used in the tracking are shown in red). An augmented model of a car has then been rendered, and its motion is handled using an animation engine (ODE). The

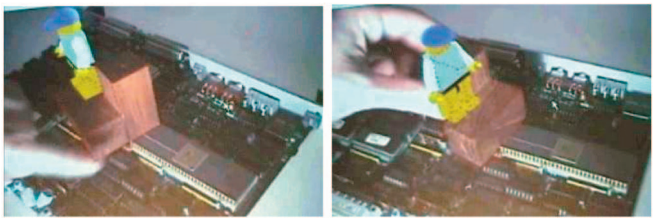


Fig. 7. Tracking a nonconvex object on a highly textured environment.

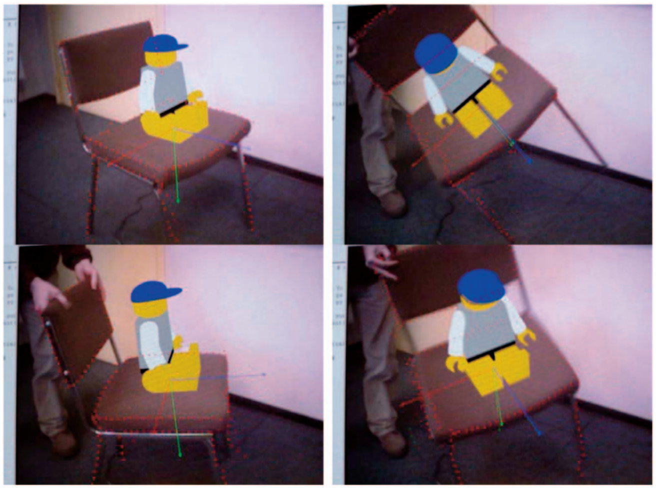


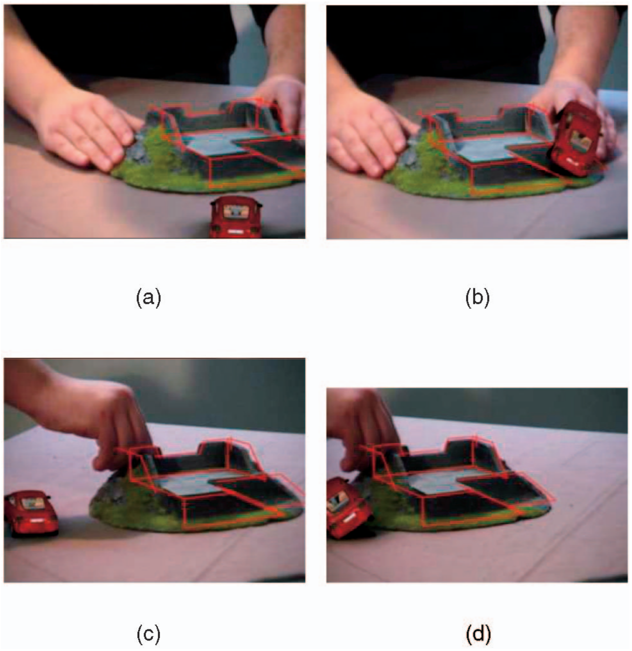
Fig. 8. The sited Lego-man: tracking a chair. This experiment illustrates the robustness of the algorithm with respect to rough CAD model as well as to fast object motion.

car is subject to gravity and to collision with the ruin model. The ruin model moves accordingly to the real ruin motion and the car interacts with its environment in a realistic way.

Fig. 10 shows the results of various other tracking and AR experiments carried out using our real-time markerless tracker.

5.1.2 Quantitative Evaluation

All the previous experiments demonstrate the qualitative capabilities of our tracking algorithm in AR experiments. We now provide numerical results that measure and quantify the tracking accuracy.



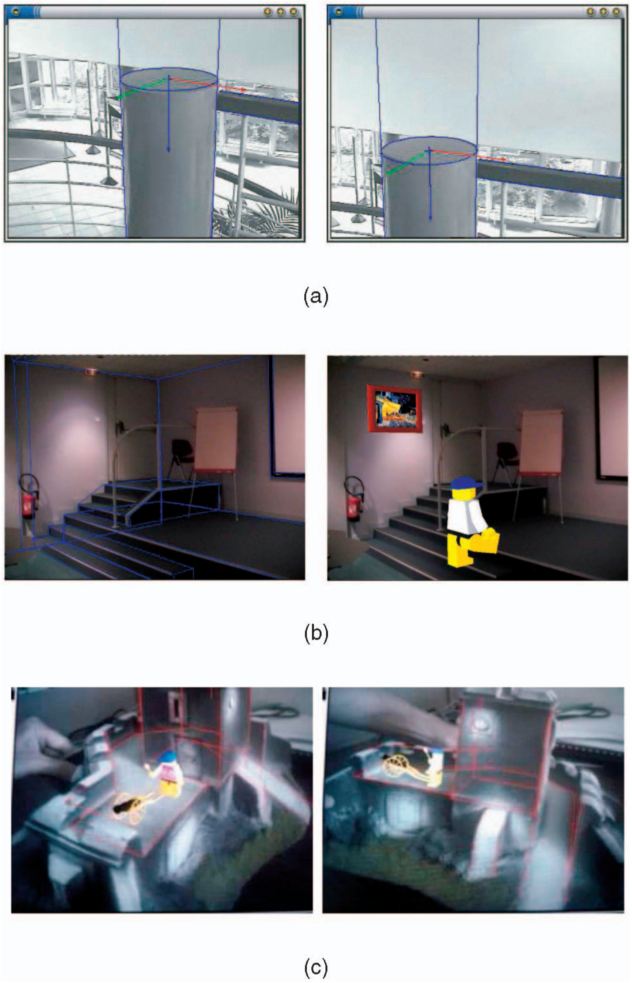
(a)

(b)

(c)

(d)

Fig. 9. Tracking a castle ruin. The virtual red car augments the video and interacts with its environment thanks to an animation engine (ODE). Images thanks to Total Immersion.



(a)

(b)

(c)

Fig. 10. Other tracking and AR experiments using Markerless. (a) Tracking a cylinder, a circle, and lines. (b) Tracking and augmenting a conference room. (c) Tracking and augmenting a castle.

In this experiment, we have mounted the camera on the end-effector of an industrial robot and computed the camera displacement between an initial position using both our algorithm and the odometry of the robot. In both cases, the camera displacement is computed in the same frame that corresponds to the initial camera location. Since the odometry of this particular robot is very precise, we can consider it as a ground truth. The tracked object can be seen on Fig. 11. The robot (and then the camera) achieves a complex motion (the six degrees of freedom are modified), makes a short pose, and then moves back to its initial position. This motion can be seen in Figs. 12a (translation in

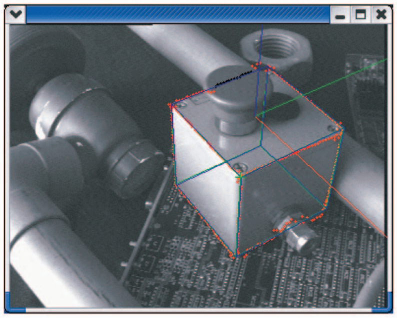
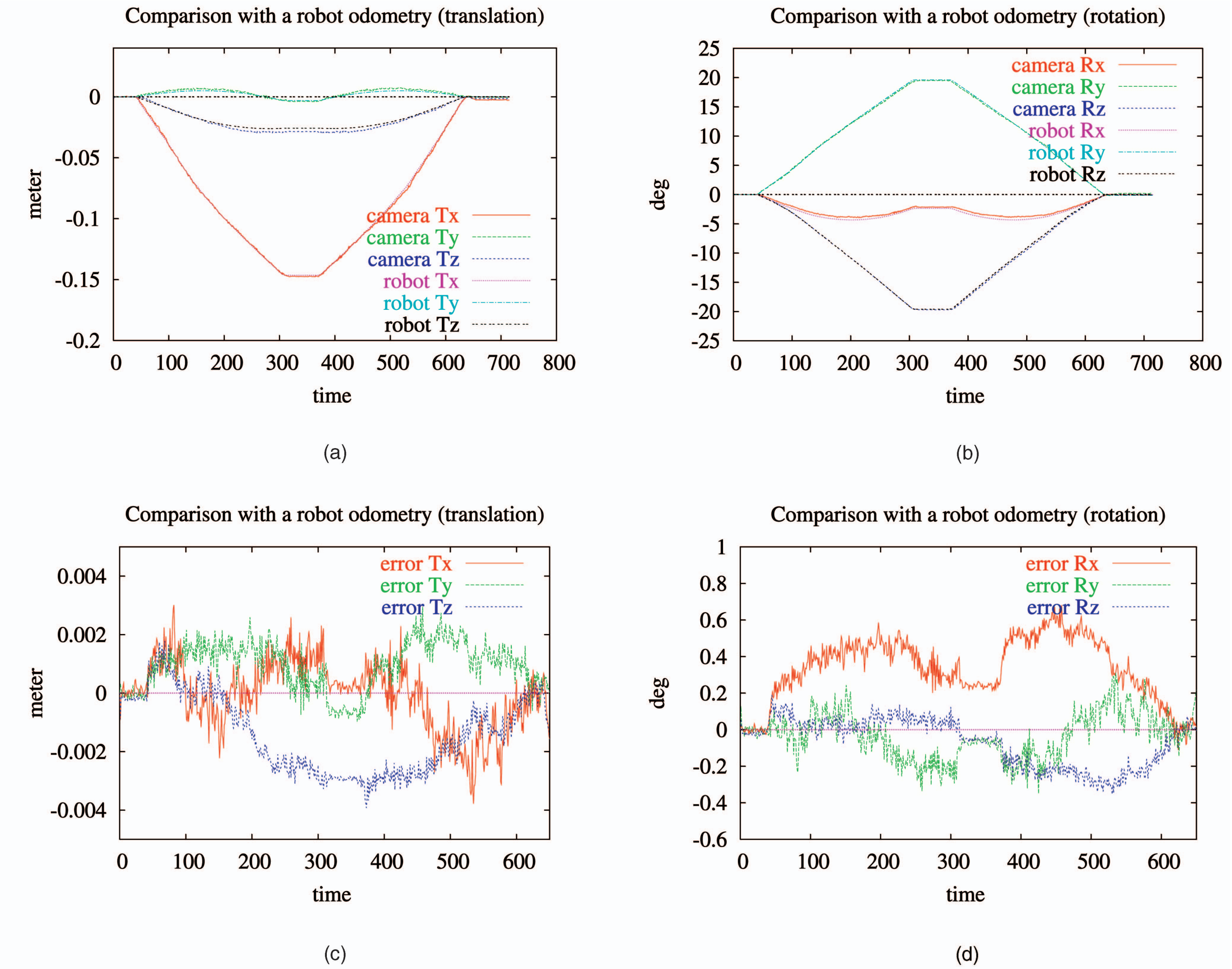


Fig. 11. Quantitative evaluation of the accuracy of the tracker: tracked object.



(a)

(b)

(c)

(d)

Fig. 12. Accuracy of the tracker: ((a) and (b)) camera displacement computed using the robot odometry and the tracker, and ((c) and (d)) corresponding errors (meters, degrees).

meters) and 12b (rotation in degrees), where the camera and robot displacement are plotted. As can be seen the plots are, as expected, well superimposed. The error between the two computed positions is plotted in Figs. 12c (translation in meters) and 12d (rotation in degrees). The average error is 1 mm in translation and 0.2 degree in rotation (with a standard deviation of 1 mm and 0.15 degree) while the tracked object was 1 meter from the camera. More detailed results can be found in Table 1. It can be noted that, despite a rough calibration of the camera, the maximum error is less than 4 mm in translation and 0.35 degrees in rotation. Fig. 13 plots the residual given by (1) and (2). As can be expected, the use of M-estimation greatly improves the minimization and the pose estimation processes.

TABLE 1
Accuracy of the Tracker: Mean Error (Millimeters and Degrees), Standard Deviation, and Maximum Error

	T_x	T_y	T_z	R_x	R_y	R_z
mean	-0.1	0.9	-1.4	0.33	-0.03	-0.06
std	1.1	0.8	1.3	0.17	0.12	0.13
max	3.7	2.9	3.9	0.6	0.34	0.35

It has to be noted that this tracker has some limitations: First, as a nonlinear minimization method, the initial pose has to be known up to a given uncertainty (the experimental cone of convergence is about 30 degrees). Second, the maximum speed of the camera or the object relies on a

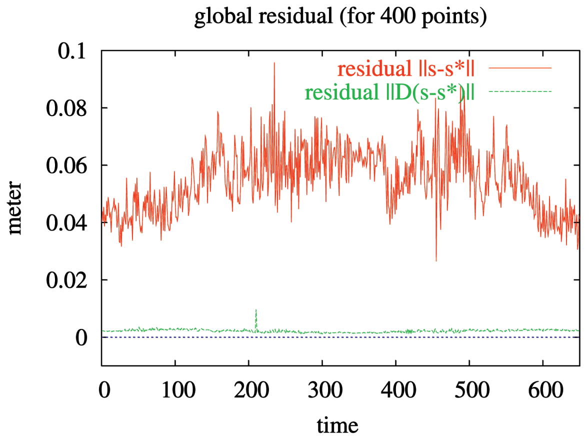


Fig. 13. Accuracy of the tracker: global residual $\|\mathbf{s}-\mathbf{s}^*\|$ and $\|\mathbf{D}(\mathbf{s}-\mathbf{s}^*)\|$.

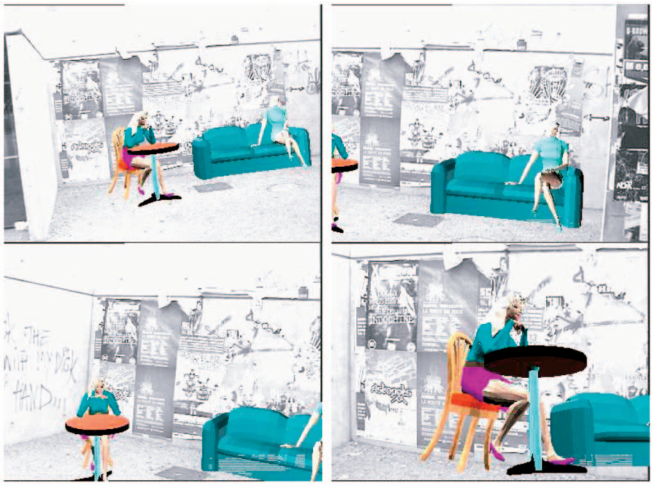


Fig. 14. Three-dimensional motion estimation on an outdoor sequence for a planar structure: augmented images obtained after the motion estimation using the robust VVS approach.

trade-off between real-time calculation and the search distance normal to the contour. With current computing power this distance is quite large (10 to 15 pixels). Finally, a lack of contrast around contours and too large occlusions are classical failure modes. Nevertheless, such a method may be considered within a more complex system with various trackers that rely on other sensors (e.g., as in [27]).

5.2 Model-Free Augmented Reality Results

In these experiments, a set of points are tracked over frames using the Shi-Tomasi-Kanade points tracker [49]. Such trackers allow quite robust tracking at nearly frame rate (15 Hz). From point correspondences, the 3D motion is computed using the algorithm presented in Section 4. In these experiments, the same camera is not always used for all the demonstrations. However the same set of intrinsic parameters is used for all of them. This demonstrates the robustness of the algorithm to calibration errors.

In the experiment reported in Fig. 14, the estimation of the camera motion is considered from a planar outdoor scene. The background wall is the planar scene from which points are tracked to estimate the 3D displacement between two successive images. The method presented in [51] (see Section 4.3) has been used to estimate the initial pose. Although it is not very accurate, it provides a good enough result for AR. The pose computation resulting from this initial pose estimation and the displacement estimations provide realistic augmented video sequence as can be seen in Fig. 14. The objects remain stable in the scene and very little jittering is observed.

In the next experiment (see Fig. 15), the estimation of a pure rotational camera motion is considered. An outdoor scene is first considered. Since the camera undergoes a pure rotational motion, any point in the scene can be tracked and used in the motion estimation process (the planarity constraint is no longer necessary). The camera was mounted on a tripod, which brings some perturbations in the motion estimation since the rotation axes were not exactly those of the camera (there are indeed some small translations in the real camera motion). Fig. 15 shows that even after 800 images, the error in displacement computation is very small. This is visible when comparing the images at the beginning and at the end which view the same scene after a long displacement. It should be pointed out that the complete change of background during the sequence does not disturb the results (in other words, no part of the scene visible in the initial image is visible in the final frame). Let it be noted that, when robust estimation is not considered, the estimation error is noticeable and important drift is clearly visible in the motion integration.

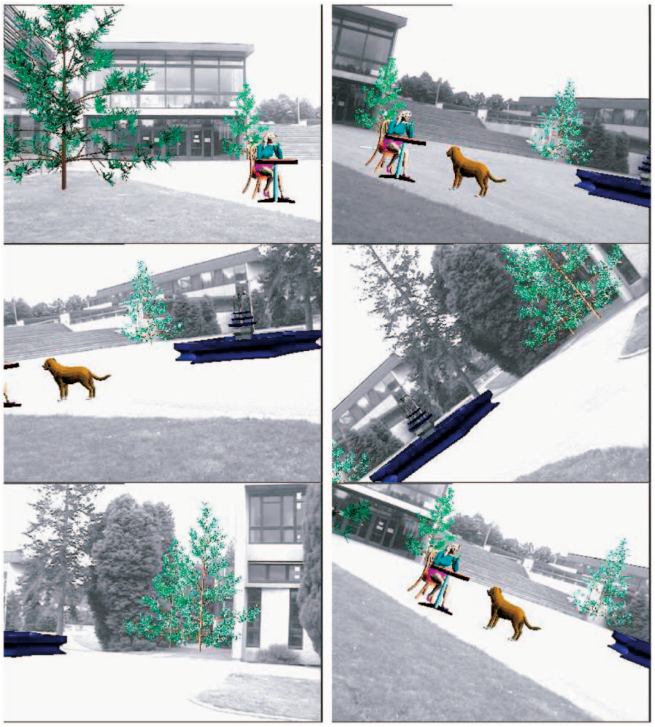


Fig. 15. Model-free AR on an outdoor sequence.

Two comparisons (for the image sequence of Fig. 14) have been made on the remaining error between the image points and the projection of the corresponding points in the other image for the estimated displacement (see Fig. 16). The presented method is first compared using the robust kernel and without. It can be noticed that after a while, the use of M-estimator gives really better displacement estimations. It is then compared with a linear algorithm, i.e., the DLT algorithm using the data normalization as recommended in [19]. It is undeniable that the presented method, even without its robust kernel, is far more efficient, mainly due to the fact that the error is minimized in both images.

5.3 Resources

Videos of the presented results (and more) are available on the Lagadic project Web site[2] (demonstrations section). The source code for the pose estimation by virtual visual servoing using point feature is available in the ViSP2 package [40] and can be downloaded from http://www.irisa.fr/lagadic/visp.

6 Conclusion and Future Perspectives

This paper has presented a framework for conception of vision-based augmented reality systems by considering either 3D model-based tracking techniques or 3D model-free tracking approaches. In each case, the method relies on

[2] http://www.irisa.fr/lagadic.

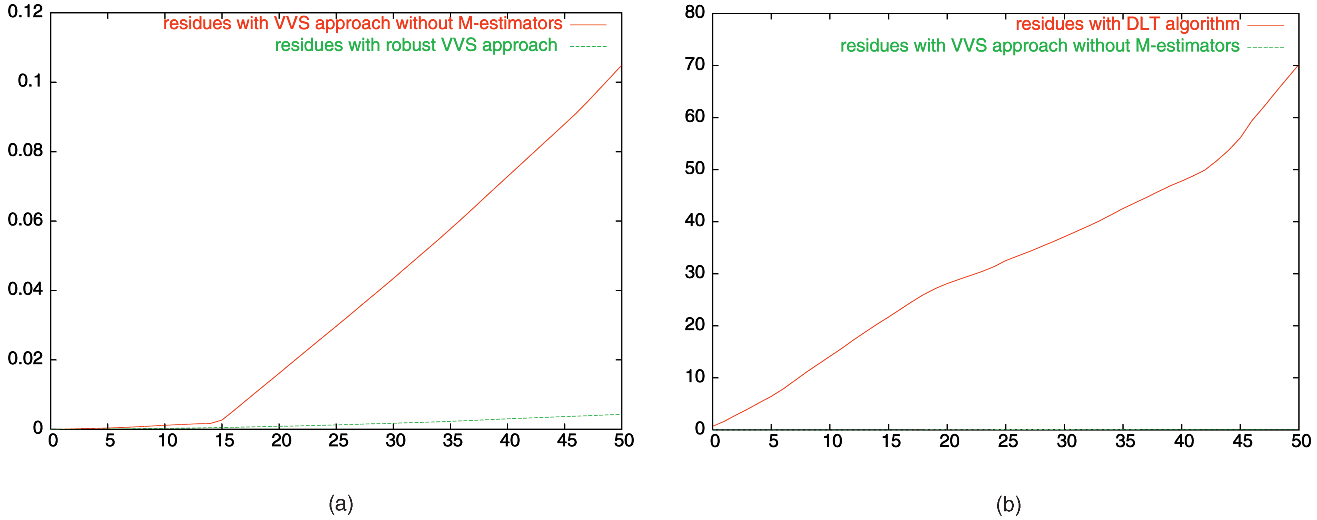


Fig. 16. Comparison with other method (for the image sequence of Fig. 14): (a) VVS without M-estimators (red) versus robust VVS (green). (b) DLT algorithm (red) versus VVS without M-estimators (green).

the minimization of a cost function expressed in the image and this minimization is achieved via a visual servoing control law. Contributions can be exhibited at three different levels:

- The derivation of the interaction matrix for distances to lines and ellipse. Determining an accurate approximation of the interaction matrix is essential to ensure the convergence of a visual servoing task. A general framework for derivation is obtained by taking advantage of the duality with visual servoing methodology. Furthermore, computational efficiencies are obtained by "stacking" and using a constant interaction matrix.
- The widely accepted statistical techniques of robust M-estimation [23] are employed. This is introduced directly in the control law by weighting the confidence on each feature. The resulting pose computation algorithm is thus able to deal efficiently with incorrectly tracked features that usually contribute to a compound effect which degrades the system until failure.
- Three-dimensional model-based tracking relies on correspondences between local features in the image and the object model. In an image stream, these correspondences are given by the local tracking of features in the image sequence. In this paper, low level tracking of the contours is implemented via the Moving Edges algorithm [4]. A local approach such as this is ideally suited to real-time tracking due to an efficient 1D search normal to a contour in the image. In a "real world" scenario, some edges may be incorrectly tracked, due to occlusion, changes in illumination and mistracking. Since many point-to-curve correspondences are made, the method given here has many redundant features which favors the use of robust statistics. Furthermore, a method is proposed for propagating uncertainty from the local edge features to a global pose determination algorithm, which means that no arbitrary predetermined edge detection threshold is necessary.
- The algorithm is extended to address a model-free AR scenario. This involves estimating the camera displacement between two successive images instead of the camera pose. This is accurately achieved by minimizing an error in the image defined using the strong constraints linking two projections in successive images of the scene.

The algorithm has been tested on various images sequences and for various applications which demonstrates a real usability of this approach. Each time tracking is handled in real time (up to 50 Hz for the 3D model-based approach and at least at 15Hz for the 3D model-free approach).

The algorithm presented here has few limitations that need to be addressed in the future. First, it relies on a coarse manual initialization on the very first image. A fast and automatic initialization would be necessary. We have presented such a method in [6], where SIFT points [33] were used along with reference images of the object. Considering matched points, the camera transformation between acquired and reference images can be computed. Other efficient methods may be found in the literature to address this issue (e.g., recent paper [31]). Second, in this study, the spatiotemporal aspect of the tracking process has not been considered in depth. Indeed, robustness can also be handled from one time-step to another, as is possible in a Bayesian framework using, for instance, the Extended Kalman Filter [45] or particle filtering [25]. The measurements provided by our algorithm could then become part of the measurement model for these filtering techniques. Such filters, though they usually introduce a delay, may be considered in a future implementation of our system.

Acknowledgments

This work was realized in the context of the French RIAM Sora national project in the Lagadic project at IRISA/INRIA Rennes. The authors would like to thank Cédric Riou from INRIA and Emmanuel Marin from Total Immersion (http://www.t-immersion.com), who have realized some of the reported experiments.

REFERENCES

[1] R. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments,* vol. 6, no. 4, pp. 355-385, Aug. 1997.

[2] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent Advances in Augmented Reality," *IEEE Computer Graphics and Application,* vol. 21, no. 6, pp. 34-47, Nov. 2001.

[3] M. Billinghurst, H. Kato, and I. Poupyrev, "The Magicbook: Moving Seamlessly between Reality and Virtuality," *IEEE Computer Graphics and Applications,* vol. 21, no. 3, pp. 6-8, May 2001.

[4] P. Bouthemy, "A Maximum Likelihood Framework for Determining Moving Edges," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 11, no. 5, pp. 499-511, May 1989.

[5] K.-W. Chia, A.-D. Cheok, and S. Prince, "Online 6 DOF Augmented Reality Registration from Natural Features," *Proc. IEEE Int'l Symp. Mixed and Augmented Reality,* pp. 305-316, Sept. 2002.

[6] A.I. Comport, D. Kragic, E. Marchand, and F. Chaumette, "Robust Real-Time Visual Tracking: Comparison, Theoretical Analysis and Performance Evaluation," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 2852-2857, Apr. 2005.

[7] A.J. Davison, "Real-Time Simultaneous Localisation and Mapping with a Single Camera," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1403-1410, 2003.

[8] S. de Ma, "Conics-Based Stereo, Motion Estimation and Pose Determination," *Int'l J. Computer Vision,* vol. 10, no. 1, pp. 7-25, 1993.

[9] D. Dementhon and L. Davis, "Model-Based Object Pose in 25 Lines of Code," *Int'l J. Computer Vision,* vol. 15, pp. 123-141, 1995.

[10] M. Dhome, J.-T. Lapresté, G. Rives, and M. Richetin, "Determination of the Attitude of Modelled Objects of Revolution in Monocular Perspective Vision," *Proc. European Conf. Computer Vision,* pp. 475-485, Apr. 1990.

[11] M. Dhome, M. Richetin, J.-T. Lapresté, and G. Rives, "Determination of the Attitude of 3D Objects from a Single Perspective View," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 11, no. 12, pp. 1265-1278, Dec. 1989.

[12] T. Drummond and R. Cipolla, "Real-Time Visual Tracking of Complex Structures," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 932-946, July 2002.

[13] B. Espiau, F. Chaumette, and P. Rives, "A New Approach to Visual Servoing in Robotics," *IEEE Trans. Robotics and Automation,* vol. 8, no. 3, pp. 313-326, June 1992.

[14] O.D. Faugeras and G. Toscani, "Camera Calibration for 3D Computer Vision," *Proc. Int'l Workshop Machine Vision and Machine Intelligence,* pp. 240-247, Feb. 1987.

[15] N. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography," *Comm. ACM,* vol. 24, no. 6, pp. 381-395, June 1981.

[16] S. Ganapathy, "Decomposition of Transformation Matrices for Robot Vision," *Pattern Recognition Letter,* vol 2, pp. 401-412, 1984.

[17] A. Glassner, "Everyday Computer Graphics, *IEEE Computer Graphics and Applications,* vol. 23, no. 6, pp. 76-82, Nov. 2003.

[18] R. Haralick, H. Joo, C. Lee, X. Zhuang, V Vaidya, and M. Kim, "Pose Estimation from Corresponding Point Data," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 19, no. 6, pp. 1426-1445, Nov. 1989.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge Univ. Press, 2001.

[20] *Visual Servoing: Real Time Control of Robot Manipulators Based on Visual Sensory Feedback,* World Scientific Series in Robotics and Automated Systems, vol. 7, K. Hashimoto, ed. Singapore: World Scientific Press, 1993.

[21] P.-W. Holland and R.-E. Welsch, "Robust Regression Using Iteratively Reweighted Least-Squares," *Comm. Statistics Theory and Methods,* vol. A6, pp. 813-827, 1977.

[22] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle, "An Analytic Solution for the Perspective Four-Points Problem," *Computer Vision, Graphics, and Image Processing,* vol. 47, no. 1, pp. 33-44, July 1989.

[23] P.-J. Huber, *Robust Statistics.* New York: Wiley, 1981.

[24] S. Hutchinson, G. Hager, and P. Corke, "A Tutorial on Visual Servo Control," *IEEE Trans. Robotics and Automation,* vol. 12, no. 5, pp. 651-670, Oct. 1996.

[25] M. Isard and A. Blake, "Condensation—Conditional Density Propagation for Visual Tracking," *Int. J. Computer Vision,* vol. 29, no. 1, pp. 5-28 Jan. 1998.

[26] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana, "Virtual Object Manipulation on a Table-Top AR Environment," *Proc. Int'l Symp. Augmented Reality 2000,* Oct. 2000.

[27] G. Klein and T. Drummond, "Robust Visual Tracking for Non-Instrumented Augmented Reality," *Proc. ACM/IEEE Int'l Symp. Mixed and Augmented Reality,* pp. 113-122, Oct. 2003.

[28] R. Koch, "Dynamic 3-D Scene Analysis through Synthesis Feedback Control," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 6, pp. 556-568, June 1993.

[29] R. Kumar and A.R. Hanson, "Robust Methods for Estimating Pose and a Sensitivity Analysis," *CVGIP: Image Understanding,* vol. 60, no. 3, pp. 313-342, Nov. 1994.

[30] K. Kutulakos and J. Vallino, "Calibration-Free Augmented Reality," *IEEE Trans. Visualization and Computer Graphics,* vol. 4, no. 1, pp. 1-20, Jan. 1998.

[31] V. Lepetit, J. Pilet, and P. Fua, "Point Matching as Classification Problem for Fast and Robust Object Pose Estimation," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* June 2004.

[32] Y. Liu, T.S. Huang, and O.D. Faugeras, "Determination of Camera Location from 2D to 3D Line and Point Correspondences," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 1, pp. 28-37, Jan. 1990.

[33] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[34] D.G. Lowe, "Three-Dimensional Object Recognition from Single Two-Dimensional Images," *Artificial Intelligence,* vol. 31, no. 3, pp. 355-394, Mar. 1987.

[35] D.G. Lowe, "Fitting Parameterized Three-Dimensional Models to Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 5, pp. 441-450, May 1991.

[36] C.P. Lu, G.D. Hager, and E. Mjolsness, "Fast and Globally Convergent Pose Estimation from Video Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 6, pp. 610-622, June 2000.

[37] Y. Ma, S. Soatto, J. Košecká, and S. Sastry, *An Invitation to 3-D Vision.* Springer, 2004.

[38] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau, "Robust Real-Time Visual Tracking Using a 2D-3D Model-Based Approach," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 262-268, Sept. 1999.

[39] E. Marchand and F. Chaumette, "Virtual Visual Servoing: A Framework for Real-Time Augmented Reality," *Eurographics Conf. Proc.,* pp. 289-298, Sept. 2002.

[40] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for Visual Servoing: A Generic Software Platform with a Wide Class of Robot Control Skills," *IEEE Robotics and Automation Magazine,* vol. 12, no. 4, pp. 40-52, Dec. 2005.

[41] N. Navab, "Developing Killer Apps for Industrial Augmented Reality," *IEEE Computer Graphics and Applications,* vol. 24, no. 3, pp. 16-20, May 2004.

[42] A.N. Netravali and J. Salz, "Algorithms for Estimation of Three-Dimensional Motion," *AT&T Technical J.,* vol. 64, no. 2, pp. 335-346, 1985.

[43] U. Neumann, S. You, Y. Cho, J. Lee, and J. Park, "Augmented Reality Tracking in Natural Environments," *Proc. Int'l Symp. Mixed Realities,* 1999.

[44] D. Nister, "Preemptive Ransac for Live Structure and Motion Estimation," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 199-207, Nov. 2003.

[45] J. Park, B. Jiang, and U. Neumann, "Vision-Based Pose Computation: Robust and Accurate Augmented Reality Tracking," *Proc. ACM/IEEE Int'l Workshop Augmented Reality,* pp. 3-12, Oct. 1998.

[46] S. Prince, K. Xu, and A. Cheok, "Augmented Reality Camera Tracking with Homographies," *IEEE Computer Graphics and Applications,* vol. 22, no. 6, pp. 39-45, Nov. 2002.

[47] R. Safaee-Rad, I. Tchoukanov, B. Benhabib, and K.C. Smith, "Three-Dimensional Location Estimation of Circular Features for Machine Vision," *IEEE Trans. Robotics and Automation,* vol. 8, no. 2, pp. 624-639, Oct. 1992.

[48] Y. Seo, M.H. Ahn, and K.-S. Hong, "Real-Time Camera Calibration for Virtual Studio," *IEEE Trans. Visualization and Computer Graphics,* vol. 6, no. 4, pp. 346-359, Oct. 2000.

[49] J. Shi and C. Tomasi, "Good Features to Track," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 593-600, June 1994.

[50] G. Simon and M.-O. Berger, "Pose Estimation for Planar Structures," *IEEE Computer Graphics and Applications,* vol. 22, no. 6, pp. 46-53, Nov. 2002.

[51] G. Simon, A. Fitzgibbon, and A. Zisserman, "Markerless Tracking Using Planar Structures in the Scene," *Proc. IEEE/ACM Int'l Symp. Augmented Reality,* pp. 120-128, Oct. 2000.

[52] C.-V. Stewart, "Robust Parameter Estimation in Computer Vision," *SIAM Rev.,* vol. 41, no. 3, pp. 513-537, Sept. 1999.

[53] V. Sundareswaran and R. Behringer, "Visual Servoing-Based Augmented Reality," *Proc. IEEE Int'l Workshop Augmented Reality,* Nov. 1998.

[54] H. Wang and D. Suter, "Robust Adaptive-Scale Parametric Estimation for Computer Vision," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 11, pp. 1459-1474, Nov. 2004.

[55] X. Zhang, S. Fronz, and N. Navab, "Visual Marker Detection and Decoding in AR Systems: A Comparative Study," *Proc. IEEE Int'l Symp. Mixed and Augmented Reality,* pp. 79-106, Sept. 2002.

Andrew I. Comport received the bachelor degrees in computer science in 1997 and in computer systems engineering with honors in 2000 from Monash University, Australia. He received the PhD degree in 2005 from the University of Rennes 1, France, at IRISA/INRIA Rennes in the Lagadic research group and is currently a postdoctoral fellow at INRIA Sophia Antipolis, France. He spent one year as a research assistant in the Intelligent Robotics Research Centre at Monash University in 2001. His research interests include real-time 3D tracking, robust statistics, nonrigid motion, visual servoing, computer vision, and the application of these techniques to robotics and augmented reality. He is a member of the IEEE.

Eric Marchand received the PhD degree and "habilitation diriger des recherches" in computer science from the University of Rennes in 1996 and 2004, respectively. He spent one year as a postdoctoral associate in the AI lab of the Department of Computer Science at Yale University. Since 1997, he has been an INRIA research scientist ("chargé de recherche") at IRISA-INRIA Rennes in the Lagadic project. His research interests include robotics, perception strategies, visual servoing and real-time object tracking. He is also interested in the software engineering aspects of robot programming. More recently, he studies new application fields for visual servoing such as augmented reality and computer animation. He is a member of the IEEE and the IEEE Computer Society.

Muriel Pressigout is currently a PhD candidate in computer science in the Lagadic project at IRISA in Rennes, France. She obtained an engineering degree from INSA Rennes in computer science and an MS degree from the University of Rennes 1 both in 2003. Her research interests include computer vision and real-time tracking for augmented reality and robotics.

François Chaumette graduated from École Nationale Supérieure de Mécanique, Nantes, France, in 1987. He received the PhD degree in computer science from the University of Rennes in 1990. Since 1990, he has been with IRISA/INRIA, Rennes, France, where he is now "directeur de recherches" and head of the Lagadic group. His research interests include robotics and computer vision, especially visual servoing and active perception. He received the AFCET/CNRS Prize for the best French thesis in automatic control in 1991. He also received with Ezio Malis the 2002 King-Sun Fu Memorial Best *IEEE Transactions on Robotics and Automation* Paper Award. He is a member of the IEEE and has been an associate editor of the *IEEE Transactions on Robotics* from 2001 to 2005. He is a member of the IEEE.