ELSEVIER

Image and Vision Computing 19 (2001) 941–955

image AND vision COMPUTING

www.elsevier.com/locate/imavis

A 2D–3D model-based approach to real-time visual tracking

Éric Marchand*, Patrick Bouthemy, François Chaumette

IRISA/INRIA Rennes, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France

Received 15 March 2000; revised 15 January 2001; accepted 21 January 2001

Abstract

We present a novel method for tracking, in a monocular image sequence, complex objects that can be approximately modeled by a polyhedral shape. The method consists of two stages of global transformation, the first one operating in 2D space and the second one in 3D space. The first stage is able to handle large displacements of the object projection in the image. It utilizes a 2D motion model estimated by a robust statistical method. Then, we refine the localization of the object silhouette by evaluating the 3D parameters related to the object pose by iteratively minimizing a nonlinear cost function. This last step aims at moving the projection of the contours of the object CAD model to the spatial intensity gradients in the image. The proposed tracking method is real-time, reliable and robust. Real tracking experiments and results embedded in a visual servoing positioning task are reported.

1. Introduction

We are interested in tracking complex objects in monocular image sequences. We consider objects that can be approximately modeled by a polyhedral shape. The aim is to use the tracking method within a robotics context. More precisely, we are concerned with the visual servoing [16] approach to control the movements of a robot using image information. This is of key interest, for example, in a hostile environment as a nuclear power plant. While most of the control issues are now well known and robust control laws can be defined to perform positioning or grasping tasks, the lack of really efficient image processing tools appears to be the main shortcoming of these techniques for a wider use. In particular, tracking an object with respect to the robotics tasks to be achieved in a not too restricted situation remains an open issue. Indeed, to fulfill visual servoing requirements, image feature extraction must be robust, sufficiently accurate, and computed in near real-time.

Techniques exploited in industrial environments usually involve the tracking of artificial landmarks (Fig. 1a), or specific well-defined geometrical features (Fig. 1b). In order to increase the versatility of visual servoing techniques, we need to develop methods to track complex objects in a non controlled environment (as the connector displayed in Fig. 1c).

Most of the available tracking techniques can be divided into two main classes: feature-based and model-based. The former approach considers features such as simple geometrical primitives (points, segments [4,15], circles [23], …), object contours [1,3], regions of interest [15], … The latter explicitly exploits a model of the tracked objects. This model can be a 2D template of the object (active contours [2,18] or statistical modal representation [6,19]) or a CAD model [7,9,12,20,22,27,29]. CAD-based tracking methods attempt to register a reference model with the projection in the image of the desired object. These approaches usually rely on a pose computation algorithm, involving the estimation of the 3D rigid transformation that links the object coordinate system to the camera coordinate system, from one view [22], multiple views or from an image sequence [7,9,12,20,29]. This class of methods usually obtains robust solutions (for example, it can cope with partial occlusion of the objects). Some approaches combine an estimation of the optical flow measurements [21] and a correlation between edge elements and model projection [20] to achieve robust tracking [14,29]. Since model registration approach may be sensitive to modeling errors, integrating optical flow information enhances tracking. Finally, more knowledge may be integrated into the tracking process. This is the case for the vehicle tracking approach where camera models, illumination models [14], various car models [14,28], vehicle motion model (ground plane constraint [28]) are considered. In parallel to our work, Drummond and Cipolla [9] have quite recently presented a CAD-based tracking system that provides a real-time pose estimation of a tracked object in visual servoing tasks. Points along the projection of the model edges are tracked in the direction normal to the

* Corresponding author. Tel.: +33-2-99-84-71-00; fax: +33-2-99-84-71-71.

E-mail address: eric.marchand@irisa.fr (E. Marchand).

0262-8856/01/$ - see front matter
PII: S0262-8856(01)00054-3

Fig. 1. Tracking features for visual servoing robotic tasks — three levels of complexity: (a) tracking artificial landmarks (white dots) for a grasping task; (b) tracking specific geometrical features (ellipse and segment); (c) tracking complex objects (a connector) in a textured environment.

edges. The 3D motion of the tracked object is then robustly computed from this information using an M-estimator. Another interesting aspect of this work is the ability to handle non-convex objects by considering a real-time hidden line removal rendering system (relying on computer graphics library) for the back-projection of the CAD model. The drawback of this approach is that it requires a very precise model of the tracked object. All these approaches may use Kalman filters to predict and smoothly estimate the position of the tracked primitives over time. Some of these approaches consider in addition full knowledge of the object motion model.

Our goal is to design a tracking method fulfilling the following properties or constraints: it should be fast and robust, it should require no prior learning step and it should not involve any complex feature extraction (such as contour extraction and linking). Therefore, we have developed an 'hybrid' 2D–3D model-based approach that relies both on the estimation of the 2D object displacement and the 3D pose of the object. This method obtains fast and robust tracking of complex objects that can be approximately modeled by a polyhedral shape. The method consists of two steps. In the first step, the object image transformation between two successive images is represented by a 2D affine model. This 2D model is estimated, using a robust statistical method, from the computation of the normal displacements along the projected object model contours between two successive images. These normal displacements are determined using the technique described in Ref. [5]. The 2D displacement model cannot always account for the real displacement of the object, a second step that consists of fitting the projection of the object model onto the spatial intensity gradients in the image is required. This is achieved using an iterative minimization of a non linear energy function with respect to the 3D pose parameters. The main advantages of this two-step method can be outlined as follows. The 2D displacement estimation stage allows us to handle large displacements of the object. It also avoids a prediction step which remains a questionable issue concerning the choice of the evolution model and the detection of model changes. The result of this first stage is used to supply an appropriate initialization to the pose estimator. Our model-based tracking only requires a coarse calibration of the camera and a rough model of the object. Neither 2D displacement estimation nor 3D pose estimation involve edge detection or image contour extraction. We only process gray level arrays. Both stages are robust to partial occlusions of the object. Finally, real-time tracking is reachable [24].

The paper is organized as follows. Section 2 describes the 2D motion-based tracking stage that acts as an initialization to the 3D model-based tracking presented in Section 3. Experimental tracking results and real-time visual servoing tasks are reported in Section 4. Section 5 contains concluding remarks.

2. 2D motion-based tracking

We first consider that the global transformation between two successive projections of the tracked object in the image plane can be represented by a 2D affine model. The goal of this first step is to estimate the parameters of this 2D transformation even in presence of large 2D displacements of the object image. Contrary to usual Kalman filtering methods, this motion-based method does not involve any prediction scheme. Therefore, it does not require the introduction of a state model evolution (e.g. a constant velocity model), and the initialization of the noise variance of the state and measurement models, which are often critical matters.

2.1. Affine and quadratic transformation models

Let $\mathbf{X}^t = [X_1^t, \ldots, X_n^t]^{\mathrm{T}}$ be a vector formed by the image coordinates X_i^t of points along the projection of the polyhedral object model boundaries at time t. This vector captures the shape and position of the image of the tracked object. The object image shape $\mathbf{X}^{t+1}$ at time $t+1$ will be given by

$$\mathbf{X}^{t+1} = \Psi_{\Theta}(\mathbf{X}^t), \tag{1}$$

where Ψ_{Θ} is a 2D affine transformation expressed as

$$\begin{bmatrix} x_i^{t+1} \\ y_i^{t+1} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x_i^t \\ y_i^t \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix} = \mathbf{W}(X_i^t)\boldsymbol{\Theta} \tag{2}$$

with $\Theta = (a_1, a_2, a_3, a_4, T_x, T_y)^{\mathrm{T}}, X_i^t = (x_i^t, y_i^t)^{\mathrm{T}}, X_i^{t+1} = \Psi_\Theta(X_i^t)$, and

$$\mathbf{W}(X) = \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix}.$$

This transformation is linear w.r.t. Θ. The displacement vector $\mathbf{d}_i(X_i) = X_i^{t+1} - X_i^t$ can be written as follows:

$$\mathbf{d}_i(X_i) = \mathbf{W}(X_i)\Theta', \tag{3}$$

where $\Theta' = \Theta - (1, 0, 0, 1, 0, 0)^{\mathrm{T}}$.

We have considered a six-parameter affine model. However, in the case of a planar rigid object, a eight-parameter quadratic model exactly accounts for the real 3D transformation. If required, we can then alternatively estimate the 2D polynomial model represented by $\Theta = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$ and given by

$$\begin{bmatrix} x_i^{t+1} \\ y_i^{t+1} \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} + \begin{bmatrix} a_2 & a_3 \\ a_4 & a_5 \end{bmatrix} \begin{bmatrix} x_i^t \\ y_i^t \end{bmatrix} + \begin{bmatrix} a_6 & a_7 & 0 \\ 0 & a_6 & a_7 \end{bmatrix} \begin{bmatrix} x_i^{t^2} \\ x_i^t y_i^t \\ y_i^{t^2} \end{bmatrix} = \mathbf{W}(X)\Theta \tag{4}$$

with

$$\mathbf{W}(X) = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{bmatrix}. \tag{5}$$

The part of the tracking algorithm concerned with the estimation of the 2D affine parameters is broken into two sub-steps:

- the first one computes normal displacements between two successive images along the projection of the object model contours using the so-called Moving Edges (ME) algorithm [5];
- the second one utilizes this normal displacement field to estimate $\hat{\Theta}'$ by adapting the robust multiresolution estimation technique introduced in Ref. [26].

We now describe these two sub-steps.

2.2. Computing normal displacements

One of the advantages of the ME method is that it does not require any prior edge extraction. We only manipulate point coordinates and image intensities. Nevertheless, we will still use the word 'contour' to refer to the list of tracked points for convenience. The ME algorithm can be implemented with convolution efficiency, thereby leading to real-time computation [4,5].

We consider a list L^t of pixels along the contour of the object model projection at time t. The model fitting step between the 3D object model and the projection of the object of interest in the very first image is performed in a semi-automatic fashion as described in Section 4.1. The normal displacement computation process consists of seeking the 'correspondent' P_i^{t+1} in the next image I^{t+1} of each point $P_i^t \in L^t$ in the direction normal to the contour. We determine a 1D search interval $\{Q_i^j, j \in [-J, J]\}$ in the direction δ of the normal to the contour (see Fig. 2). For each point P_i^t in the list L^t, and for every entire position Q_i^j (lying in fact for computational issue in the direction δ^*, closest to δ, from the set $\{0°, 45°, 90°, 135°\}$), we compute a criterion corresponding to the square root of a log-likelihood

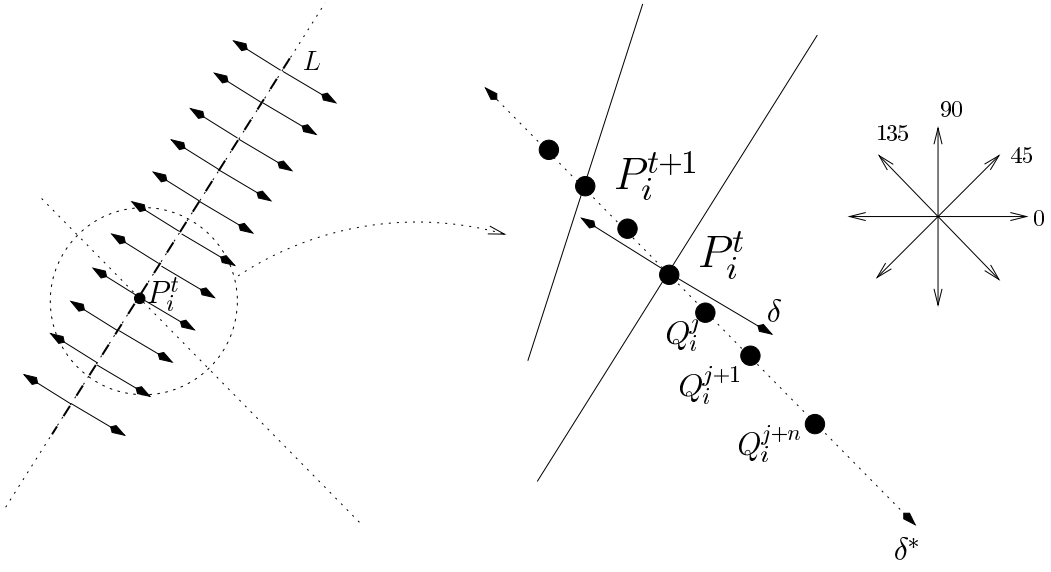

Fig. 2. Determining point positions of the tracked object contours in the next image using the ME algorithm.

ratio ζ^j. The latter is nothing but the absolute sum of the convolution values, computed at P_i^t and Q_i^j, respectively, in images I^t and I^{t+1}, using a pre-determined mask M_δ function of the orientation of the contour [5].

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

$$Q_i^{j^*} = \arg \max_{j \in [-J,J]} \zeta^j$$

with

$$\zeta^j = |I^t_{\nu(P_i)} * M_\delta + I^{t+1}_{\nu(Q_i^j)} * M_\delta|$$

provided that ζ^{j*} is greater than a given threshold λ. $\nu(\cdot)$ is the neighborhood of the considered pixel. Then, pixel P_i^{t+1} given by $Q_i^{j^*}$ is stored in L^{t+1}.

At this step, we have a list of k pixels as well as their displacement components orthogonal to the object model contour: $(P_i^t, d_i^\perp)_{i=1\ldots k}$ (see Fig. 3). This is computed for each new frame, it never requires the extraction of new contours. Since it is a local approach, the presence of partial occlusions of the object leads to losing some local measurements only, without perturbing the available ones.

Other tracking approaches also consider searching orthogonally to an hypothesized edge segment or to a tracked edge. This is the case, for example, in work by the Reading group (e.g. [28]), Nagel's group (e.g. [14]) or Drummond and Cipolla [9].

2.3. 2D affine transformation estimation

From $(P_i^t, d_i^\perp)_{i=1,\ldots,k}$, we can estimate the 2D affine transformation $\boldsymbol{\Theta}'$. Using Eq. (3), we have

$$d_i^\perp = \mathbf{n}_i^{\mathrm{T}} d(P_i) = \mathbf{n}_i^{\mathrm{T}} \mathbf{W}(P_i) \boldsymbol{\Theta}', \tag{6}$$

where $\boldsymbol{n}_i$ is the unit vector orthogonal to the object model contour at point P_i^t. Relying on Eq. (6), we can use a robust estimator (an M-estimator ρ) to obtain $\hat{\boldsymbol{\Theta}}'$ as follows [25]:

$$\hat{\boldsymbol{\Theta}}' = \arg \min_{\boldsymbol{\Theta}'} \sum_{i=1}^{n} \rho(d_i^\perp - \mathbf{n}_i^{\mathrm{T}} \mathbf{W}(P_i^t) \boldsymbol{\Theta}'). \tag{7}$$

This robust statistical criterion is not to be affected by locally incorrect or missing measurements due to shadows, local miss-matching, partial occlusions, etc.

3. 3D model-based tracking

3.1. Overview

Knowing the position $\mathbf{X}^t$ of the projection of the tracked object contours at time t and the estimated parameters $\hat{\boldsymbol{\Theta}}'$ of the 2D global affine displacement model between t and $t+1$, we are able to compute the positions of points $\mathbf{X}^{t+1}$ at time $t+1$ according to:

$$\mathbf{X}^{t+1} = \Psi_{\hat{\Theta}}(\mathbf{X}^t)$$

with

$$\hat{\boldsymbol{\Theta}} = \hat{\boldsymbol{\Theta}}' + (1,0,0,1,0,0)^{\mathrm{T}}.$$

However, the 2D affine transformation cannot generally completely account for the real transformation undergone by the projection of the object (e.g. due to perspective effects, important rotations, non shallow environment), and may fail after tracking in a few images. To alleviate this problem, different approaches can be proposed:

- In the case of a planar object, the quadratic 2D transformation fully described the 2D object displacement. To avoid the integration of errors over time, the previous 2D tracking process could be completed with a fine registration of the 2D template on the intensity gradients of the images w.r.t. 2D quadratic displacement parameters. In the case of non-planar objects, however, the introduction of a quadratic model does not result in much improvement.
- In a first version of the algorithm, the 2D affine displacement model was augmented with 2D local deformations [13,26]. However, when adding local deformations, we cannot ensure global 3D rigidity constraints. Moreover, this was highly time consuming.
- Finally, we can exploit a rough CAD polyhedral model of the object. This is the approach we present now.

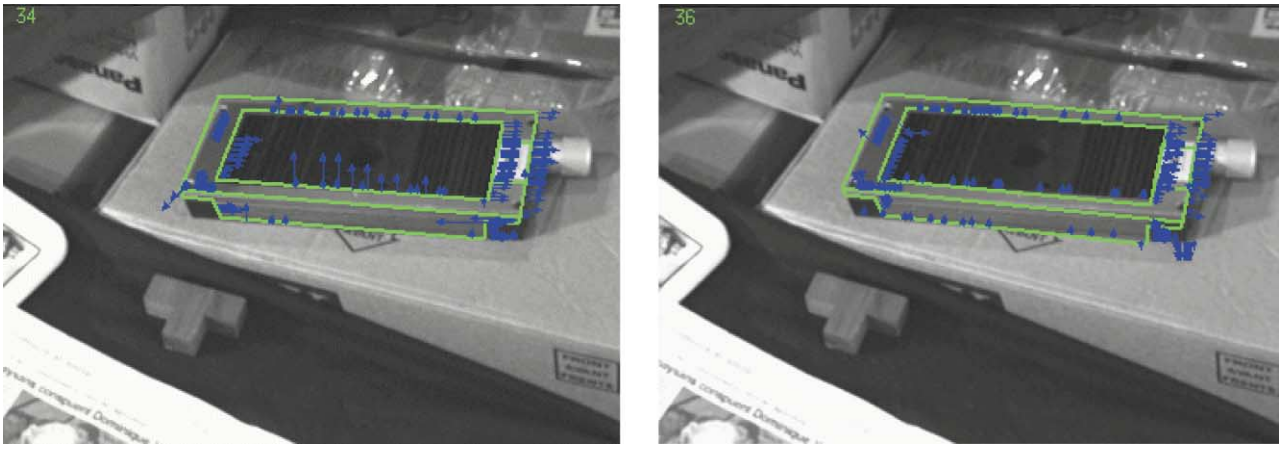

Fig. 3. Computed normal displacement vectors on two successive frames ($J = \pm 5$, $\lambda = 1500$).

3.2. Fitting CAD model on the spatial intensity gradients

Our goal now is to fit a CAD model of the tracked object onto the intensity spatial gradients in the successive images. To achieve this goal, we consider a pose computation algorithm, where we find the 3D rotation and the 3D translation transformations (i.e. the pose denoted by Φ) that map the object coordinate system onto the camera coordinate system.

3.2.1. Initial pose computation

We can use the output of the 2D tracking stage as an appropriate initialization for pose computation. Next, we estimate the pose of the object w.r.t. the camera from the positions $\mathbf{X}^{t+1}$ obtained after the first 2D tracking stage described in Section 2. A number of methods to compute pose from points have been proposed. We have used the method designed by Dementhon and Davis [8] completed by Lowe's non-linear method [22]. Using Dementhon's method, we calculate the rigid transformation iteratively using the coordinates of at least four points in the object coordinate system, and of their corresponding projections in the image as computed by the estimated 2D transformation: $\Psi_{\hat{\Theta}}(\mathbf{X}^t)$. Dementhon's method principle consists in approximating perspective projection by scaled orthographic projection, and then iteratively modifying the scaled orthographic projection to converge to the perspective projection. We then apply Lowe's method to improve the pose estimation: Lowe's approach is based on an iterative minimization of a residual using the non linear Levenberg-Marquardt minimization technique. We thus compute a first estimate of the pose parameters Φ_{init}^{t+1} :

$$\Phi_{\text{init}}^{t+1} = f(\mathbf{X}^{t+1}) = f(\Psi_{\hat{\Theta}}(\mathbf{X}^t)),$$

where $f(\cdot)$ denote the pose computation process. Once the pose parameters are available, we can easily determine visible and invisible faces of the object, which are of particular interest for the fitting stage described in the next paragraph.

However, since this initial pose Φ_{init}^{t+1} is directly related to the 2D tracking process, it should be updated to correspond to the real new aspect of the object as closely as possible.

3.2.2. Fitting CAD model

The next step consists of fitting the projection of the object model onto the spatial intensity gradients in the image. This is achieved using an iterative minimization w.r.t. the pose parameters Φ of a non-linear cost function using Φ_{init}^{t+1} as the initial estimate. Final pose parameters Φ are then given by:

$$\hat{\Phi} = \arg \min_{\Phi} E(\Phi), \tag{8}$$

where the cost function $E(\Phi)$ is defined as

$$E(\Phi) = - \int_{\Gamma_\Phi} \| \nabla I_{\pi_\Phi(s)} \| \, \mathrm{d}s, \tag{9}$$

where Γ_Φ represents the visible part of the 3D object model contours for the pose Φ, and $\nabla I_{\pi_\Phi(s)}$ denotes the spatial gradient of the intensity function at image point $\pi_\Phi(s)$ along $\pi_\Phi(\Gamma_\Phi)$, where π_Φ is the perspective projection function.

The cost function defined in Eq. (9) is the simplest one to

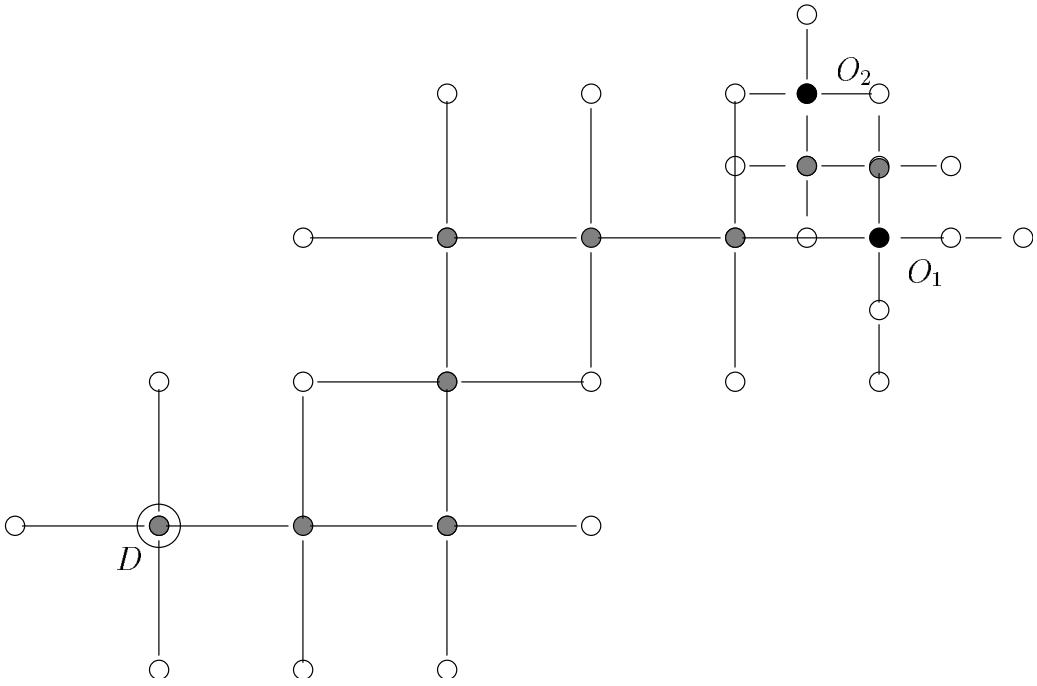

Fig. 4. Minimization algorithm illustrated for two pose parameters relying on a discrete hierarchical search procedure. In fact, we have six parameters to estimate.

express the fitting problem. However, we may exploit other available information than the norm of the spatial image gradient. When projecting the object model for a given pose Φ, we are able to compute the expected direction of the projected model contour at a 2D point $\mathfrak{s} = \pi_\Phi(s)$. If we denote $\mathbf{n}_\Phi(s)$ as the unit vector corresponding to this expected direction, the dot product $\nabla I_{\mathfrak{s}} \cdot \mathbf{n}_\Phi(s)$ should be ideally equal to zero, and in practice close to zero. Another expression of the cost function exploiting this information can then be considered as follows (see also [14]:

$$E(\Phi) = \int_{\Gamma_\Phi} \frac{|\nabla I_{\mathfrak{s}} \cdot \mathbf{n}_\Phi(s)|}{\| \nabla I_{\mathfrak{s}} \|^2} \, \mathrm{d}s, \tag{10}$$

where we only consider points where $\| \nabla I_{\mathfrak{s}} \| > \epsilon$. This cost expression enhances results in case of textured environments.

The projection function π_Φ depends on the camera intrinsic parameters I. The minimization of the cost function (9) requires that the camera calibration is available. Nevertheless, a rough knowledge of the camera parameters is sufficient. If the calibration is wrong, the resulting estimation of Φ will be obviously biased, but the projection of the CAD model onto the image, which is the useful information for image tracking purposes, is still correct. On the other hand, these intrinsic camera parameters could also be estimated (or at least updated) on-line. In that case, the criterion to be minimized can be rewritten as follows:

$$(\hat{\Phi}, \hat{I}) = \arg \min_{(\Phi, I)} \{E(\Phi, I)\}. \tag{11}$$

In the general case, we have 11 parameters to estimate (if we consider the radial distortion). In practice, we have only performed experiments dealing with the on-line estimation of the radial distortion.

3.3. Computational issues

To ensure a fast tracking, some computational issues must be considered especially dealing with the discretization of Eqs. (9) and (10) and with the optimization of criterion (8).

3.3.1. Discretization

Discretization of Γ_Φ can be considered in different ways. If we consider N_e visible model contours to be discretized into N_p 2D points, expression (9) can be rewritten as:

$$E(\Phi) = -\frac{1}{N_e} \sum_{e=1}^{N_e} \left(\frac{1}{N_p^e} \sum_{p=1}^{N_p^e} \| \nabla I_{\mathfrak{s}_p^e} \| \right). \tag{12}$$

Next, we determine the $N_e \times N_p^e$ 2D points $\mathfrak{s}$. The first approach is to discretize each contour into N_p^e points s in the 3D space, and then, to project these points onto the image plane ($\mathfrak{s} = \pi_\Phi(s)$) $\cdot \mathfrak{s}_p^e$ is thus computed as:

$$\mathfrak{s}_p^e = \pi_\Phi \left(s_0^e + \frac{p-1}{N_p^e} (s_1^e - s_0^e) \right) \qquad p = 1, \ldots, N_p^e, \tag{13}$$

where s_0^e and s_1^e are the two 3D extremities of the model contour e. Alternatively, a second approach can be considered. The discretization can be performed after the projection of the extremities of the 3D visible object contour in the image. Since the considered objects are polyhedral, the projection of their contours are also segments and $\mathfrak{s}_p^e$ can be computed as

$$\mathfrak{s}_p^e = \pi_\Phi(s_0^e) + \frac{p-1}{N_p^e} (\pi_\Phi(s_1^e) - \pi_\Phi(s_0^e)) \qquad p = 1 \ldots N_p^e,$$

$$\mathfrak{s}_p^e = \mathfrak{s}_0^e + \frac{p-1}{N_p^e} (\mathfrak{s}_1^e - \mathfrak{s}_0^e).$$

These discretization schemes do not include the distortion term, but knowing K_d, the correct position of the image points can be easily computed. These two versions are similar in term of complexity when distortion is important, but the second one avoids a discretization step in 3D. When there is no need to consider radial distortion, the latter approach is indeed more efficient, since we compute only two projections in the image per segment while the former implies N_p^e projections. Furthermore, there exist efficient algorithms to determine all the pixels (entire positions) attached to a given real segment (e.g. using Bresenham algorithm [11]).

3.3.2. Optimization algorithm

Another important issue is the optimization procedure. Expressions (9) and (10) are non linear, and involve numerous local minima. To solve this issue we resort to an explicit discrete search algorithm. The generalized Hough transform, consisting of building a cumulative histogram in the pose space, presents two main shortcomings: the size of the pose space ($\mathbb{R}^6$) and the presence of false peaks. Instead we have considered a recursive search algorithm inspired from a classical algorithm for fast block matching [17]. First, $E(\phi)$ is minimized considering large variation steps of parameters values. When the current minimum is found, the process is iterated with smaller variation steps around this value (see Fig. 4, D is the initial estimate of the camera location, O_1 corresponds to the first minimum found with a first variation step Δ_1, and O_2 is an improved minimum computed from O_1 with a second variation step $\Delta_2 < \Delta_1$). In practice, the initial solution Φ_{init}^{t+1} is a proper initialization of this search algorithm. Therefore, we can bound the search space, allowing the algorithm to converge very quickly toward an appropriate minimum.

4. Experimental results

Experiments reported hereafter involve various objects. The object complexity is representative of the applications in which EDF (Électricité de France) is interested: disassembly and monitoring tasks in the nuclear power plant context. These systems are of interest for the Research and Development division of EDF to achieve maintenance and monitoring tasks in hostile environment. We have selected to track a nut, a micro-controller testbed, and a

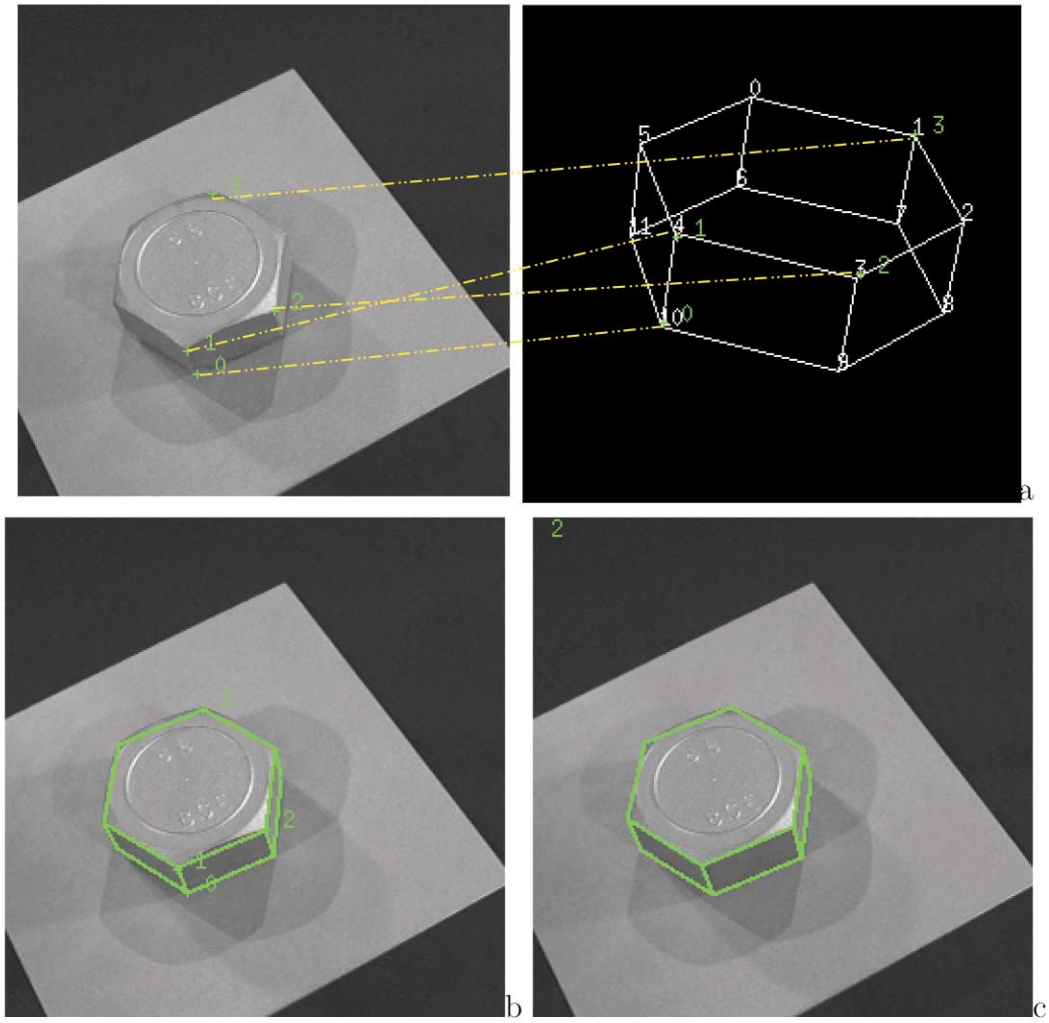

Fig. 5. Tracking initialization: (a) selected points in the first image and the CAD model of the object (the registration is done manually by 'clicking' a point in the image and the corresponding point on the CAD model. The virtual dotted lines account for this manual 2D–3D registration), (b) back projection of the 3D model using the pose computation process by Dementhon-Davis algorithm, (c) back projection of the 3D model after pose computation using our CAD fitting stage.

serial connector. For all the experiments reported in this section, camera calibration is not precisely known. All the images were acquired using the IRISA's robotic testbed displayed in Fig. 11.

4.1. Initialization of the tracking in the very first image

Currently, manual intervention is required to initialize the tracking algorithm for the very first image: the clicks of at least four points on both the initial image and the CAD model of the object (see Fig. 5). This is performed within an interactive procedure ensuring the matching between the appropriate model points (vertices of the CAD model) and their corresponding projections in the images selected by the user. A completely automatic object localization procedure could be implemented, but this is outside the tracking issue considered in this paper. Let us note that, if the user clicks a minimum of six points, a full camera calibration can be performed (the calibration will be actually rough with this small number of points). The obtained intrinsic parameters can be used afterwards in the pose computation algorithm. After this interactive early step, a first pose is computed for the same first image using the Dementhon algorithm (Fig. 5b), and this pose is then refined using our CAD fitting stage (Fig. 5c) to account for imprecision in the manual initialization.

4.2. Tracking experiments

4.2.1. Nut tracking

We first consider the tracking of the nut silhouette in image sequences acquired along predefined trajectories of the camera. Let us point out that we deal with low intensity contrast (as it can be seen in the intensity gradient image of

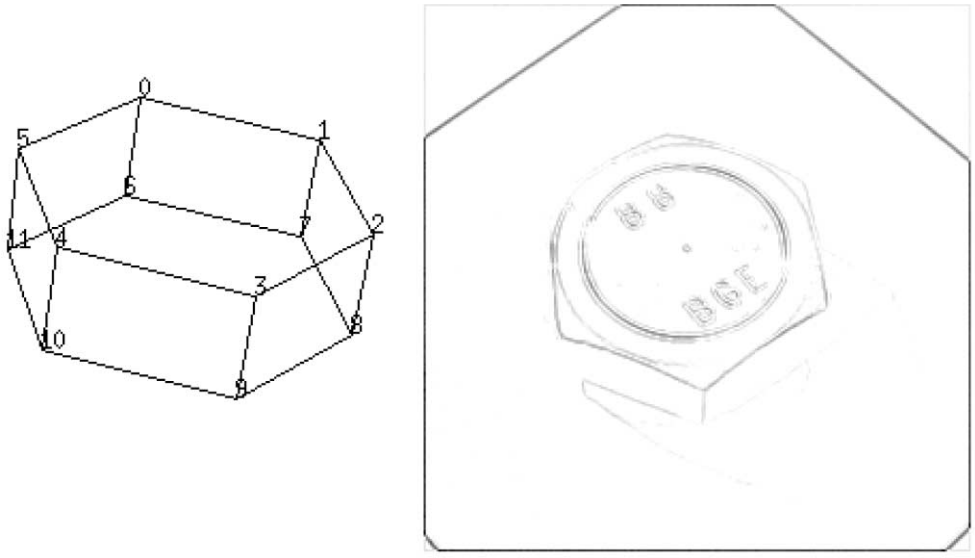

Fig. 6. Object of interest — the nut: (a) considered approximate CAD model of the nut; (b) magnitude of spatial intensity gradients in a typical image of the sequence.

Fig. 6b), the presence of cast shadows, of specularities,… Moreover, the nut is not exactly polyhedral, since it presents no physically angular ridges. Finally, the CAD model was hand-made, implying low precision. Despite these difficulties, the proposed method has proven the ability to efficiently track this object along long image sequences.

Fig. 7 shows the results of the tracking of the nut along a sequence of 44 images. Fig. 7a shows the results of the tracking if we consider only the 2D displacement estimation step. In that case, tracking is performed at video rate (25 Hz). However, after a few images, the algorithm is no longer able to track accurately the object shape because a 2D affine model cannot completely account for the

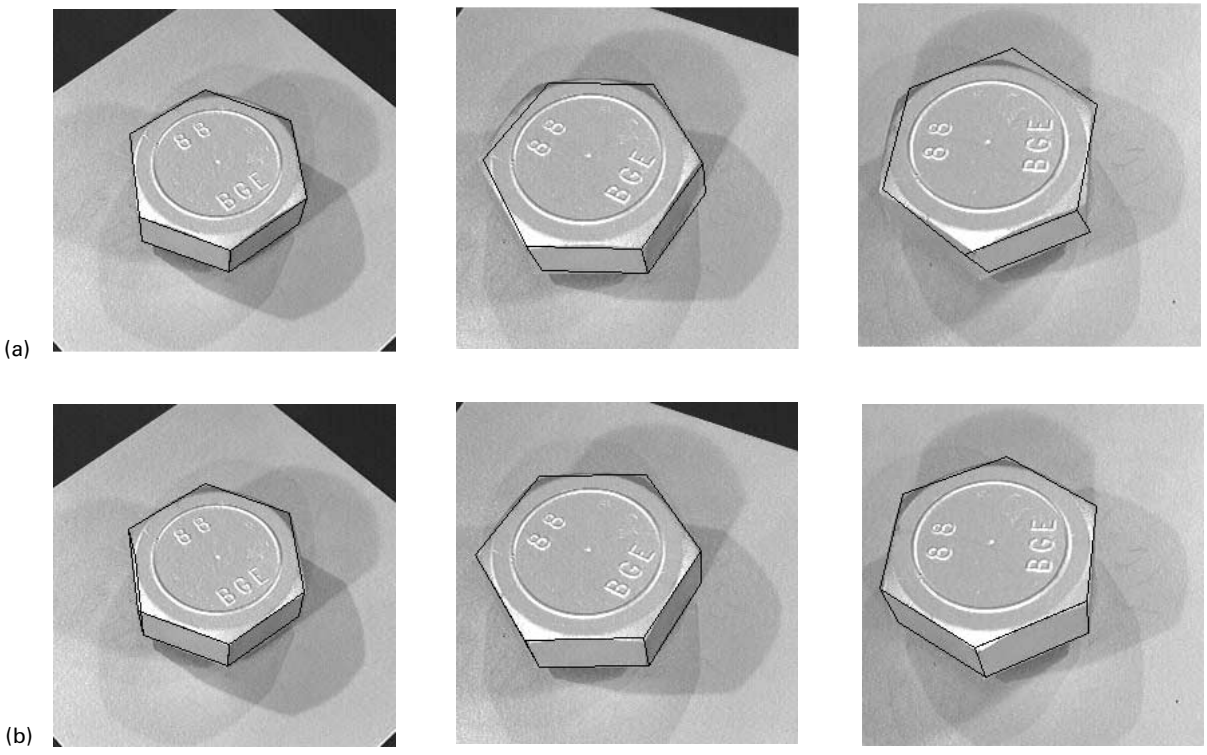

Fig. 7. Nut tracking (images 256×256, $J = \pm 5$, $\lambda = 1500$): (a) tracking with only the 2D stage; (b) tracking with both 2D displacement estimation and 3D pose computation stages.

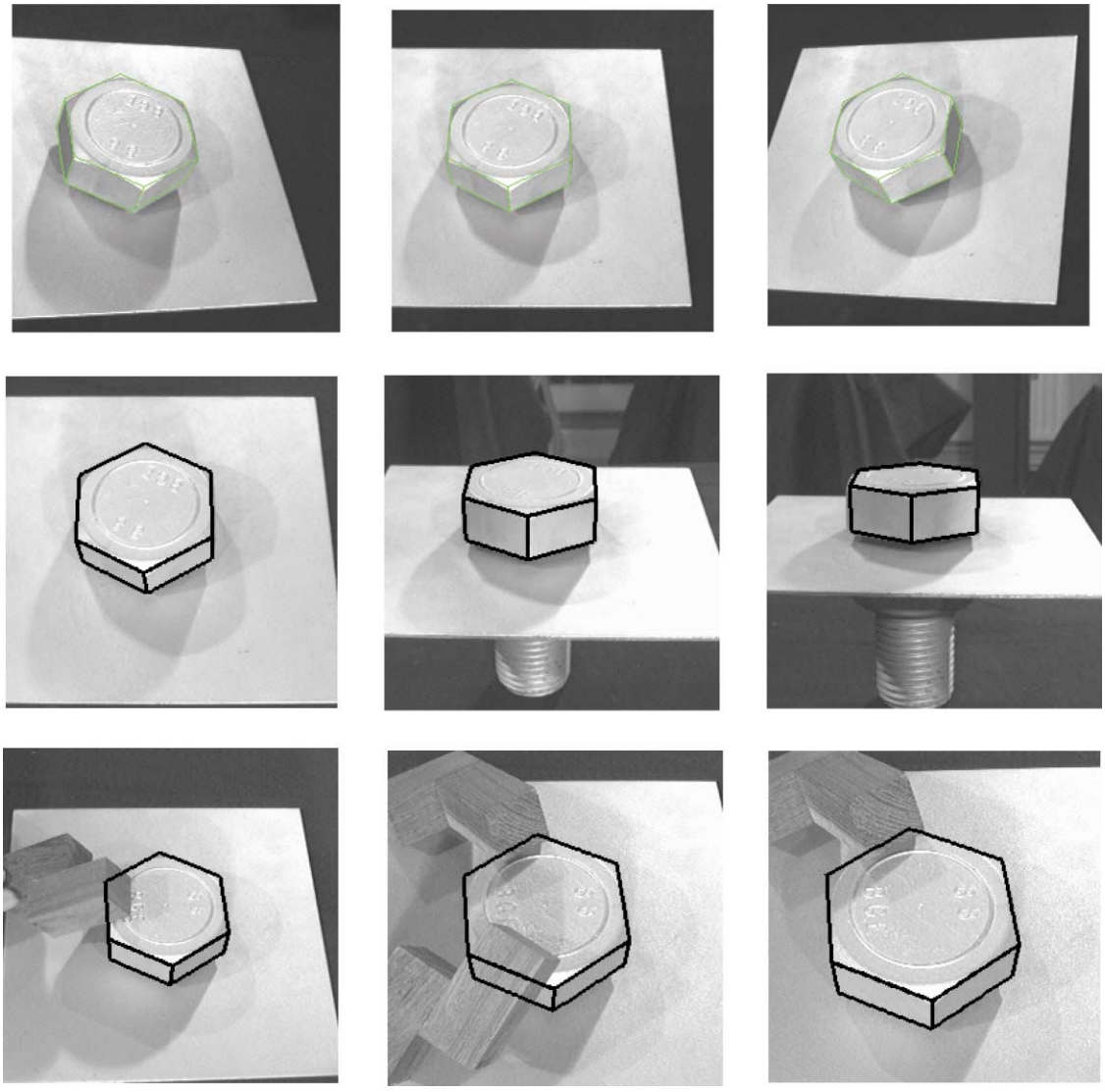

Fig. 8. Successful nut tracking experiments featuring various difficulties (images 256×256, see text for details $J = \pm 5$, $\lambda = 1500$).

projection of the 3D motion of the object. Fig. 7b shows the results of the tracking using both stages, 2D displacement estimation and 3D pose computation. In that case, tracking can be performed at 10 Hz on a PC 400 Mhz under Linux OS.

We have also validated the performance of the tracking algorithm in the presence of various difficulties. In the experiment shown in Fig. 8a, camera motion is performed around the y-axis[1] resulting in one face of the nut appearing as another disappears. In Fig. 8b, the main difficulty is the very important rotation around the x-axis. Furthermore, the illumination conditions are not constant along the sequence. In Fig. 8c, partial occlusions of the nut occur. In Fig. 14d, the nut is tracked within a highly textured environment during a visual servoing experiment (see Section 4.3. In all these experiments, satisfactory results are obtained.

4.2.2. Tracking a serial connector

We have evaluated our tracking method on a more complex object. In the experiments reported in Fig. 9, we consider a serial port connector placed on a newspaper forming a 'cluttered' background. Here, we have also dealt with low intensity gradient images, specularities, and no precisely defined contours. The serial connector is successfully tracked over an image sequence of 170 frames. As the camera performed a large displacement around the object, the face of the object is appearing as another disappears. Tracking is performed at 3 Hz (this lower processing rate is mainly due to the size of the CAD model of the object comprising more contours and leading to consider a greater number of points ς in the minimization of the cost function).

[1] z-Axis follows the optical axis, while x-axis is parallel to the image rows and y-axis is parallel to image columns.

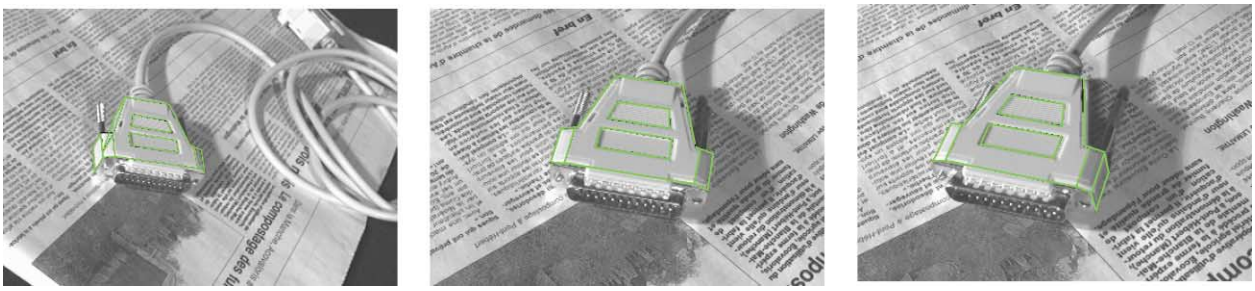

Fig. 9. Tracking a connector on a newspaper background (images 365×256, $J = \pm 2$, $\lambda = 1500$).

4.2.3. On-line estimation of radial distortion

We have also tried to estimate on-line the radial lens distortion. We have considered a simple object (a box) and a camera with an important, but unknown, radial distortion (see Fig. 10). The use of this simple object enables to easily illustrate the algorithm capability to correctly handle distortion estimation. The focal lens of the camera is 3.5 mm. We estimate on-line the distortion with an initial value set to 0. The estimation of the radial distortion improves when the object projection moves toward the image border, since a better estimation of this parameter is then required. In that case, the tracking is performed at 1 Hz.

4.3. Tracking within visual servoing experiments

4.3.1. Visual servoing overview

Image-based visual servoing consists of specifying a task as the regulation in the image of a set of visual features P that have to match a desired value P_d [10,16]. The control law supplying an exponential decrease of the error $P - P_d$ in the image is given by [10]:

$$T_c = -\lambda \mathbf{J}^+ (\mathrm{P} - \mathrm{P_d}), \tag{14}$$

where T_c is the camera velocity, λ a scalar, and $\mathbf{J}^+$ denotes the pseudo-inverse of the interaction matrix or image Jacobian $\mathbf{J}$ that links the camera motion to the image object motion.

We have considered positioning tasks. From an initial position of the robot, we want to reach a desired position expressed as a desired position of the object in the image. The complete implementation of the visual servoing task, including tracking and control, was carried out on an experimental testbed involving a CCD camera mounted on the end effector of a six degrees of freedom cartesian robot (see Fig. 11). The realization of such an experiment involves the following steps:

- the camera is first positioned at the final desired position in order to learn the targeted image features P_d;
- the camera is then placed at the initial position. After a

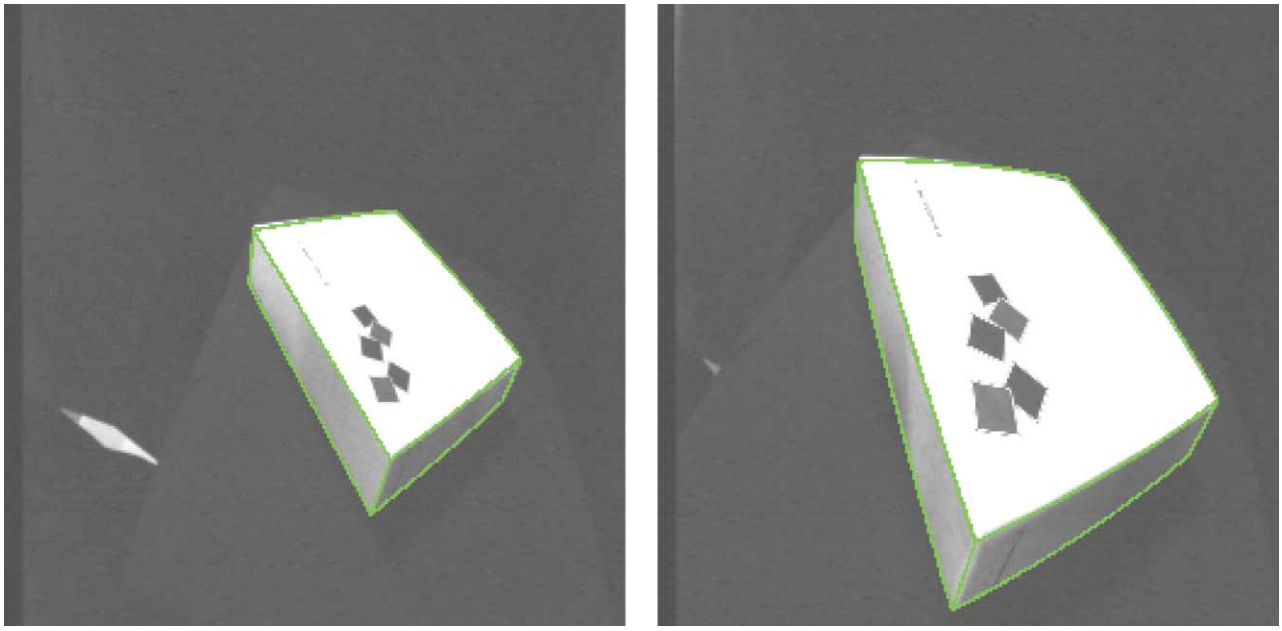

Fig. 10. Handling distortion: distortion is very important in this example due to the use of a 3.5 mm lens (images 512×512, $J = \pm 10$, $\lambda = 1500$).

Fig. 11. Experimental setup at IRISA/INRIA Rennes: six degrees of freedom cartesian robot with a camera mounted on the end-effector.

semi-automatic initialization as described in Section 4.1, current values of selected visual features P are computed from the results of the tracking algorithm at each iteration of the control low by backprojection of the CAD model. Camera motion is computed according to Eq. (14). The synoptic of the full process is given in Fig. 12.

Let us note that, even if we recover the object pose, i.e. the 3D position of the object w.r.t. the camera, we do not use this information within the visual servoing loop. Hence, we can cope with a rough camera calibration; the pose may be instable, or even biased, but this does not matter as long as the projection of the object model in the image is correct w.r.t. to the specification and the realization of the task at hand. 3D visual servoing [30] could also be considered but, in that case, instability and errors in pose computation may become a major problem.

4.3.2. Positioning with respect to a nut

Fig. 13a displays some of the images delivered by the camera during the positioning task. The current polygonal object model contours (in blue), depicting the results of the tracking of the nut projection in the image, and the desired final pattern (in red) are drawn over the images. In this experiment, the image features P are the six corners of the upper face of the nut. Fig. 13c shows the apparent trajectory in the image plane of points P during the achievement of the task. Fig. 13c presents the temporal evolution of the error $P - P_d$. These figures show the stability and the convergence of the control law. The error on each coordinate of

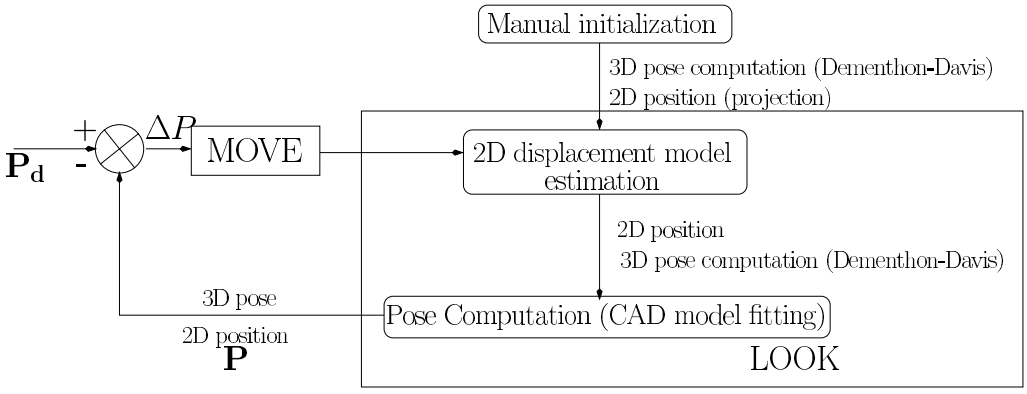

Fig. 12. Overview of the visual servoing task.

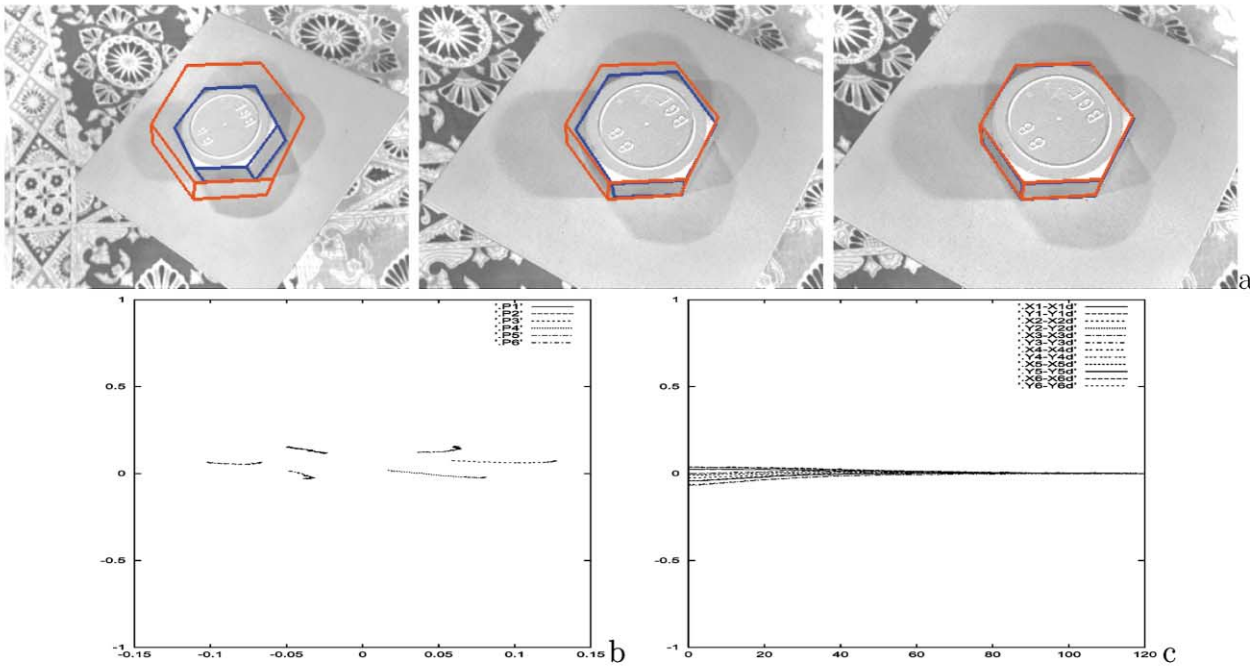

Fig. 13. Positioning on the nut by visual servoing: (a) images supplied by the camera mounted on the end-effector of the robot during the positioning task at time 15, 50 and 95. In red projection of the CAD model corresponding to the desired position while the projection of tracked object appears in blue (image 365×256, $J = \pm 5$, $\lambda = 1500$); (b) temporal variation of the positions P of the control points in the image; (c) plots of the errors between the current and desired positions of the control points considered in the specification of the task.

the six points specifying the task rapidly tends to zero. Noise appearing in the plots is mainly due to the fact that image processing is performed only at 3 Hz.

4.3.3. Robustness

To prove the robustness of our algorithm, we put the nut on a highly textured environment as shown in Fig. 14. As in the previous experiments, our tracking algorithm embedded in the visual servoing scheme has correctly achieved the positioning task w.r.t. the nut. Other experiments were carried out using a micro-manipulation device as object of interest (see Fig. 15). Multiple temporary and partial occlusions by various tools were imposed during the realization of the positioning task.

4.3.4. Accuracy

Grasping is one application utilizing results from a positioning task. In this context, repeatability is very important. The final position of the object in the image must be accurate enough, and the final 3D position of the robot end-effector must also be consistent in order to achieve the grasping task. The accuracy obtained in the positioning task w.r.t. the nut was computed from 40 experiments using the very precise robot odometry. We obtains an accuracy (expressed as the standard deviation of the final 3D positions of the end-effector) within ± 0.7 mm in translation and $\pm 0.17°$ in rotation, when the object is located 40 cm from the camera. The mean error $\| P - P_d \|$ is less than 0.2 pixels with a standard deviation of 0.3 pixels.

5. Conclusion

We have presented a novel method for tracking complex objects in an image sequence at a high processing rate (but

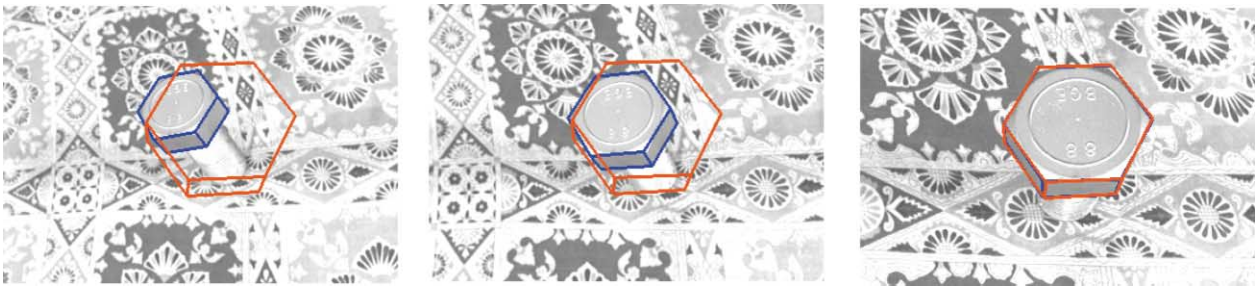

Fig. 14. Positioning w.r.t. the nut by visual servoing within a highly textured environment at time 1, 40 and 154 (image 365×256, $J = \pm 3$, $\lambda = 1500$). In red projection of the CAD model corresponding to the desired position while the projection of tracked object appears in blue.

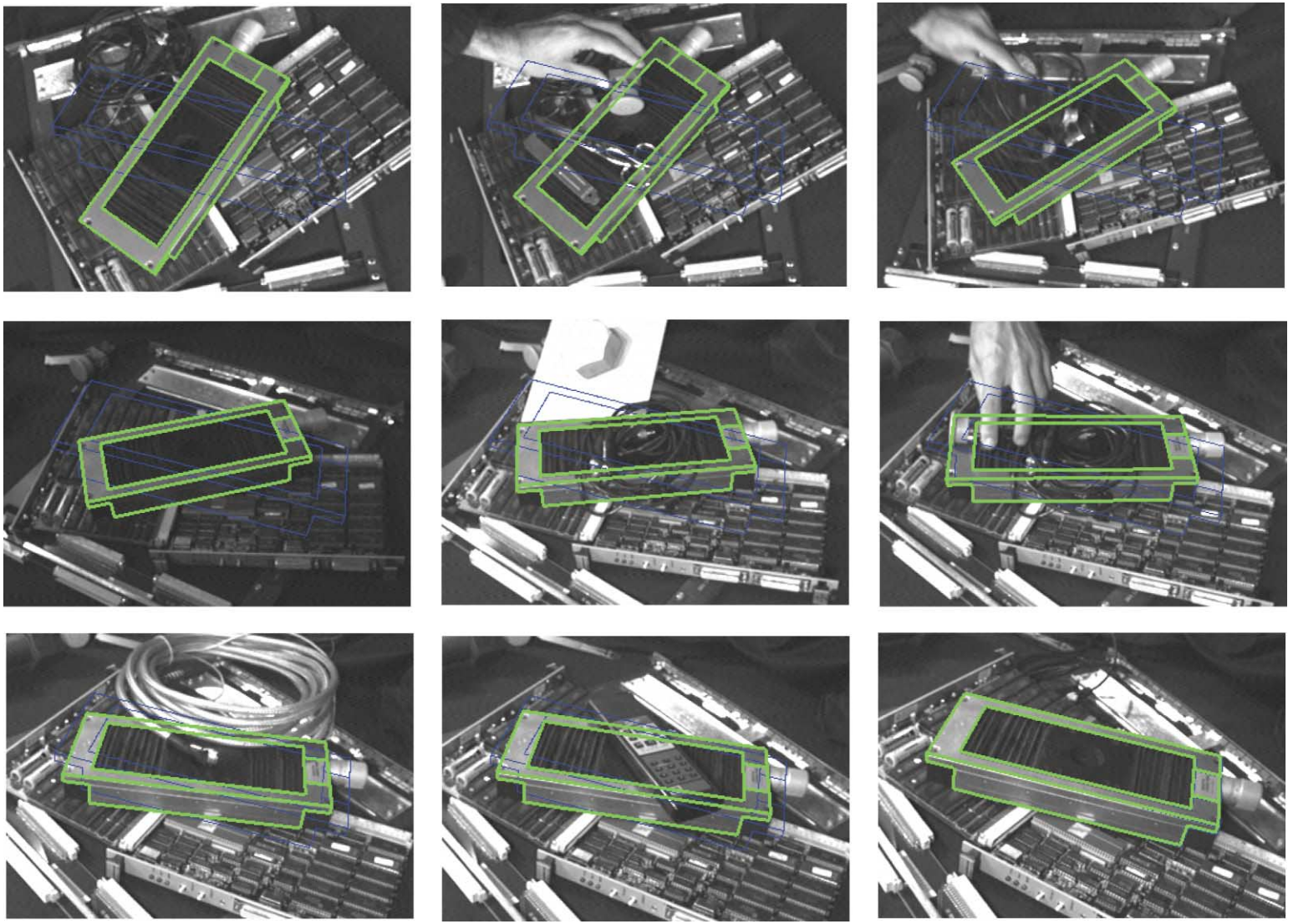

Fig. 15. Positioning w.r.t. a micro-manipulation device, the target pattern appears in blue (image 365×256, $J = \pm 5$, $\lambda = 1500$). In blue projection of the CAD model corresponding to the desired position while the projection of tracked object appears in green.

not yet exactly at video rate). The object tracking method is a two-step process based on the robust estimation between two successive images of a 2D global affine transformation undergone by the object projection, and on the computation of the object pose formulated as a cost minimization process. To perform this last step, an approximate polyhedral model of the object is sufficient. Appearance and disappearance of hidden faces of the object can be handled in a straightforward manner. Both steps of the tracking algorithm are robust to partial occlusions. The direct extension to non polyhedral object can be considered provided a 3D description of the object is available, since this approach only requires the contours of the 3D object to be projected onto the image. This tracking algorithm allows us to efficiently realize visual servoing tasks in the robotics domain. We experimentally demonstrate this through various positioning tasks with respect to different real objects (without any landmarks) in complex situations. Visual servoing is not the only application of this tracking method. Indeed, if we are able to achieve such a 2D tracking, we can also recover an appropriate precise estimation of the position of the camera w.r.t. the object if the camera is well calibrated, and then we can also perform a real 3D tracking.

6. Additional information

Full result sequences can be found at http://www.irisa.fr/vista; follows the *demonstrations* link then the *Robust real-time tracking* link.

Acknowledgements

This study was supported by EDF — Pôle Industrie — Division Recherche et Développement under contract 1.97.C234. The image sequence of Fig. 10 has been supplied by Lasmea, University of Clermont-Ferrand, France. The authors wish to thank V. Sundareswaran for his careful reading of the paper.

References

[1] B. Bascle, P. Bouthemy, N. Deriche, F. Meyer, Tracking complex primitives in an image sequence, in: Proceedings of the International Conference on Pattern Recognition, ICPR'94, Jerusalem, October 1994, pp. 426–431.

[2] M.-O. Berger, How to track efficiently piecewise curved contours with a view to reconstructing 3D objects, in: Proceedings of the International Conference on Pattern Recognition, ICPR'94, Jerusalem, October 1994, pp. 32–36.

[3] A. Blake, M. Isard, Active Contours, Springer, Berlin, 1998.

[4] S. Boukir, P. Bouthemy, F. Chaumette, D. Juvin, A local method for contour matching and its parallel implementation, Mach. Vision Appl. 10 (5/6) (1998) 321–330.

[5] P. Bouthemy, A maximum likelihood framework for determining moving edges, IEEE Trans. Pattern Anal. Mach. Intell. 11 (5) (1989) 499–511.

[6] T.F. Cootes, C.J. Taylor, D.H. Cooper, J. Graham, Active shape models—their training and application, CVGIP: Image Understanding 61 (1) (1994) 38–59.

[7] N. Daucher, M. Dhome, J.T. Lapreste, G. Rives, Modelled object pose estimation and tracking by monocular vision, British Machine Vision Conference, BMVC'93, Guildford, UK, 1993, pp. 249–258.

[8] D. Dementhon, L. Davis, Model-based object pose in 25 lines of codes, Int. J. Comput. Vision 15 (1995) 123–141.

[9] T. Drummond, R. Cipolla, Real-time tracking of complex structures for visual servoing, in: B. Triggs, A. Zisserman, R. Szeliski (Eds.), Vision Algorithms: Theory and Practice, LNCS 1883Springer, Berlin, 2000, pp. 69–84.

[10] B. Espiau, F. Chaumette, P. Rives, A new approach to visual servoing in robotics, IEEE Trans. Robot. Automat. 8 (3) (1992) 313–326.

[11] J.D. Foley, A. Van Dam, Fundamentals of Interactive Computer Graphics, Addison Wesley, USA, 1982.

[12] D.B. Gennery, Visual tracking of known three-dimensional objects, Int. J. Comput. Vision 7 (3) (1992) 243–270.

[13] N. Giordana, P. Bouthemy, F. Chaumette, F. Spindler, J.-C. Bordas, V. Just, 2D model-based tracking of complex shapes for visual servoing tasks, in: M. Vincze, G. Hager (Eds.), Robust Vision for Vision-based Control of Motion, IEEE Press, 2000, pp. 67–75 (Chapter 6).

[14] M. Haag, H.-H. Nagel, Combination of edge element and optical flow estimates for 3D-model-base vehicle tracking in traffic images sequences, Int. J. Comput. Vision 35 (3) (1999) 295–319.

[15] G. Hager, K. Toyama, The X vision system: a general-purpose substrate for portable real-time vision applications, Comput. Vision Image Understanding 69 (1) (1998) 23–37.

[16] S. Hutchinson, G. Hager, P. Corke, A tutorial on visual servo control, IEEE Trans. Robot. Automat. 12 (5) (1996) 651–670.

[17] J.K. Jain, A.K. Jain, Displacement measurement and its application in interframe image coding, IEEE Trans. Commun. 29 (12) (1981) 1799–1808.

[18] M. Kass, A. Witkin, D. Terzopolous, Snakes: active contour models, in: Proceedings of the International Conference on Computer Vision, ICCV'87, London, UK, 1987, pp. 259–268.

[19] C. Kervrann, F. Heitz, A hierarchical Markov modeling approach for the segmentation and tracking of deformable shapes, Graph. Models Image Process. 60 (3) (1998) 173–195.

[20] H. Kollnig, H.-H. Nagel, 3D pose estimation by directly matching polyhedral model to gray value gradients, Int. J. Comput. Vision 23 (3) (1997) 283–302.

[21] H. Kollnig, H.-H. Nagel, Matching object models to segments from an optical flow fields, European Conference on Computer Vision, LNCS 1065, vol. II, Springer, Cambridge, UK, 1996, pp. 14–18.

[22] D.G. Lowe, Robust model-based motion tracking through the integration of search and estimation, Int. J. Comput. Vision 8 (2) (1992) 113–122.

[23] E. Marchand, Visp: a software environment for eye-in-hand visual servoing, in: IEEE International Conference on Robotics and Automation, ICRA'99, vol. 4, Detroit, MI, 1999, pp. 3224–3229.

[24] E. Marchand, P. Bouthemy, F. Chaumette, V. Moreau, Robust real-time visual tracking using a 2D–3D model-based approach, in: IEEE International Conference on Computer Vision, ICCV'99, vol. 1, Kerkira, Greece, 1999, pp. 262–268.

[25] J.-M. Odobez, P. Bouthemy, Robust multiresolution estimation of parametric motion models, J Visual Commun. Image Represent. 6 (4) (1995) 348–365.

[26] J.-M. Odobez, P. Bouthemy, E. Fleuet, Suivi 2D de pièces métalliques en vue d'un asservissement visuel, in: 11th Congrès de reconnaissance des formes et d'intelligence artificielle, RFIA'98, vol. 2, Clermont-Ferrand, 1998, pp. 173–182.

[27] A. Pece, A. Worrall, A statistically-based Newton method for pose refinement, Image Vision Comput. 16 (8) (1998) 541–544.

[28] T.N. Tan, G.D. Sullivan, K.D. Baker, Model-based localisation and

recognition of road vehicles , Int. J. Comput. Vision 27 (1) (1998) 5–26.

[29] M. Tonko, K. Schäfer, V. Gengenbach, H.-H. Nagel, Multi-level 3D tracking of objects integrating velocity estimation based on optical flow and Kalman-filtering, in: Proceedings of the Fourth International Symposium on Experimental Robotics, ISER'95, LNCS 223, Springer, Stanford, USA, July 1995, pp. 212–221.

[30] W. Wilson, C. Hulls, G. Bell, Relative end-effector control using cartesian position-based visual servoing, IEEE Trans. Robot. Automat. 12 (5) (1996) 684–696.