# Real-Time Hybrid Tracking using Edge and Texture Information

Muriel Pressigout[1] and Eric Marchand[2]

[1] Université de Rennes 1, IRISA, Lagadic, F-35042 Rennes, France ;[*]
[2] INRIA, IRISA, Lagadic, F-35042 Rennes, France.[†]

Published in
The International Journal of Robotics Research
Vol 26, No 7, July 2007

## Abstract

This paper proposes a real-time, robust and effective tracking framework for visual servoing applications. The algorithm is based on the fusion of visual cues and on the estimation of a transformation (either a homography or a 3D pose). The parameters of this transformation are estimated using a non-linear minimization of a unique criterion that integrates information both on the texture and the edges of the tracked object. The proposed tracker is more robust and performs well in conditions where methods based on a single cue fail. The framework has been tested for 2D object motion estimation and pose computation. The method presented in this paper has been validated on several video sequences as well as in visual servoing experiments considering various objects. Results show the method to be robust to occlusions or textured backgrounds and suitable for visual servoing applications.

**Keywords : Visual Tracking, Visual Servoing, Hybrid Tracking**

## 1 Introduction

Development of object tracking algorithms is an important issue for applications related to visual servoing and more generally for robot vision. A robust extraction and real-time spatio-temporal tracking process of image motion/object's pose is indeed one of the keys to success of a visual servoing task. To consider

[*]Contact : `Muriel.Pressigout@irisa.fr` // +33 (0) 2 99 84 73 05.
[†]Contact : `Eric.Marchand@irisa.fr` // +33 (0) 2 99 84 74 27.

visual servoing within large scale applications, it is now fundamental to handle natural scenes without any fiducial markers and with complex objects in various illumination conditions. From a historical perspective, the use of fiducial markers allowed the validation of theoretical aspects of visual servoing research. Even if such features are still useful to validate new control laws, it is no longer possible to limit ourselves to such techniques if the final objectives are the transfer of these technologies in realistic applications.

Most of the available tracking techniques can be divided into two main classes: 2D image and 3D pose-based tracking. The former approaches mainly focus on tracking 2D features such as geometrical primitives (points [33, 44], segments [5, 21, 35], circles [35, 49],. . . ) or object contours [3, 4], regions of interest [20],. . . The latter explicitly use a 3D model of the tracked objects [11, 12, 13, 14, 15, 18, 29, 32, 38, 46, 48].

**Edge-based tracking.** Regarding the low level information that is extracted from the images, one can consider edge-based information or texture-based information. Edge-based trackers rely on the high spatial gradients outlining the contour of the object or some geometrical features of its pattern (points, lines, circles, distances, splines,...). When 2D tracking is considered, such edge points enable to estimate the geometrical features parameters whose values define the position of the object [21]. Snakes or active contours can be used to outline a complex shape [4]. If a 3D model of the object is available [12, 15], edge-based tracking is closely related to the pose estimation problem and is therefore suitable for any visual servoing approach. In general, edge-based techniques have proved to be very effective for applications that require a fast tracking process. Nevertheless, they may fail in the presence of highly textured environments.

**Texture-based tracking.** On the other hand, texture information has been widely used for object tracking. Contrarily to edge-based trackers, it is well adapted to textured objects and does usually less suffer from jittering. However, this solution is not appropriate for poorly textured objects and is mainly exploited in 2D tracking, such as the KLT algorithm [44] or region of interest tracking [2, 20, 28]. Points or regions of interest can also be used within a 3D model-based tracking as reported in [48] where the camera viewpoint can be estimated by minimizing the projection errors of the different points of interest, or as in [27] where the grey level values are integrated directly in the minimization process of the 3D tracking. Furthermore these approaches usually lack of precision if there is a significant difference between current and reference texture scales.

As one can note, model-based trackers can be mainly divided in two groups, the edge-based ones and the textured-based ones. Both have complementary advantages and drawbacks. The idea is then to integrate both approaches in the same process. This paper addresses the problem of robust tracking of 2D and 3D objects by closely integrating edge and texture information. Considering various kind of features in a tracking received little attention in the literature.

**Hybrid tracking overview** Among approaches to cue integration one can find i) a sequential use of the available information (mainly motion and edges), ii) probabilistic approaches such as Extended Kalman Filter or particle filter, iii) voting approaches and iv) registration process of the different cues within the same minimization process. We try to analyze these different approaches.

Some methods rely on a sequential estimation of motion and of 2D or 3D edge-based registration in order to combine robustness and accuracy, as in [1, 10, 36, 7]. In these approaches, motion estimation (dominant motion or optical flow) provides a prediction of the edge (*i.e.* , of the 2D object location) which is helpful for the edge-based registration step and improves tracking reliability. Nevertheless, although both motion and edges are (sequentially) considered these are not strictly hybrid algorithms and these approaches do not take benefit of several advantages from using them simultaneously.

Most of the current approaches that integrate multiple cues in a tracking process are probabilistic techniques. Most of these approaches rely on the well known Kalman filter, its non-linear version the Extended Kalman filter (EKF) or particle filter. [45] fuses measurements of the object's center of mass using color information, edge orientations and positions and some feature displacements obtained by a SSD minimization of the grey level difference between the current image and the prediction in a Kalman filter. [31] integrates the outputs from two trackers (a 3D model-based tracker [15] and a point of interest tracker) using an EKF. [19] fuses edge-based tracking and optical-flow estimation within an Iterated Extended Kalman Filter to update object position. Let note that many approaches rely on a particle filtering as [26] or Probabilistic Multiple Hypothesis Tracker (PMHT) [43] but are usually very slow. 2D visual cues fusion using voting has also been studied in [30] and considered for visual servoing applications. However, this work is not directly related to edge and texture fusion.

In [48] the proposed model-based approach considers both 2D-3D matching against a key-frame that represents a single pose as in a classical model-based approach but considering multiple hypothesises for the edge tracking and 2D-2D temporal matching (which introduces multiple view spatio-temporal constraints in the tracking process). A nice extension is proposed in [47] to integrate contribution of an edge-based tracker similar to [12, 15]. The work of [39] extends the tracker of [27] by integrating contour information in the case of planar structures. In this latter approach a global error function (that considers both distance to the edge and difference of intensity) is defined and the Jacobian that links the variation of a homography to the variation of the feature vector is learnt using the approach presented in [27].

The framework presented in this paper fuses a classical model-based approach based on the edge extraction and a temporal matching relying on texture analysis into a single non-linear objective function that has then to be minimized. Tracking is formulated in terms of a full scale non-linear optimization. We will consider within the same framework both a 2D and a 3D tracker. Dealing with the 2D tracker, our goal is to define a unique state vector that describes both the appearance of the template as well as its edge boundaries. Considering this state vector, we are able to compute the parameters of a 2D transformation (a homography) that minimizes the error between a current multi-cue template and the transformed reference one. When considering a 3D tracker, estimating both pose and camera displacement introduces an implicit spatio-temporal constraint a 3D model-based tracker based on edge features lacks of. This general framework is used to create a system which is capable of treating complex scenes in *real-time*. To improve robustness, an M-estimator is integrated in a robust control law. The resulting pose or displacement computation algorithm is thus able to deal effectively with incorrectly tracked features that usually degrade the performance and result in a failure.

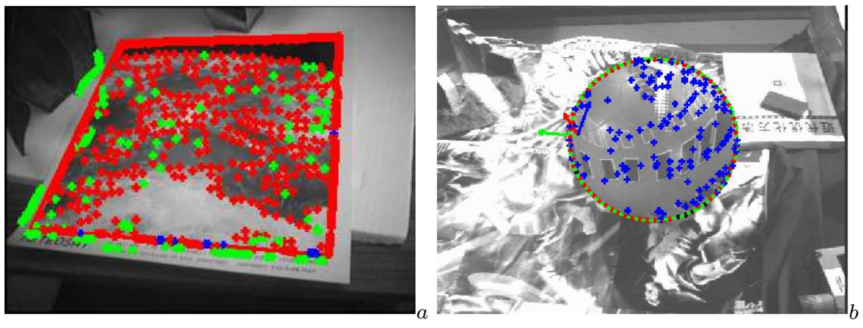


Figure 1: Tracking issues. (a) estimating the 2D position of an object in the image : its outline can be determined all along the sequence, (b) retrieving the position and the orientation of the object in the 3D space : the frame of the scene with respect to the camera is estimated in every image. Both problems are addressed in this paper using the same hybrid transformation estimation framework.

In the remainder of this paper, section 2 presents the principle of the approach. Two different tracking issues are addressed in sections 3 and 4 as illustrated in Figure 1. The section 3 deals with the estimation of the 2D object position in the image by applying this general framework to the estimation of 2D transformation, a homography. The pose computation issue is described in section 4 to estimate the pose of the object in the 3D space, once again using the same framework. Finally, in order to validate this approach both trackers are tested on several realistic image sequences as well as used as an input to a visual servoing experiments. Those experimental results are reported in Section 5.

## 2 Tracking : general framework

This section is dedicated to the description of the general framework of the algorithm. It is based on a transformation (either a 2D homography or a 3D pose) estimation that exploits image information. This transformation estimation is first described in subsection 2.1. After the introduction of different image information used in this scheme in subsection 2.2, their fusion in the proposed framework is explained in subsection 2.3.

### 2.1 General 2D or 3D transformation estimation

Whatever the tracking considered, either the estimation of the 2D object position in the image or its pose in the 3D space with respect to the camera, the process relies on the estimation of a transformation. The framework presented in this section describes the estimation process of this transformation, disregarding the model of the transformation.

This transformation is parametrized by $M$ parameters $\mu_i$ stored in a vector $\mu$. $\mu_t$ will be the notation for the current transformation for the image $\mathbf{I}_t$. Its estimation relies on the analysis of image features $\mathbf{s}$. The first subsection presents the basis of the estimation process, whatever the image features $\mathbf{s}$, then its robust version. Subsections 2.2 and 2.3 will describe the different image features that will be considered in this work and their fusion in the transformation estimation process.

The value of the current image features $\mathbf{s}_{\mu_t}$ estimated according to $\mu_t$ depends on stored data $\mathbf{x}$ and on $\mu_t$:

$$\mathbf{s}_{\mu_t} = f(\mu_t, \mathbf{x}) \tag{1}$$

The observations extracted from the image $\mathbf{I}_t$ provide a ground truth $\mathbf{s}^*$ for these features. The idea is to determine the transformation parameters that minimize the difference between those desired values and the current ones, *i.e.* to estimate $\mu_t$ that minimizes the error $\Delta$ such as:

$$\Delta = \sum_{i=1}^{n} (s^i_{\mu_t} - s^{*i})^2 \tag{2}$$

If each image feature is stored in a vector $\mathbf{s} = (s^1, \ldots, s^i, \ldots, s^n)^\top$, it comes to minimize the error vector $\mathbf{e}$ defined by:

$$\mathbf{e} = \mathbf{s}_{\mu_t} - \mathbf{s}^* \tag{3}$$

If an exponential decrease of the error is specified:

$$\dot{\mathbf{e}} = -\lambda \mathbf{e} \tag{4}$$

where $\lambda$ is a positive scalar, one then has:

$$\dot{\mathbf{e}} = \frac{\partial \mathbf{s}_{\mu_t}}{\partial \mu_t} \frac{d\mu_t}{dt} = -\lambda \mathbf{e} \tag{5}$$

With $\mathbf{J}_{\mathbf{s}_{\mu_t}} = \frac{\partial \mathbf{s}_{\mu_t}}{\partial \mu_t}$ and $\frac{d\mu_t}{dt} = \delta\mu$, a vector can be computed such as:

$$\delta\mu = -\lambda \mathbf{J}^+_{\mathbf{s}_{\mu_t}} (\mathbf{s}_{\mu_t} - \mathbf{s}^*) \tag{6}$$

where $\mathbf{J}^+_{\mathbf{s}_{\mu_t^k}}$ is the pseudo-inverse of the Jacobian $\mathbf{J}_{\mathbf{s}_{\mu_t}}$[1], in order to update the vector $\mu_t$ at each iteration of the iterative minimization process:

$$\mu_t^{k+1} = \mu_t^k \oplus \delta\mu \tag{7}$$

with $\mu_t^0 = 0$ until the error is minimized. $\oplus$ is an update operator that depends on the considered transformation. It will be explained later for each case. The final $\mu_t$ is the vector that stores the estimated transformation.

Since input data are extracted from the images, the process is sensitive to outliers originating from noise, occlusions, mismatching, *etc* and a robust optimization has to be performed [23, 40]. Equation 2 can be rewritten by:

$$\Delta = \sum_{i=1}^{n} \rho(s^i_{\mu_t} - s^{*i}) \tag{8}$$

where $\rho(u)$ is a robust function [23] that grows sub-quadratically and is monotonically non-decreasing with increasing $|u|$. Iteratively Re-weighted 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.

The error to be regulated to zero is thus defined as $\mathbf{e} = \mathbf{D}(\mathbf{s}_{\mu_t} - \mathbf{s}^*)$ where $\mathbf{D} = diag(w_1, \ldots, w_n)$ is a diagonal weighting matrix. The weights $w_i$, which represent the different elements of the $\mathbf{D}$ matrix, reflect the confidence of each feature. In our case these weights are computed using the Tukey M-estimator [23].

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

Tukey's estimator allows to completely reject outliers and gives them a zero weight. A complete description of the way to compute $w_i$ is given in [12]. The update of the transformation parameters is now given by:

$$\delta\mu = -\lambda(\mathbf{D}\mathbf{J}_{\mathbf{s}_{\mu_t^k}})^{+}\mathbf{D}(\mathbf{s}_{\mu_t} - \mathbf{s}^*) \tag{9}$$

## 2.2 Visual features

Any kind of geometrical feature can be considered within the proposed framework as soon as it is possible to compute its corresponding Jacobian matrix $\mathbf{J}$. It is 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:

$$\dot{\mathbf{s}} = \begin{bmatrix} \dot{\mathbf{s}}^1 \\ \vdots \\ \dot{\mathbf{s}}^n \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{s^1} \\ \vdots \\ \mathbf{J}_{s^n} \end{bmatrix} \delta\mu = \mathbf{J}_s \delta\mu \tag{10}$$

The redundancy yields more accurate result with the computation of the pseudo-inverse of $\mathbf{J}$ as given in equation (6). Furthermore if the number or the nature of visual features is modified over time, the interaction matrix $\mathbf{J}$ and the vector error $\mathbf{s}$ is easily modified consequently.

Two kinds of visual features will be considered in this work: edge-based and texture-based features. Their description is given in the following paragraphs.

**Edge-based features** In this case, the visual features $\mathbf{s}$ are composed of a set of distances $d_\perp$ (see Figure 2) between local point features $\mathbf{p}_t$ obtained from an edge-based tracker (described in appendix A) and the contours of the object $\mathcal{C}$. In this case, the desired value $\mathbf{s}^*$ is zero. An assumption is made that the contours of the object in 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.

Minimizing (2) using only such features comes to minimize:

$$\Delta = \sum_{i=1}^{n} (d_\perp(\mathbf{p}_t^i, \mathcal{C}_{\mu_t}))^2 \tag{11}$$

where $\mathcal{C}_{\mu_t}$ denotes the geometrical features that outline the object contour estimated according to the current transformation parameters $\mu_t$. Note that the parameters of the object contours observed in the image do not need to be estimated.

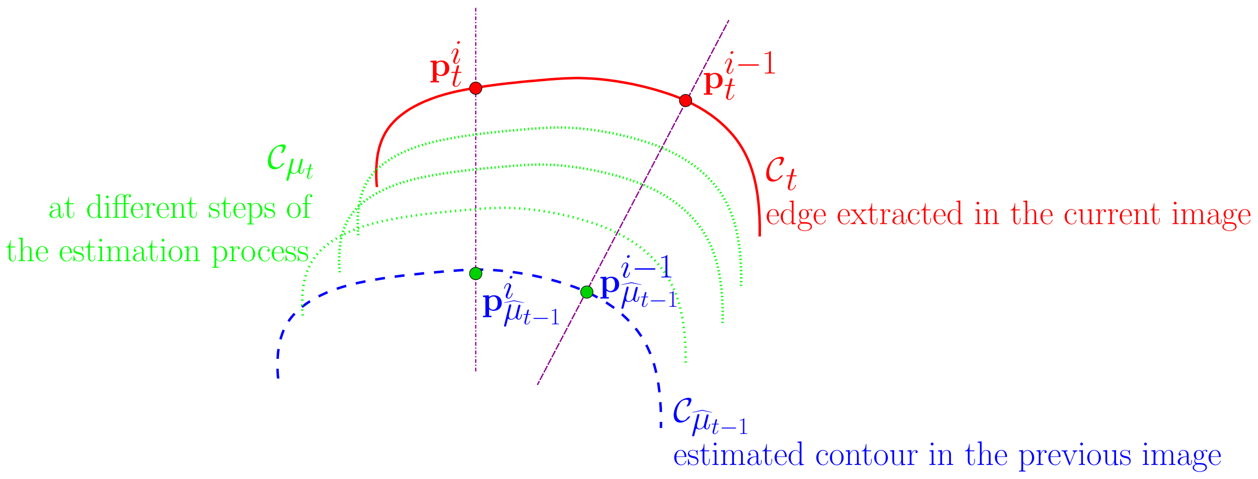

$\mathbf{p}_t^i$
$\mathbf{p}_t^{i-1}$
$\mathcal{C}_{\mu_t}$
at different steps of the estimation process
$\mathcal{C}_t$
edge extracted in the current image
$\mathbf{p}^i_{\widehat{\mu}_{t-1}}$
$\mathbf{p}^{i-1}_{\widehat{\mu}_{t-1}}$
$\mathcal{C}_{\widehat{\mu}_{t-1}}$
estimated contour in the previous image

Figure 2: Edge-based tracking

Point-to-contour distances avoid a matching step that is necessary to algorithms that estimate the motion by minimizing a point-to-point distance. As an example, in the Iterative Closest Point algorithm [17], at each iteration of the minimization process, point matching must be performed before estimating the transformation parameters.

An edge-based tracker is fast, effective and robust to illumination changes. However, it is mainly a mono image process. As a consequence, if the geometrical features can not be accurately extracted without any ambiguity, the tracker may lack of precision. This sensitivity to the textureness of the object or the background may lead to jittering effects or even divergence.

**Texture-based features** Second type of features are grey levels $I_t(\mathbf{p})$ that describe the pattern of the object in image $\mathbf{I}_t$. With the constant illumination assumption, the desired values of such features is given by $\mathbf{s}^* = I_t(\mathbf{p}_{\mu_t^*}) = I_0(\mathbf{p}_0)$ and the current value of the features by $\mathbf{s}_{\mu_t} = I_t(\mathbf{p}_{\mu_t})$.

Minimizing (2) using only such features becomes:

$$\Delta = \sum_{i=1}^{n} (I_t(\mathbf{p}^i_{\mu_t}) - I_0(\mathbf{p}^i_0))^2 \tag{12}$$

The initial samples extracted from a reference image $\mathbf{I}_0$ are chosen following the Harris criteria to select locations that will give some reliable information about the motion. Indeed, the Jacobian matrix of such a feature depends on the image spatial gradient $\nabla\mathbf{I}$ and the Jacobian matrix of the point location:

$$\mathbf{J}_{\mathbf{s}_{\mu_t}} = \nabla\mathbf{I}(\mathbf{p}_{\mu_t})^\top \mathbf{J}_{\mathbf{p}_{\mu_t}} \tag{13}$$

A small camera motion with respect to the object can lead to a large image intensity change. To avoid the systematic elimination of the most interesting points of the pattern, the image gradient is taken into account in the weight computation. Indeed, $\| \nabla I \|$ is a good measure of reliability of the point. The larger $\| \nabla I \|$, the more significant the measure of the intensity difference $I_t(\mathbf{p}^i_{\mu_t}) - I_0(\mathbf{p}^i_0)$. So we prefer to consider the intensity difference weighted by the norm of the spatial intensity gradient: the following normalized vector $(\ldots, \frac{I_t(\mathbf{p}^i_{\mu_t})-I_0(\mathbf{p}^i_0)}{\|\nabla I_0(\mathbf{p}^i_0)\|}, \ldots)^\top$ is used to compute the M-estimators instead of the error $\mathbf{s}_{\mu_t} - \mathbf{s}^*$. Similar normalizations are used in [25, 41].

If only texture-based features are exploited in the framework to estimate the transformation, the process is relatively robust if the object is textured. It is however sensitive to scale and illumination changes.

## 2.3 Merging features

As already said, any kind of features can be considered in the presented framework. Using equation (10) not only enables to consider several features but also several types of features of different nature. If there are $N_c$ edge-based features (*i.e.* point-to-contour distances) and $N_t$ texture-based features (*i.e.* grey level samples), one has:

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}^1 \\ \vdots \\ \mathbf{s}^{N_c+N_t} \end{bmatrix} \tag{14}$$

where:

$$\mathbf{s}^i = \begin{cases} d_\perp(\mathbf{p}^i, \mathcal{C}) & if \quad i \leq N_c \\ I_t(\mathbf{p}^i) & if \quad i > N_c \end{cases} \tag{15}$$

Merging two different types of features is quite simple. However one must care of the order of magnitude of each one. Indeed, a point-to-contour distance is far smaller than an intensity difference and thus the edge-based features may have not enough influence on the minimization process. As a consequence, a normalization is performed respectively on each error vector (one storing the edge-based error, the other the texture-based error) before stacking them in equation (3) such as each of their terms belongs to the interval $[-1; 1]$. This is done by computing the maximal absolute value of the errors associated to the edge-based (resp. texture-based) features and dividing the error vector associated to the edge-based (resp. texture-based) features by this maximal value.

The tracking framework described in this section applies for different kinds of features as it has been said but also for various transformation models. The two next sections will be dedicated to two cases. Section 3 deals with the estimation of the object position in the image, *i.e.* relies on a 2D transformation estimation, more specifically a homography estimation, and section 4 with the camera pose/displacement computation. In each case, details about the image features and their Jacobian matrix are given.

## 3 2D tracking: Homography estimation

Here, we consider 2D tracking and therefore the problem is to estimate the position of the object in a video sequence. We work there in the 2D space of the images, therefore the process relies on the estimation of a 2D transformation.

Different types of models have been studied in the literature: pure translation, affine, homography, *etc*. The most generic transformation for a planar structure is a homography since it is able to account for the full 3D motion of such a structure.

In the case of an homographic model, the points of an image are linked to those ones of another image of the same planar structure by a $3 \times 3$ matrix $\mathbf{H}$ :

$$\mathbf{p}_{\mu_t} \propto \mathbf{H}\mathbf{p}_{\hat{\mu}_{t-1}} \tag{16}$$

Therefore, there are nine parameters to be estimated such as:

$$\mu = (\mu_0, \dots, \mu_8) \tag{17}$$

and:

$$\mathbf{H} = \begin{pmatrix} \mu_0 & \mu_1 & \mu_2 \\ \mu_3 & \mu_4 & \mu_5 \\ \mu_6 & \mu_7 & \mu_8 \end{pmatrix} \tag{18}$$

and if $\mathbf{H}_t^{k+1}$, $\mathbf{H}_t^k$ and $\delta\mathbf{H}_t$ denote respectively the homography matrices obtained from $\mu_t^{k+1}$, $\mu_t^k$ and $\delta\mu$ as defined in equation( 7), the update operator is given by:

$$\mathbf{H}_t^{k+1} = \mathbf{H}_t^k \; \delta\mathbf{H}_t \tag{19}$$

The transformation being defined, the image features can be more precisely described and their Jacobian computed.

### 3.1 Edge-based features

Let us recall that the edge-based features are point-to-contour distances and that using only such features comes to minimize (11). If we call $\epsilon_j$ the geometrical feature parameters describing the contours $\mathcal{C}$, the general analytical form of the Jacobian matrix of $\mathbf{s}_{\mu_t}$ is:

$$\mathbf{J}_{\mathbf{s}_{\mu_t}} = \sum_j \frac{\partial d_\perp(\mathbf{p}_t, \mathcal{C}_{\mu_t})}{\partial \epsilon_j} \frac{\partial \epsilon_j}{\partial \mu_t} \tag{20}$$

The 2D tracking has been implemented for two classes of contours: piecewise linear contours, described by lines and curved contours, described by NURBS. The approach used to extract this low level information is described in Annex A.

**Lines.** In this case, the parameters $\epsilon_j$ are the three coefficients $a_{\mu_t}$, $b_{\mu_t}$ and $c_{\mu_t}$ that define the line according to the current 2D transformation parameters by:

$$x_t a_{\mu_t} + y_t b_{\mu_t} + c_{\mu_t} = 0 \tag{21}$$

where $\mathbf{p}_t = (x_t, y_t)$ is a point belonging the line. In the previous image, this contour is represented by the estimated coefficients $a_{\hat{\mu}_{t-1}}$, $b_{\hat{\mu}_{t-1}}$ and $c_{\hat{\mu}_{t-1}}$.

The feature $\mathbf{s}_{\mu_t}$ is given by:

$$\mathbf{s}_{\mu_t} = d_\perp(\mathbf{p}_t, \mathcal{C}_{\mu_t}) = \frac{x_t a_{\mu_t} + y_t b_{\mu_t} + c_{\mu_t}}{r} \tag{22}$$

where $r = \sqrt{a_{\mu_t}^2 + b_{\mu_t}^2}$ and its Jacobian matrix:

$$\mathbf{J}_{\mathbf{s}_{\mu_t}} = \frac{x_t\ r - a_{\mu_t} d_\perp}{r^2} \mathbf{J}_{a_{\mu_t}} + \frac{y_t\ r - b_{\mu_t} d_\perp}{r^2} \mathbf{J}_{b_{\mu_t}} + \frac{1}{r} \mathbf{J}_{c_{\mu_t}} \tag{23}$$

with $d_\perp = d_\perp(\mathbf{p}_t, \mathcal{C}_{\mu_t})$ to simplify the notations.

$\mathbf{J}_{a_{\mu_t}}$, $\mathbf{J}_{b_{\mu_t}}$ and $\mathbf{J}_{c_{\mu_t}}$ are the respective Jacobian matrices of $a_{\mu_t}$, $b_{\mu_t}$ and $c_{\mu_t}$. They are detailed in Annex B as well as the update of $\mathcal{C}\mu_t$ along the sequence.

**NURBS.** In this case, the parameters $\epsilon_j$ are the coordinates $\mathbf{Q}_i = (\alpha_i, \beta_i)^\top$ and the weights $w_i$ of the control points $\mathbf{Q}_i$ of the NURBS which is defined as follows [42]:

$$\mathcal{C}(s) = \sum_{i=0}^{n} R_{i,p}(s) \mathbf{Q}_i \tag{24}$$

$R_{i,p}$ are the rational basis functions, they are piecewise rational functions on $s \in [0; 1]$ defined by:

$$R_{i,p}(s) = \frac{N_{i,p}(s) w_i}{\sum_{j=0}^{n} N_{j,p}(s) w_j} \tag{25}$$

$N_{i,p}$ are the B-spline basis functions, they are piecewise polygonal functions on $s \in [0;1]$. The NURBS are more general curves than B-splines, their main advantage being their invariance to perspective transformation thanks to the weight associated with each control point of the curve. A NURBS is therefore updated from an image to another simply by applying the homographic transformation to its control points [42] considering the weights as their third homogeneous coordinate.

The distance between a point and the curve is approximated by the distance between the point and the line tangent to the NURBS. The minimization problem is then similar to the piecewise linear outline case since a distance between a point and a line is considered. The selection of the points to be tracked is such as there is the same number of points for each span of the NURBS, evenly spread.

### 3.2 Texture-based features

As said in section 2, the texture-based features are samples of the grey levels of the object pattern and minimizing (2) with only such features comes to minimize (12). The reference image is the image in which the initial sampling is performed in the first image of the sequence.

The general form of the Jacobian matrix is given by (13) and using (16), $\mathbf{J}_{\mathbf{p}_{\mu_t}}$ is given by:

$$\mathbf{J}_{\mathbf{p}_{\mu_t}} = \frac{1}{w_{\hat{\mu}_{t-1}}} \begin{pmatrix} x_{\hat{\mu}_{t-1}} & y_{\hat{\mu}_{t-1}} & 1 & 0 & 0 & 0 & -x_{\hat{\mu}_{t-1}} x_{\mu_t} & -y_{\hat{\mu}_{t-1}} x_{\mu_t} \\ 0 & 0 & 0 & x_{\hat{\mu}_{t-1}} & y_{\hat{\mu}_{t-1}} & 1 & -x_{\hat{\mu}_{t-1}} y_{\mu_t} & -y_{\hat{\mu}_{t-1}} y_{\mu_t} \end{pmatrix} \quad (26)$$

The texture-based 2D tracker is similar to the work proposed by [20] and extended to homography estimation by various authors such as [2, 8, 28].

## 4 3D tracking : camera pose/displacement computation

In this section, the general framework will be applied for the pose computation problem. Now the tracking is performed in the 3D space and requires a 3D model of the object. The position and the orientation of the camera with respect to the scene has to be determined, *i.e.* six parameters: three for the position and three for the rotations of axes. It is supposed the intrinsic parameters $\xi$ are available but it is possible, using the same approach, to also estimate these parameters. One thus has $\mu = (t_x, t_y, t_z, r_x, r_y, r_z)$.

The pose matrix ${}^{ct}\mathbf{M}_w$ (obtained from $\mu$) links the 3D features ${}^{w}\mathbf{P}$ of the object, expressed in the world frame, to their projection $\mathbf{p}$ in the image by:

$$\mathbf{p} = pr_\xi({}^{ct}\mathbf{M}_w, {}^{w}\mathbf{P}_i) \quad (27)$$

where $pr_\xi({}^{ct}\mathbf{M}_w, {}^w\mathbf{P}_i)$ is the chosen projection model. For a point with a simple perspective projection model, we have:

$$\mathbf{p} = \mathbf{K}\ {}^{ct}\mathbf{M}_w\ {}^w\mathbf{P} \tag{28}$$

where $\mathbf{K}$ is a projective matrix obtained from the intrinsic parameters $\xi$.

If ${}^{ct}\mathbf{M}_w^{k+1}$, ${}^{ct}\mathbf{M}_w^k$ and $\delta\mu$ denote respectively the matrices obtained from $\mu_t^{k+1}$, $\mu_t^k$ and $\delta\mu$ as defined in equation (7), the update operator is given by:

$$^{ct}\mathbf{M}_w^{k+1} = {}^{ct}\mathbf{M}_w^k\ \ e^{[\delta\mu]} \tag{29}$$

where $e^{[\delta\mu]}$ is the exponential map of SE(3) of $\delta\mu$ computed using the Rodrigues' formula (e.g.[34], p. 33).

$\delta\mu$ can be seen as a virtual camera velocity and the pose/displacement estimation as the process that enables a virtual camera to align the observation in the image with the projection of the scene in its image plane [12, 37]. To illustrate the principle, consider the case of an object with various 3D features $\mathbf{P}$ (for instance, ${}^w\mathbf{P}$ are the 3D coordinates of object points in the object frame). A virtual camera is defined whose position and orientation in the object frame is defined by $\mu$. The approach consists of estimating the real pose or displacement by minimizing the error $\Delta$ between the observed data $\mathbf{s}^*$ and the current value $\mathbf{s}_\mu$ of the same features computed by forward-projection according to the current pose/displacement:

$$\Delta = \sum_{i=1}^{n} (s_\mu^i - s^{*i})^2 \tag{30}$$

In this formulation of the problem, a virtual camera (initially at $\mu_t$) is moved using a visual servoing control law in order to minimize this error $\Delta$. At convergence, the virtual camera reaches the pose $\mu^*$ which minimizes this error. $\mu^*$ is the real camera pose we are looking for.

As it will be seen, the edge-based features enable to perform a pose computation while the texture-based features are more suitable for a camera displacement estimation but it will be shown it is the same problem.

## 4.1 Edge-based features

When edge-based features are considered, the pose computation is performed as in a classical model-based tracker [12, 15, 32]. The approach consists of estimating the real camera pose by minimizing the error $\Delta$ between the observed data $\mathbf{s}^*$ and the position $\mathbf{s}_{\mu_t}$ of the same features computed by a forward-projection according to the current pose:

$$\Delta = \sum_{i=1}^{n} \left(pr_\xi(\mu, {}^w\mathbf{P}_i) - \mathbf{s}^{*i}\right)^2 = \sum_{i=1}^{n} \left(d_\perp(\mathbf{p}_t^i, \mathcal{C}_{\mu_t})\right)^2 \tag{31}$$

where $pr_\xi(\mu, {}^w\mathbf{P})$ is the projection model according to the intrinsic parameters $\xi$ and camera pose $\mu$, $\mathbf{s}^{*i} = d_\perp(\mathbf{p}_t^i, \mathcal{C}_t) = 0$ as said in section 2 and $\mathbf{s}_{\mu_t}^* = d_\perp(\mathbf{p}_t^i, \mathcal{C}_{\mu_t})$, $\mathcal{C}_{\mu_t}$ being computed by the projection of the 3D model in the image according to the current pose parameters $\mu_t$. At convergence, the pose minimizing the error $\Delta$ is assumed to be the real one.

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 [12]. In Figure 3, $\mathbf{p}$ is the tracked point and $\mathbf{l}(\mu)$ is the current line feature position.

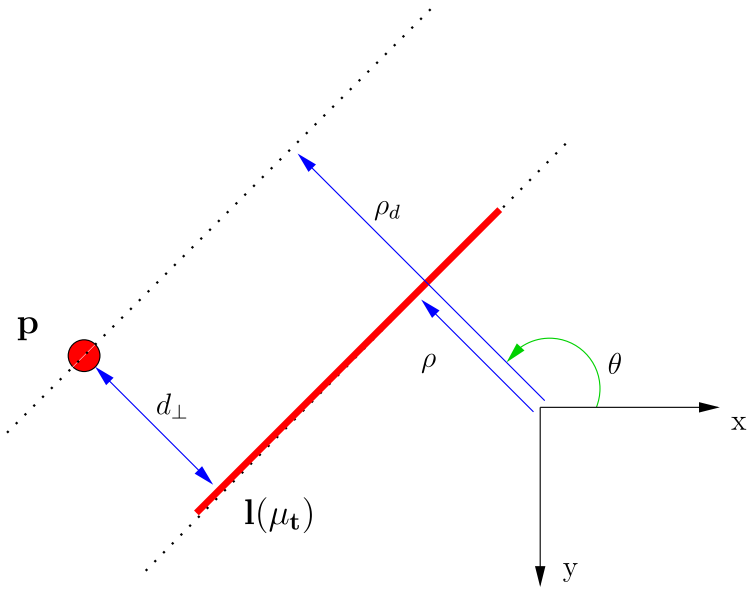


Figure 3: Distance of a point to a straight line

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

$$x\cos\theta + y\sin\theta = \rho, \forall(x,y) \in \mathbf{l}(\mu), \tag{32}$$

The distance between point $\mathbf{p}$ and line $\mathbf{l}(\mu)$ 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:

$$d_l = d_\perp(\mathbf{p}, \mathbf{l}(\mu)) = \rho(\mathbf{l}(\mu)) - \rho_d, \tag{33}$$

where

$$\rho_d = x_d\cos\theta + y_d\sin\theta, \tag{34}$$

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

$$\dot{d_l} = \dot{\rho} - \dot{\rho_d} = \dot{\rho} + \alpha\dot{\theta}, \tag{35}$$

where $\alpha = x_d \sin\theta - y_d \cos\theta$. Deduction from (35) gives the Jacobian related to $d_l$: $\mathbf{J}_{d_l} = \mathbf{J}_\rho + \alpha \mathbf{J}_\theta$. $\mathbf{J}_{d_l}$ can be thus derived from the Jacobian related to a straight line given by (see [16] for its complete derivation):

$$\begin{array}{lcl} \mathbf{J}_\theta & = & [\ \lambda_\theta \cos\theta \quad \lambda_\theta \sin\theta \quad -\lambda_\theta\rho \quad \rho\cos_\theta \quad -\rho\sin_\theta \quad -1\ ] \\ \mathbf{J}_\rho & = & [\ \lambda_\rho \cos\theta \quad \lambda_\rho \sin\theta \quad -\lambda_\rho\rho \quad (1+\rho^2)\sin\theta \quad -(1+\rho^2)\cos\theta \quad 0\ ] \end{array} \tag{36}$$

where $\lambda_\theta = (A_2 \sin_\theta - B_2 \cos_\theta)/D_2$, $\lambda_\rho = (A_2 \rho\cos_\theta + B_2 \rho\sin_\theta + C_2)/D_2$, and $A_2X + B_2Y + C_2Z + D_2 = 0$ is the equation of a 3D plane which the line belongs to.

From (35) and (36) the following is obtained:

$$\mathbf{J}_{d_l} = \begin{bmatrix} \lambda_{d_l}\cos_\theta \\ \lambda_{d_l}\sin_\theta \\ -\lambda_{d_l}\rho \\ (1+\rho^2)\sin_\theta - \alpha\rho\cos_\theta \\ -(1+\rho^2)\cos_\theta - \alpha\rho\sin_\theta \\ -\alpha \end{bmatrix}^\top, \tag{37}$$

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

### 4.2 Texture-based features

As said in section 2, the texture-based features are samples of the grey levels of the object pattern and minimizing (2) with only such features corresponds to minimizing (12)).

The geometry of a multi-view system (or of a moving camera) introduce very strong constraints in feature location across different views. In the general case, the point transfer can be achieved considering the epipolar geometry and the essential or fundamental matrices (see, for example, [22]). In this paper we restrict ourselves to the less general case where point transfer can be achieved using a homography. Since any kind of 3D motion must be considered, this means that the texture lies on a plane in the 3D space. We first suppose that the object is piecewise planar and then release this assumption.

**Planar structure.** This case is quite similar to the 2D case but the homography is now computed from the camera displacement parameters [22]:

$$\mathbf{p}_t \propto \mathbf{K}^{-1}\ {}^{c_t}\mathbf{H}_{c_0}\ \mathbf{K}\ \mathbf{p}_0 \tag{38}$$

with:

$$ {}^{ct}\mathbf{H}_{c_0} = {}^{ct}\mathbf{R}_{c_0} + \frac{{}^{ct}\mathbf{t}_{c_0}}{d_0}\,\mathbf{n}_0^\top \tag{39} $$

where $\mathbf{n}_0$ and $d_0$ are the normal and distance to the origin of the reference plane expressed in the camera reference frame. ${}^{ct}\mathbf{R}_{c_0}$ and ${}^{ct}\mathbf{t}_{c_0}$ are respectively the rotation matrix and the translation vector between the two camera frames.

**Non-planar structure.** In the case of a non-planar structure, the point transfer given by (38) becomes [22]:

$$ \mathbf{p}_t \propto \mathbf{K}^{-1}\, {}^{ct}\mathbf{H}_{c_0}\, \mathbf{K}\, \mathbf{p}_0 + \beta_0 \mathbf{c}_t \tag{40} $$

where ${}^{ct}\mathbf{H}_{c_0}$ is the homography induced by a reference plane $\pi$ as seen previously, the scalar $\beta_0$ is the parallax relative to the homography ${}^{ct}\mathbf{H}_{c_0}$ and $\mathbf{c}_t = \mathbf{K}\ {}^{ct}\mathbf{t}_{c_0}$ the epipole projected onto the image $\mathbf{I}_t$ in pixel coordinates. $\beta_0$ may be interpreted as a depth relative to the plane $\pi$:

$$ \beta_0 = \frac{d_0 - \mathbf{n}_0^\top (Z_0 \mathbf{K}^{-1}\mathbf{p}_0)}{Z_0\ d_0} \tag{41} $$

with $Z_0$ the depth coordinate of the 3D point associated with $\mathbf{p}_0$ expressed in camera frame 1. As $\beta_0$ depends only on parameters expressed in the camera reference frame, it can be precomputed. The value of $Z_0$ is given by the intersection of the 3D structure and the ray passing through the camera center and $\mathbf{p}_0$.

**Jacobian matrix.** Independent of the object shape, the Jacobian matrix $\mathbf{J}_{\mathbf{p}_{\mu_t}}$ is estimated using (13) with:

$$ \mathbf{J}_{\mathbf{p}_{\mu_t}} = \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \begin{pmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & (1+y^2) & -xy & -x \end{pmatrix} \tag{42} $$

**From displacement estimation to pose estimation.** The texture-based 3D tracker presented here relies on the camera displacement parameters. However, estimating the camera displacement or its pose is similar since a virtual camera velocity $\delta\mu$ is computed, which is equivalent if the camera pose ${}^{c0}\mathbf{M}_w$ in the reference image $\mathbf{I}_0$ is known, since:

$$ {}^{ct}\mathbf{M}_w = {}^{ct}\mathbf{M}_{c_0}{}^{c0}\mathbf{M}_w \tag{43} $$

Combining both approaches allows to introduce a spatio-temporal constraint in the pose estimation by considering information in the current and past images and the underlying multi-view geometrical constraints.

**Multiscale model.** The displacement estimation has been presented for two images $\mathbf{I}_0$ and $\mathbf{I}_t$. In practice, $\mathbf{I}_t$ is the current image for which the camera pose has to be estimated and $\mathbf{I}_0$ a reference image of the tracked plane. There is a reference image $\mathbf{I}_{0i}$ for each plane $\pi_i$ with texture to track on the object. In the case of a piecewise planar structure, there are then as many reference images as planes, such as the plane is not too far from a fronto-parallel position with respect to the image plane. If it is a non-planar structure, it is necessary to cover the whole object. Such a set of images is also used in [48], however, the features extracted from these images is different (points locations *versus* grey levels). The model of the object is then composed of the CAD model for the edge-based part of the tracker and the reference images for the texture-based one. A pose computation is performed for each reference image using the edge-based model-based tracker to get the plane parameters with respect to the camera frame needed in (39) and the depth computation. The homographies needed to transfer the points are computed at each step for each plane following (39). Because of (43), they all depend on the same pose parameters and by consequence tracking different planes is not an issue.

If several planes are tracked, the number of grey-level samples per plane must be updated at each image since the visibility of each plane changes. If there are $n_t$ grey level samples to be considered in the minimization process, the number of grey level samples $n_{t_i}$ belonging to the plane $\pi_i$ to be involved in the minimization process is $n_{t_i} = \frac{n_t}{\sum_i a_i} a_i$ where $a_i$ is the area of the plane $\pi_i$ in the image, $a_i$ being equal to 0 if the plane $\pi_i$ is not visible. For each reference image, $n_t$ points are subsampled following a trade-off between the Harris criteria and covering as much as possible the whole pattern to enforce the robustness of the tracking [44]. In Figure 4, an example is given for each object tracked in the experiment Section. Depending on the visibility of the plane, a set of these samples will be updated and tracked following the rule given above.

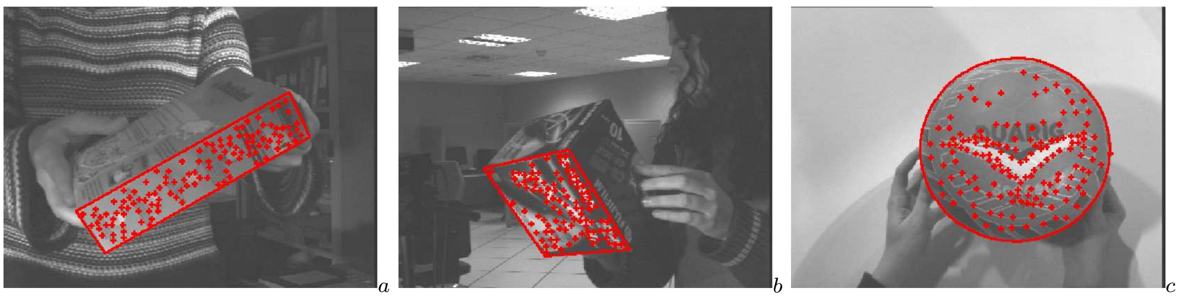


Figure 4: Texture model for a face of: (a) a rice box, (b) a DVD box, (c) a ball

This model of the object patterns enable to depict its textureness for a given camera-to-object distance interval. A pyramid of reference images will be now introduced to represent the grey levels that best describe the object for larger distances. From the reference image $\mathbf{I}_{0i}$ associated with a plane $\pi_i$, $K$ images $\mathbf{I}_{0i}^k$ are built using a Gaussian filter and a sub-sampling step. Image $\mathbf{I}_{0i}^k$ is obtained from image $\mathbf{I}_{0i}^{k-1}$ by:

$$\mathbf{I}_{0i}^k = f(\mathbf{G} \otimes \mathbf{I}_{0i}^{k-1}) \tag{44}$$

where $f$ is the sub-sampling function , $\mathbf{G}$ a Gaussian filter and $\otimes$ the convolution operator. The bottom of the pyramid is given by the reference image $\mathbf{I}_{0i} = \mathbf{I}_{0i}^1$. Only one of the images $\mathbf{I}_{0i}^k$ will be tracked if the plane $\pi_i$ is visible. The choice is simply based on the current distance between the object and the camera. As we assume that the reference images are not too far from a fronto-parallel position with respect to the image plane, if $\alpha$ is the object distance from the camera for image $\mathbf{I}_{0i}$, then the distance $\alpha_k = 2^k\alpha$ is associated with image $\mathbf{I}_{0i}^k$. The comparison with the current distance allows to select the image that is the nearest to the current one. Figure 5 shows a pyramid obtained for a face of the DVD box. The Harris selection will be performed for each image of this pyramid.

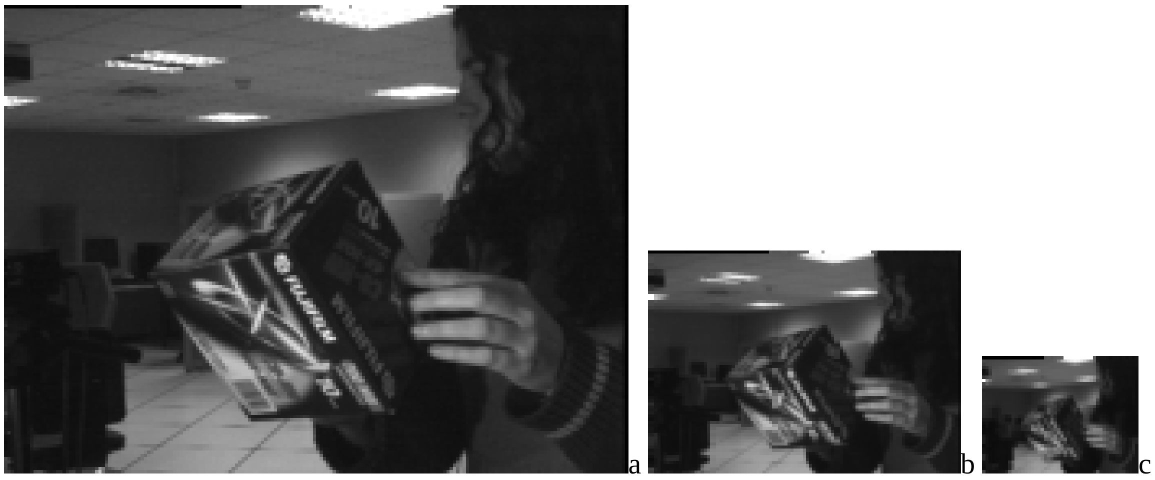


Figure 5: Pyramid obtained for a face. (a) Level 1, (b) Level 2, (c) Level 3

## 5 Experimental Results

This section presents some qualitative and quantitative results for the homography estimation and the pose computation. In each case, experiments have been performed to validate the efficiency of the tracker which is then applied to visual servoing positioning tasks [16, 24]. Visual servoing aims to control a robotic system such as it realizes a given task by exploiting the information extracted from the images acquired by a camera.

The precision of the tracking is therefore a key point of the success or failure of such a task.

Each experiment is performed using i) the edge-based features, ii) the texture-based features and iii) their integration hybrid tracker. The edge locations and/or the texture points used in the minimization process are displayed in the first image of each sequence.

## 5.1 Results on homography estimation

To begin with, experiments have been performed to validate the efficiency of the tracker. The two first experiments test the tracker on objects whose contours are modeled first by lines and then by NURBS. It is then applied to visual servoing tasks.

In each experiment red crosses are used for inliers and green ones for outliers. Blue crosses are used for edge locations that are not sharp enough and therefore not used in the tracking process. The object position in each image is given by the current outline in red. During the visual servoing experiment, the desired position is described by the green outline.

### 5.1.1 Video sequences

**Tracking a piecewise linear object.** In this experiment, a video sequence is captured. The tracked object is outlined by four lines. The edge-based tracker diverges quite quickly (see Figure 6(b)), mistaken by the neighboring sharp edges and the texture-based tracker slowly drifts (see Figure 6(a)), especially when occlusions occur. However, the complementarity of the two kinds of features and the robust estimation process enable the hybrid tracker to succeed (see Figure 6(c)). On Figure 7, one can see the occluded parts are well-detected and withdrawn from the minimization by a low confidence weight.

**Tracking a curved-shaped picture** The framework has been applied to objects outlined by a NURBS as described in this section. Figure 8 is an example of such a tracker. The object to track is a picture of an apple. The challenge here is to obtain an accurate contour, which is quite difficult due to the background and the shadow. Once again, the only tracker that succeeds to track the object is the hybrid one (see Figure 8(c)). The edge-based is misled by the shadows that are very near the real object outline and ends to be attracted by texture in the neighborhood and the texture-based one drifts on one side. As previously, the selected features are shown in the first image. The red crosses are for the inliers ones and the green crosses for the features considered as outliers.

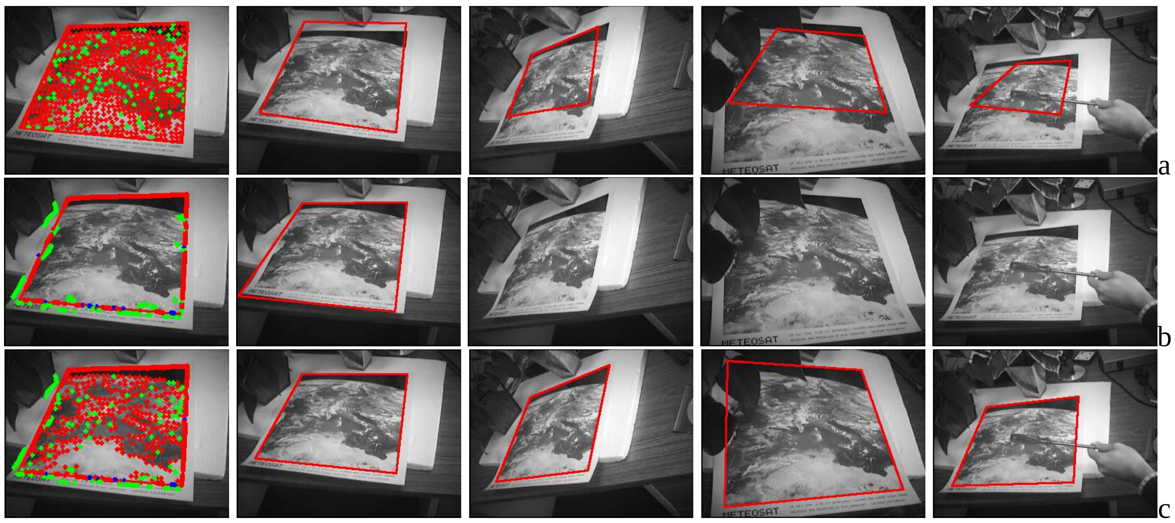


Figure 6: Tracking a planar structure. Images for: (a) the texture-based tracker, (b) the edge-based tracker and (c) the hybrid tracker. The green crosses are points associated with features considered as outliers (due to noise, occlusions or shadow) and the red ones are for the inliers ones. Blue crosses are used for edge locations that are not sharp enough and therefore not used in the tracking process. The hybrid is the only one that succeeds to track the object, although significant occlusions occur.

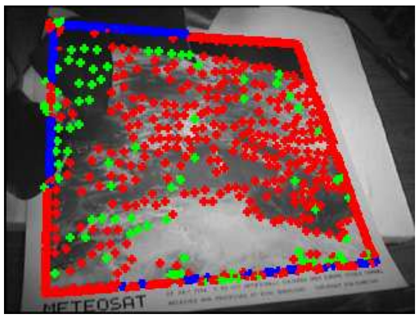


Figure 7: Tracking a planar structure. Example of an occlusion detection. The green crosses are points associated with features considered as outliers (due to noise, occlusions or shadow) and the red ones are for the inliers ones. Blue crosses are used for edge locations that are not sharp enough and therefore not used in the tracking process.

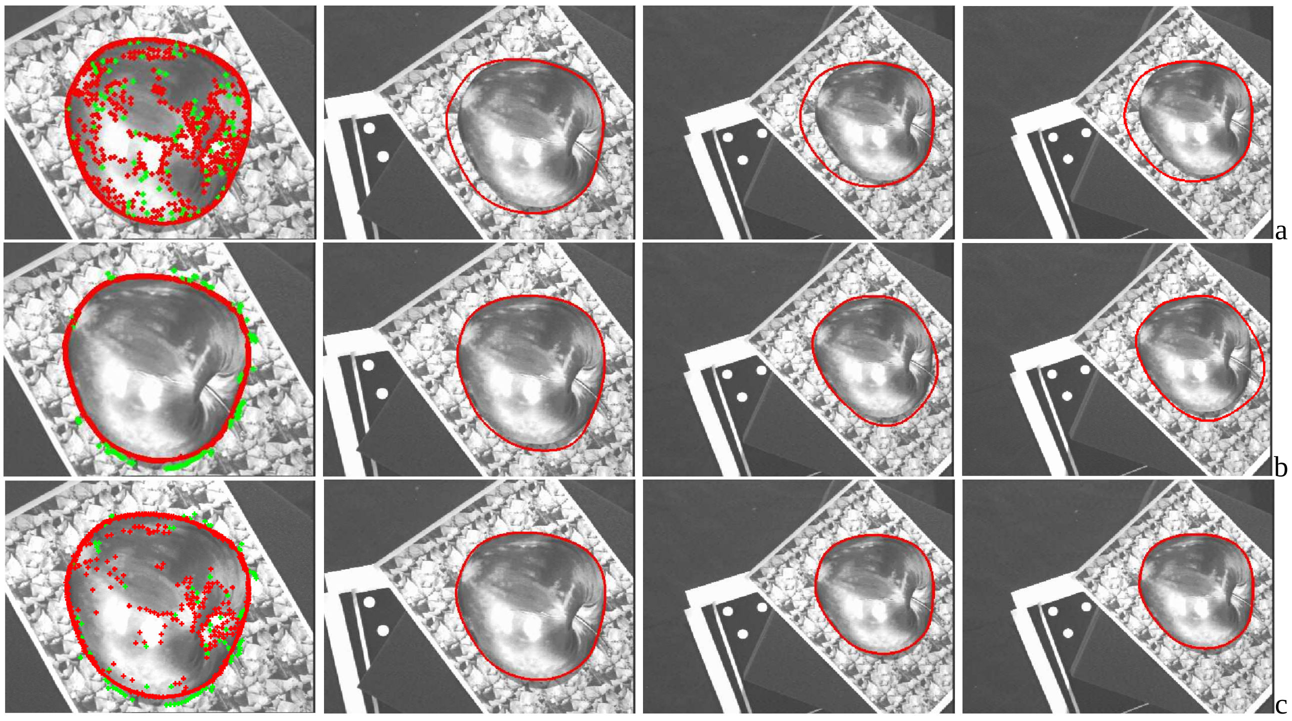


Figure 8: Apple sequence: NURBS tracking. Initial and final images. The hybrid tracker (c) succeeds to track the object while the two other ones fail((a) and (b)). The green crosses are points associated with features considered as outliers (due to noise, occlusions or shadow) and the red ones are for the inliers ones.

### 5.1.2 Others experiments : outdoor environment and significant motions

The tracker presented in this paper has been tested in various conditions: hand-held moving objects, camera mounted on a robot, *etc*. Outdoor environment has also been studied, for example to track a building facade as in Figure 9. Attention has also been paid to significant motions as shown in Figure 10 where the maximal motion of the object during the experiment is displayed.

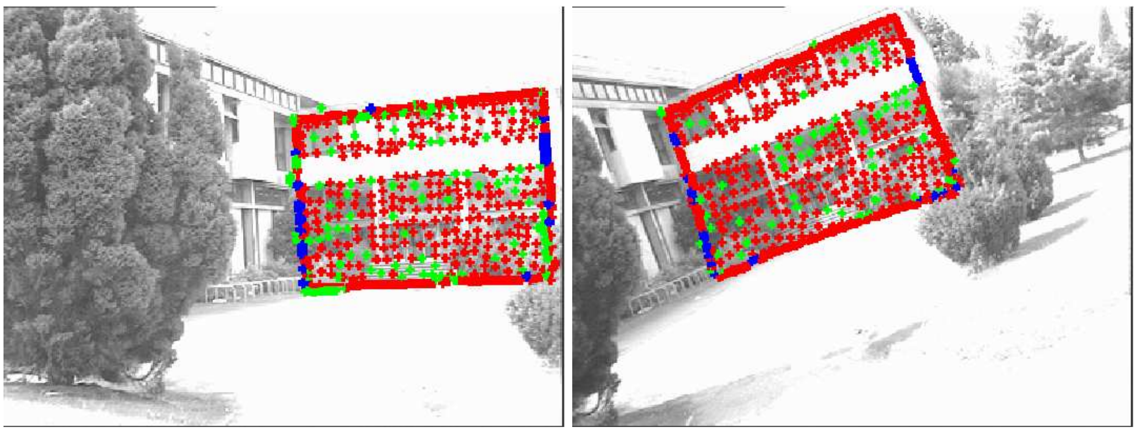


Figure 9: Outdoor environments. The scene, rich in contours and texture, is made of planes and therefore, the hybrid algorithm is an effective one for such an application

### 5.1.3 Visual servoing positioning task based on image moments

The task, here a positioning task, is specified by a set of desired features $\mathbf{x}^*$ associated to its desired position in the image. The velocity of a camera mounted on the end-effector of a 6 d.o.f robot is controlled such that the error between the desired features $\mathbf{x}^*$ and the current value of the features $\mathbf{x}$ gets minimized. The camera velocity $\mathbf{v}$ that is computed to move the robot is such as:

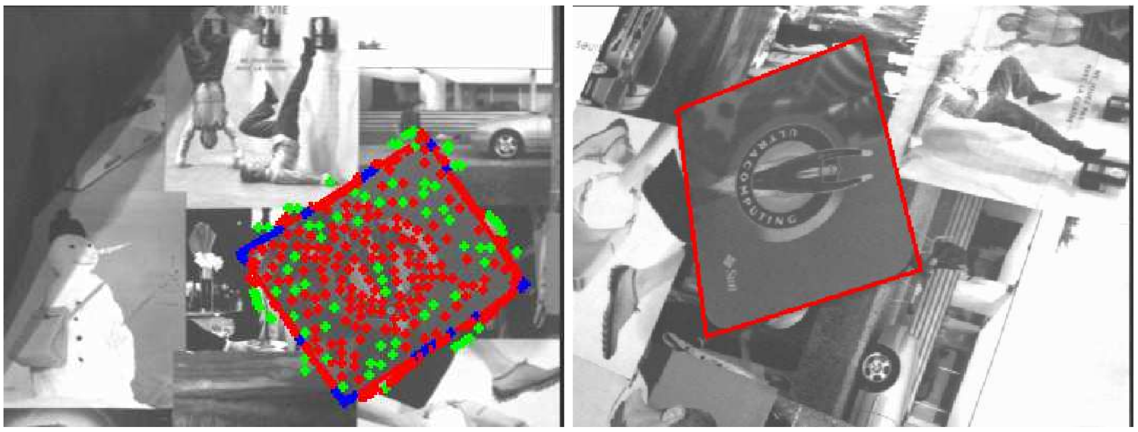
$$\mathbf{v} = -\lambda \mathbf{L_x}^+ (\mathbf{x} - \mathbf{x}^*) \tag{45}$$

where $\mathbf{L_x}$ is the interaction matrix related to $\mathbf{x}$ (which links the motion of $\mathbf{x}$ in the image to the camera velocity: $\dot{\mathbf{x}} = \mathbf{L_x}\mathbf{v}$).

In this experiment, image moments are used in the control law to achieve the task [9]. We then have:

$$\mathbf{x} = (x_g, y_g, a, p_x, p_y, \theta) \tag{46}$$

where $x_g$ and $y_g$ are the coordinates of the center of gravity of the object, $a$ its area, $\theta$ its orientation in the image, $p_x$ and $p_y$ depending of moments of order 3 as described in [9]



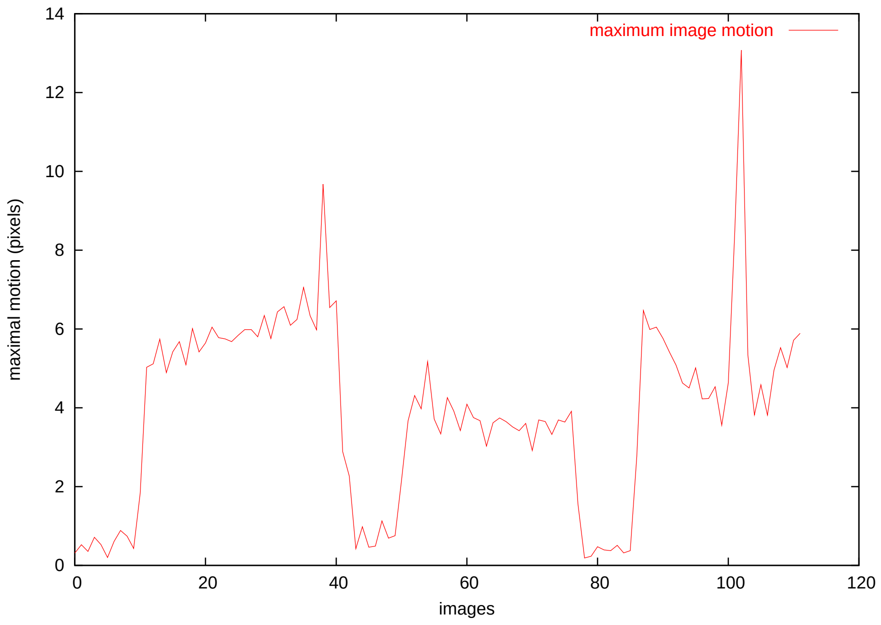
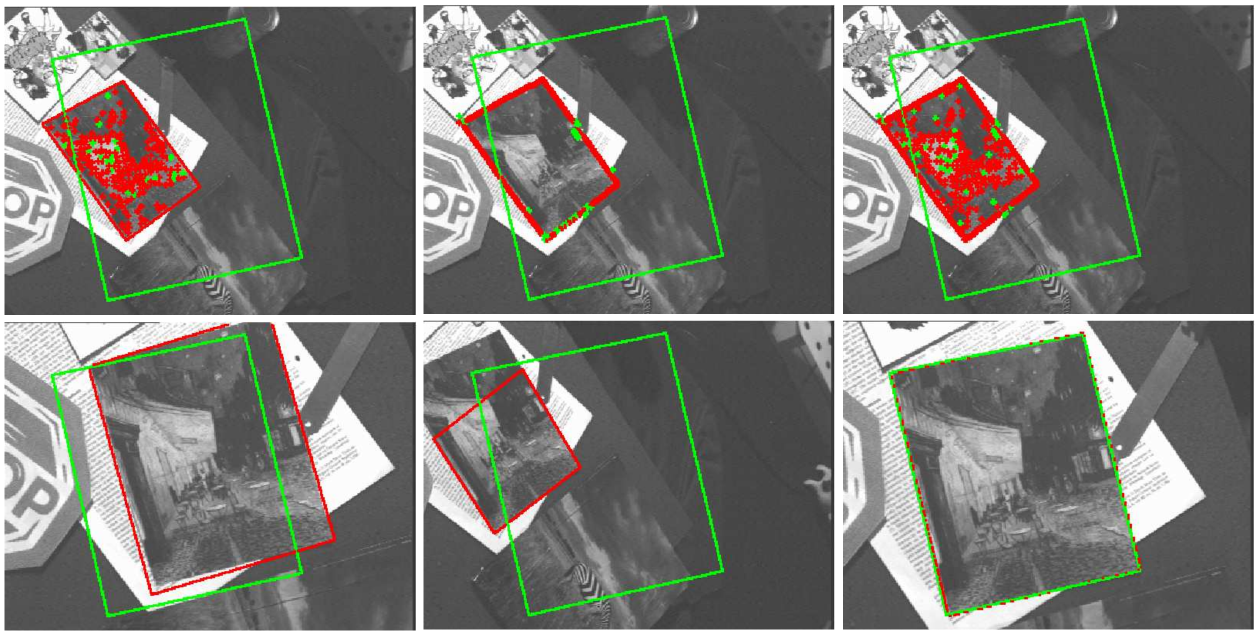


Figure 10: Significant motions. The merging of contour-based and texture-based results in an approach that is more robust to large motions. Sudden increases or decreases of the object motion in the image are also considered in this experiment.

The first experiment will show the accuracy of the positioning task although significant occlusions occur when using the hybrid tracker and the second one compares its accuracy to a single-based one when both succeed to track the object.

**Visual servoing with occlusions.** In this experiment, the task is performed four times: once to test the output of each tracker when no occlusion occurs and once again to test the output of the hybrid tracker when multiple occlusions occur.

The initial and final images of the experiment performed without occlusion are shown in Figure 11. One can see the tracking was not successful in the single cue cases. Although the tracker proposed in this paper is slower than the single-cue trackers (near video rate for the hybrid tracker and the texture-based one, three time faster for the edge-based tracker), the experiments show that it is better than the single-cue ones. Using only intensity information is not accurate enough because of the object scale changes during the experiment and because of the poor texture in the areas of the pattern where the drift begins. In such cases, the edge-based features are important to adjust more accurately the object position in the image.



(a) texture-based (b) Edge-based (c) Hybrid

Figure 11: First 2D visual servoing experiments without occlusion. Green rectangle: desired position of the object in the image. Initial and final images. Only the hybrid tracker performs a good tracking. The edge-based tracker completely diverges and the texture-based one lacks of accuracy.

Some of the intermediate and the final images of the experiment performed with occlusions are shown in Figure 12. The green crosses are points associated with features considered as outliers (due to noise, occlusions or shadow) and the red ones are for the inlier ones. Hidden edge locations are represented in blue. One can see that the occluded parts are well detected.

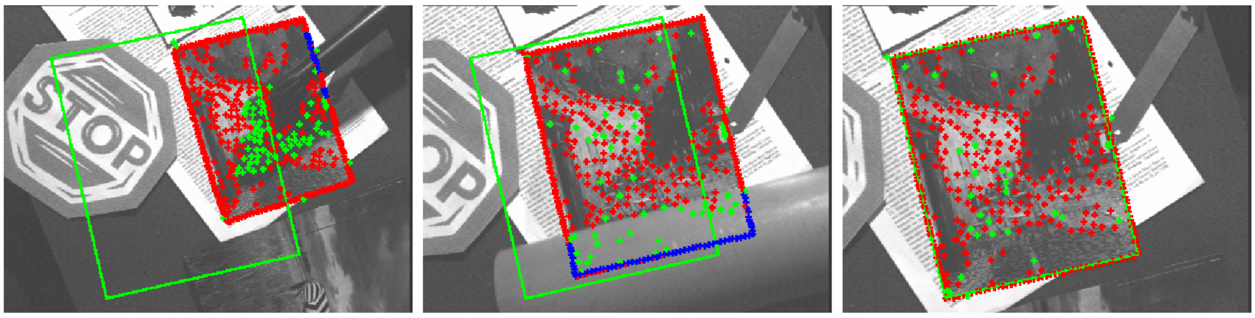
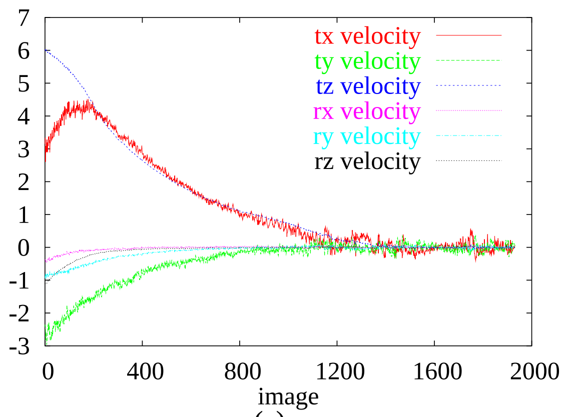


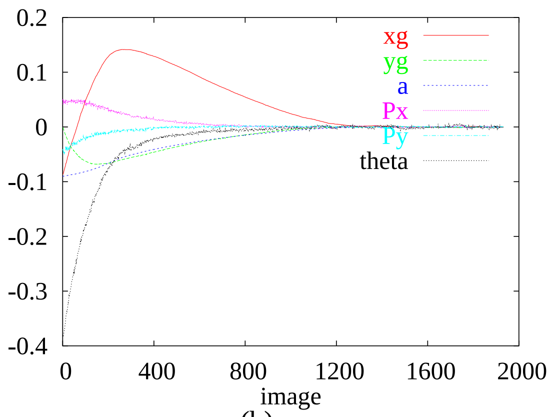
Figure 12: First 2D visual servoing experiment with occlusions. Green rectangle : desired position of the object in the image. The green crosses are points associated with features considered as outliers (due to noise, occlusions or shadow) and the red ones are for the inliers ones.

The output of the hybrid tracker enables a good behavior of the camera and the positioning task is correctly achieved. In Figure 13(a), the evolution of the camera velocity is shown, as well as the error between the desired features and the current ones in Figure 13(b). The camera displacement is smooth and the accuracy of our tracker enables to achieve a very good positioning. In Figure 14, the desired position and the two final ones (without and with occlusions) obtained using the hybrid tracker are presented. The positioning is well achieved in both cases: the error on the camera pose is below 1 degree on rotation and 5 mm on translation when no occlusion occurs and below 1.5 degree on rotation axis and 10 mm on translation when occlusions occur.

**Second visual servoing experiment.** In this experiment, contrarily to the previous one two trackers succeed: the texture-based one (see Figure 15(a)) and the hybrid one (see Figure 15(c)). The servo-control task has been stopped in the case of the edge-based case since the tracker completely diverges without any chance to recover (see Figure 15(b)). Although the texture-based tracker succeeds to track the object, the hybrid tracker achieves the positioning task with a better accuracy. As shown in Figure 16, the final camera position obtained with the hybrid tracker is closer to the desired one than the one obtained by the texture-based tracker. Furthermore, the velocity of the camera that depends on the output of the tracker is noisier when using this latter one (see Figure 17).



(a)

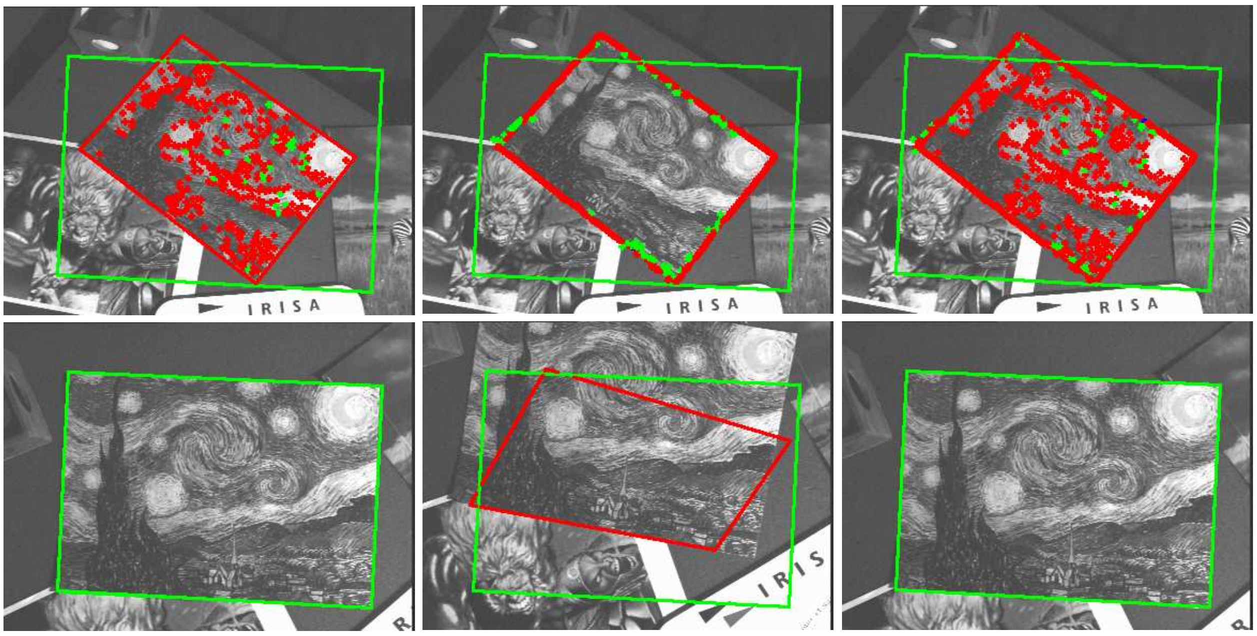


(b)

Figure 13: First 2D visual servoing without occlusion using our hybrid tracker. (a) camera velocity, translation velocities are in $cm/s$ and angle velocities in $degree/s$, (b) error in the image of each visual feature

| Axes | $t_x$ | $t_y$ | $t_z$ | $r_x$ | $r_y$ | $r_z$ |
|---|---|---|---|---|---|---|
| Desired pose | 40.3 | -5.1 | 30.0 | 17.8 | 0 | 5.2 |
| Final pose without occlusion(a) | 40.9 | -5.1 | 29.9 | 18.4 | 0 | 4.7 |
| Motion from this pose to the desired one | -0.6 | -0.6 | -0.1 | -0.6 | -0.2 | 0.5 |
| Final pose with occlusions (b) | 41.4 | -4.9 | 30.0 | 18 | -0.2 | 4 |
| Motion from this pose to the desired one | -1.1 | -1.1 | -0.1 | -0.2 | -0.2 | 1.2 |

Figure 14: First 2D visual servoing using our hybrid tracker: desired and final positions (a) case without occlusion, (b) case with occlusions. $t_x$, $t_y$ and $t_z$ are in $cm$ while $r_x$, $r_y$ and $r_z$ are in degrees.



(a) texture-based (b) Edge-based (c) Hybrid

Figure 15: Second 2D visual servoing experiments. Green rectangle: desired position of the object in the image. Initial and final images.The texture-based and the hybrid tracker perform a good tracking. The edge-based tracker completely diverges.

| Axes | $t_x$ | $t_y$ | $t_z$ | $r_x$ | $r_y$ | $r_z$ |
|---|---|---|---|---|---|---|
| Desired pose | 8.0 | -19.1 | 17.9 | 90.3 | 0.3 | 0.0 |
| Final pose with the texture-based tracker (a) | 7.3 | -21.2 | 17.5 | 91 | 0.9 | 2.1 |
| Motion from this pose to the desired one | 0.3 | 2 | 0.4 | -0.7 | 2.1 | -0.6 |
| Final pose with the hybrid tracker (b) | 7.9 | -18.7 | 17.9 | 90.2 | 0.4 | -0.5 |
| Motion from this pose to the desired one | 0.3 | -0.3 | 0 | 0.1 | -0.5 | -0.1 |

Figure 16: Second 2D visual servoing: comparison of the final positioning: desired and final positions: (a) texture-based tracker, (b) hybrid tracker. $t_x$, $t_y$ and $t_z$ are in $cm$ while $r_x$, $r_y$ and $r_z$ are in degrees. The hybrid tracker is more accurate than the texture-based tracker.

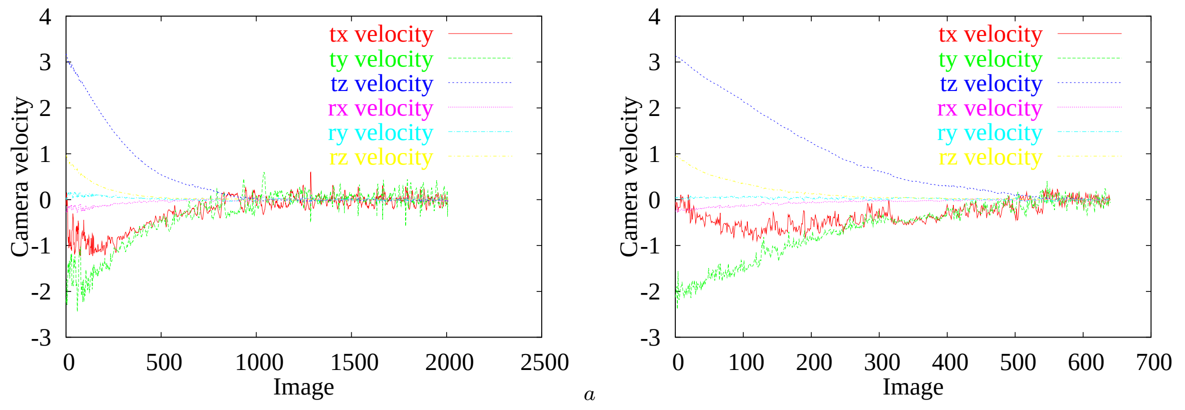


Figure 17: Second 2D visual servoing: comparison of the camera velocity. (a) texture-based tracker, (b) hybrid tracker. Translation velocities are in $cm/s$ and angle velocities in $degree/s$. The camera velocity with the texture-based tracker is noisier, which is something to be avoided.

## 5.2 Results on pose computation

As in the 2D case, some experiments will be first performed to validate the efficiency of the tracker and then its reliability will be shown in visual servoing experiments.

Blue crosses are used for inliers and green ones for outliers. Black crosses are used for edge locations that are not sharp enough and therefore not used in the tracking process. The object position in each image is given by the current outline in green. During the visual servoing experiment, the desired position is described by the red outline.

### 5.2.1 Video sequences

**Tracking a box.** In the considered image sequence, tracking the rice box is a very complex task since the object achieves a complete rotation. Therefore, the features to be tracked change as some faces appear or disappear. If the tracking begins to drift, it may be difficult to rectify the error, all the more that the light positions lead to big specularities and the background is quite complex. The object contours are permanently partially occluded by the hands or hardly visible: the edge-based tracker ends to lose the object (see Figure 18a). The object scale in the image is different from the one in the reference images, consequently this leads the texture-based tracker to fail to track the object quite quickly (see Figure 18b). However, even if each single-cue trackers were not sufficient for a good tracking in this image sequence, their fusion in the hybrid tracker enables to track the object correctly (see Figure 18c).

The camera pose parameters evolution is shown in Figure 19a and the evolution of the number of grey

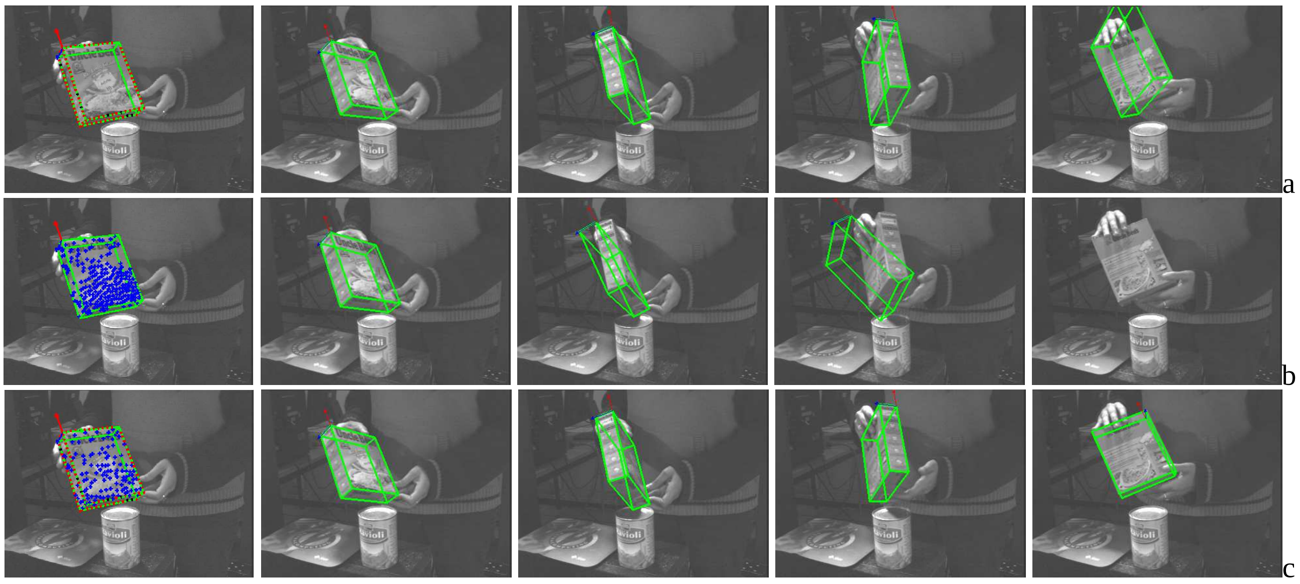


Figure 18: Rice box sequence. Images for (a): the edge-based tracker, (b): the texture-based one, (c): the hybrid one. Only the hybrid tracker succeeds to track correctly the object all along the sequence, despite the specularities and the misleading environment. The grey level samples are represented in the first image by blue crosses and the edge location by red points.

level samples used in the control law per face in Figure 19b. These curves are quite smooth and the output of the tracking is not prone to jittering. Let us note that the object being hand-held, the evolution of the pose is not regular. Figure 19c shows an example of specularity the tracker has to deal with. The grey level samples in the concerned area are considered as outliers by the M-estimators (they are drawn in green whereas the inliers are in blue) as well as a few ones in the top of the object covered by the shadow due to the hand. The hybrid tracker runs at an average rate of 25 Hz (see Figure 19d).

**Tracking a ball.** The difficulty of this experiment is to track a sphere which raises some illumination problems (permanent specularities,...). The contour-based tracker (Figure 20(b)) succeeds to track the contour of the ball but gives no information about the ball orientation. One can see the frame linked to the object remaining to the same place. The texture-based tracker (Figure 20(a)) succeeds to track the object for a while but ends to lose it due to the illumination changes. The hybrid tracker gives the full information about the ball position and orientation during the whole sequence (Figure 20(c)).

### 5.2.2 2 1/2 D visual servoing experiment

Figure 22 presents a first example of 2 1/2 D visual servoing task. In this case, the visual feature vector $\mathbf{x}$ is selected as $(\mathbf{t}, x, y, \theta u_z)$ where $\mathbf{t}$, expressed in the desired camera frame, is the translation that the camera has to realize, $x$ and $y$ are the coordinates of an image point, and $\theta u_z$ is the third component of vector $\theta\mathbf{u}$ (where $\theta$ and $\mathbf{u}$ are the angle and the axis of the rotation that the camera has to realize). $\mathbf{t}$ and $\theta\mathbf{u}$ are directly computed from the current estimated pose and the desired one.

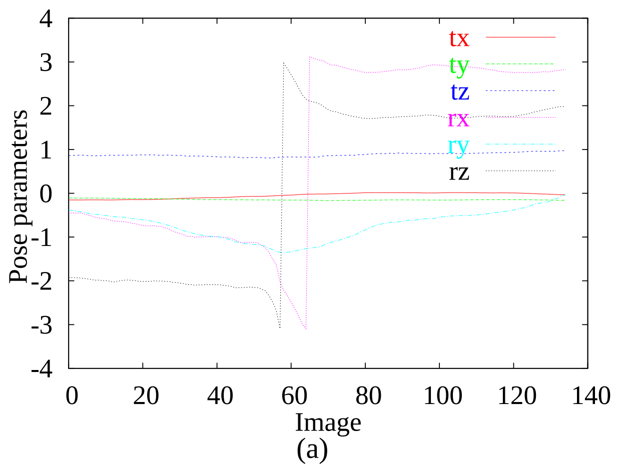
Similarly to the first VS experiment with the 2D tracker, the hybrid 3D tracker is able to perform an accurate positioning task while the two others trackers fail. In Figure 23(a), the evolution of the camera velocity is given and Figure 23(b) shows the task error decreasing. This leads to a precise positioning: the desired pose and the obtained one are given in Figure 24. The error in the positioning is below 1 cm for the position parameters and 1 degree for the orientation ones.

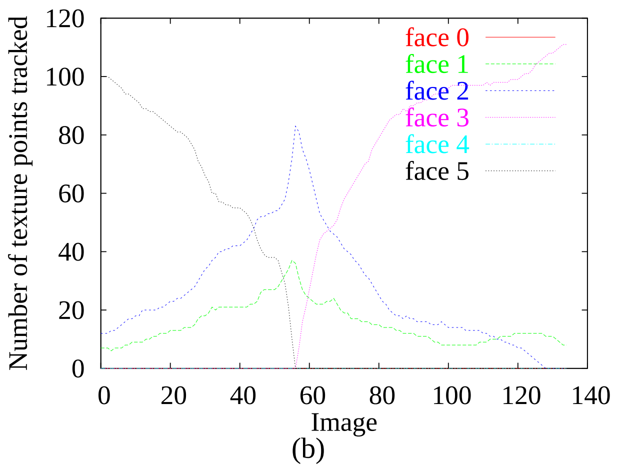
A more complex object is considered in the next experiment. The camera has to turn around the object to achieve the positioning task, which makes some parts of the object disappearing/appearing as illustrated in Figure 26. The experiment has been performed with each tracker without occlusion. The texture-based tracker fails immediately as the robot motion is quite significant at the beginning (see Figure 25(d)). The edge-based (see Figure 25(b)) and the hybrid tracker (see Figure 25(c)) both succeed to track the object and enable a precise positioning of the robot as presented in Figure 28. However, when other objects occlude the tracked one or are very near, the edge-based tracker drifts since the edges outlining the neighborhood mistake the pose estimation (see Figure 25(e)). However, the hybrid tracker is not sensitive to these occlusions and succeeds to track the object all along the positioning task (see Figure 25(f)).

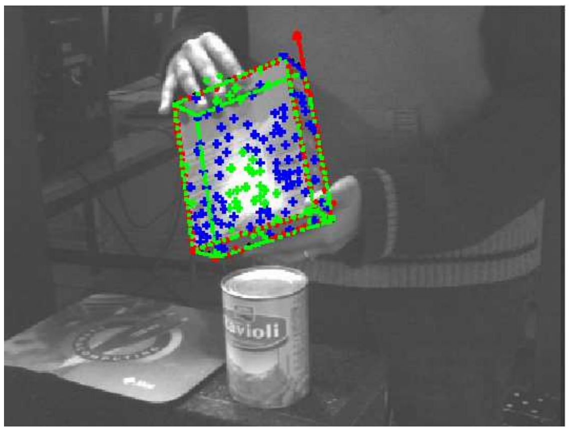
One can see in Figure 27 that the hybrid tracker enables a proper robot behavior even though the robot motion is quite fast at the beginning, and leads to an accurate positioning (see Figure 28). Whether there are occlusions or not, the error in the positioning is below 1 cm for the position parameters and 1degree for the orientation ones.

## 6 Conclusion

From two classical model-based trackers, a new hybrid one has been built, exploiting both edge extraction and texture information to obtain a more robust and accurate pose computation. The integration of the texture-based camera motion estimation in the edge-based camera pose estimation process enables a robust







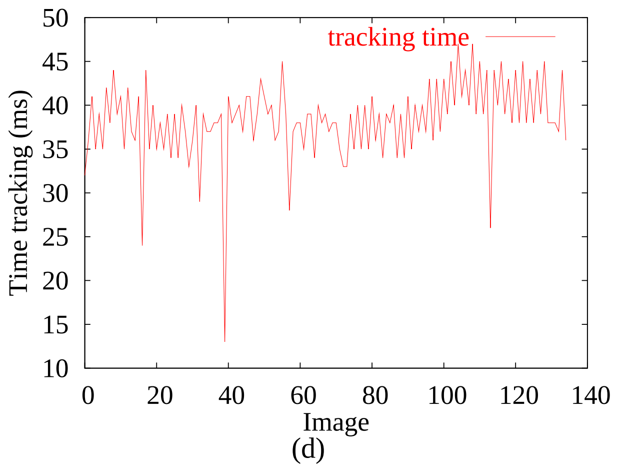


Figure 19: Rice box sequence. (a) camera pose parameters, (b) evolution of the number of grey level samples per face used in the control. The hybrid approach succeeds a tracking without jittering, which is illustrated by the smoothness of these curves. (c) example of specularity. The outliers are displayed in green and the inliers in blue for the grey level samples or red for the edge locations. (d) evolution of the time tracking

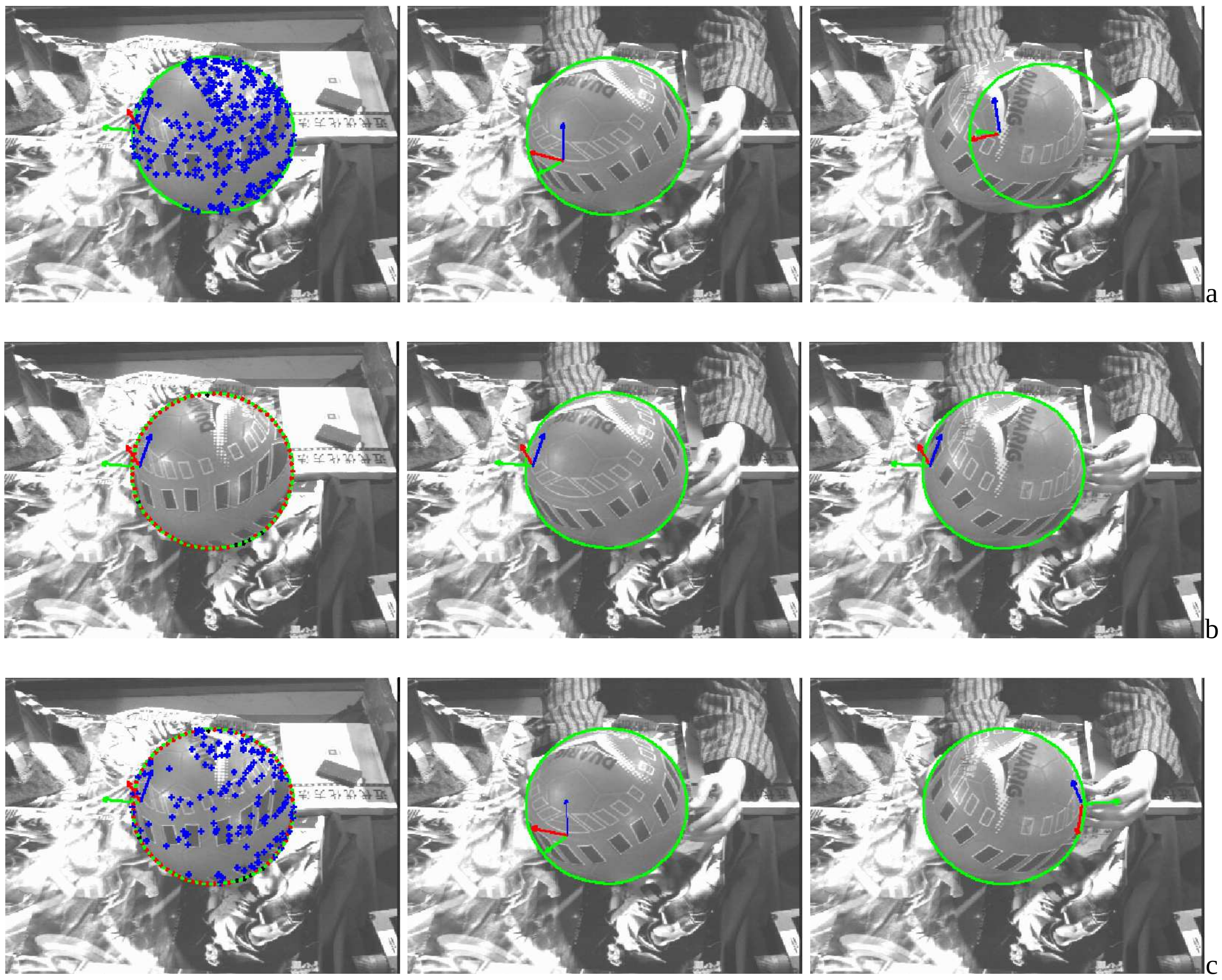
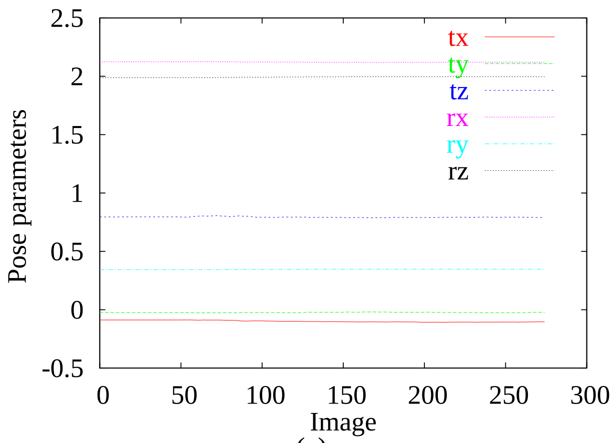
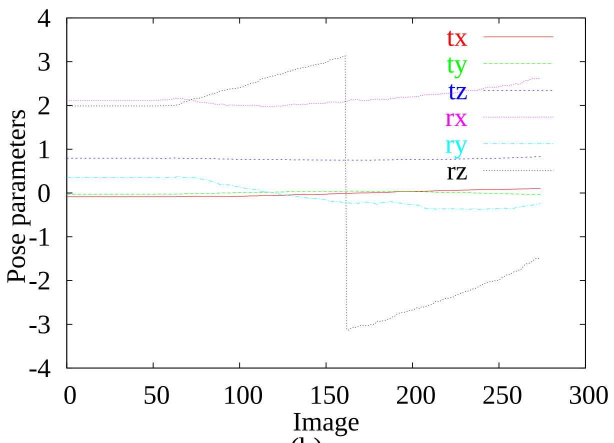


Figure 20: Ball sequence. Images for (a): the texture-based tracker that succeeds to estimate correctly the whole pose parameters for a while, (b): the edge-based tracker: the reference frame remains still in the image since the ball rotation is not observable using only the edge information, (c): the hybrid one. Only the hybrid tracker succeeds to track correctly the object all along the sequence, despite the specularities and the misleading environment.

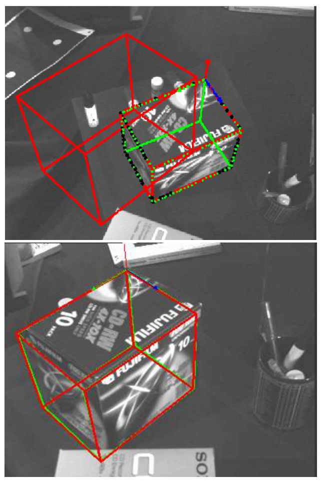


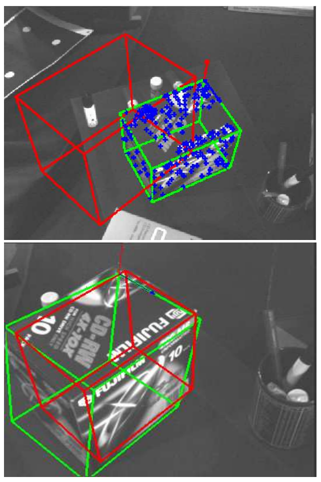
(a)



(b)

Figure 21: Ball sequence. Camera pose parameters for (a) edge-based tracker, (b) hybrid tracker. As the ball is rotated, the edge-based tracker detects no motion. The hybrid tracker can estimate fully this motion thanks to the texture information.


(a)


(b)

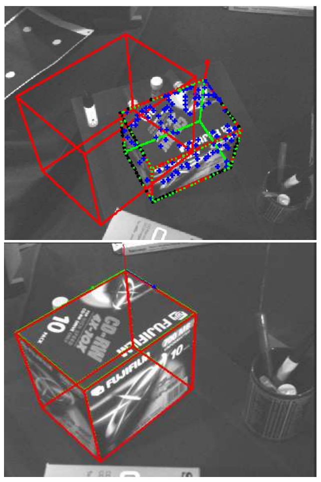

(c)

Figure 22: First 2 1/2 D visual servoing experiment, initial and final images for (a) the edge-based tracker, (b) the texture-based one, (c) the hybrid one. The desired (resp current) position of the object in the image is given by the red (resp green) drawing. Only the hybrid tracker succeeds to track the object and achieve a accurate positioning since the edge-based one drifts a little.

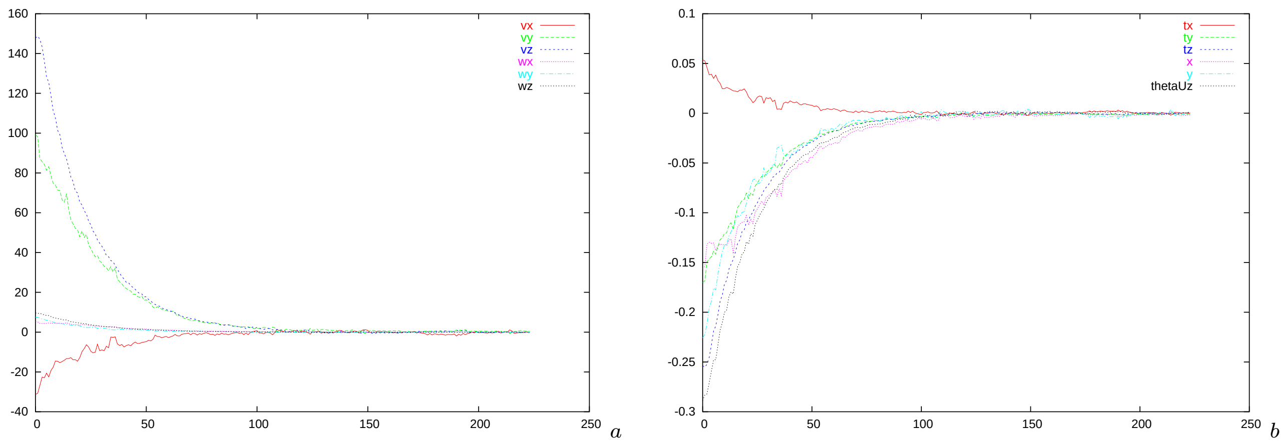


Figure 23: First 2 1/2 D visual servoing experiment using the hybrid algorithm. (a) Evolution of the camera velocity (mm/s and deg/s) (b) Evolution of the error.

| Axes | $t_x$ | $t_y$ | $t_z$ | $r_x$ | $r_y$ | $r_z$ |
|---|---|---|---|---|---|---|
| Desired pose | 56.1 | 50.8 | 11.0 | 10.7 | 42.9 | 0.0 |
| Final pose | 55.4 | 50.6 | 10.8 | 10.2 | 42.8 | 0.9 |
| Motion from this pose to the desired one | 0.1 | 0.8 | 0.5 | -0.1 | 0.2 | -0.6 |

Figure 24: First 2 1/2 D visual servoing experiment using the hybrid algorithm. Desired camera pose and the obtained one. $t_x$, $t_y$ and $t_z$ are the position parameters in cm and $r_x$, $r_y$ and $r_z$ are the orientation parameters in degrees.

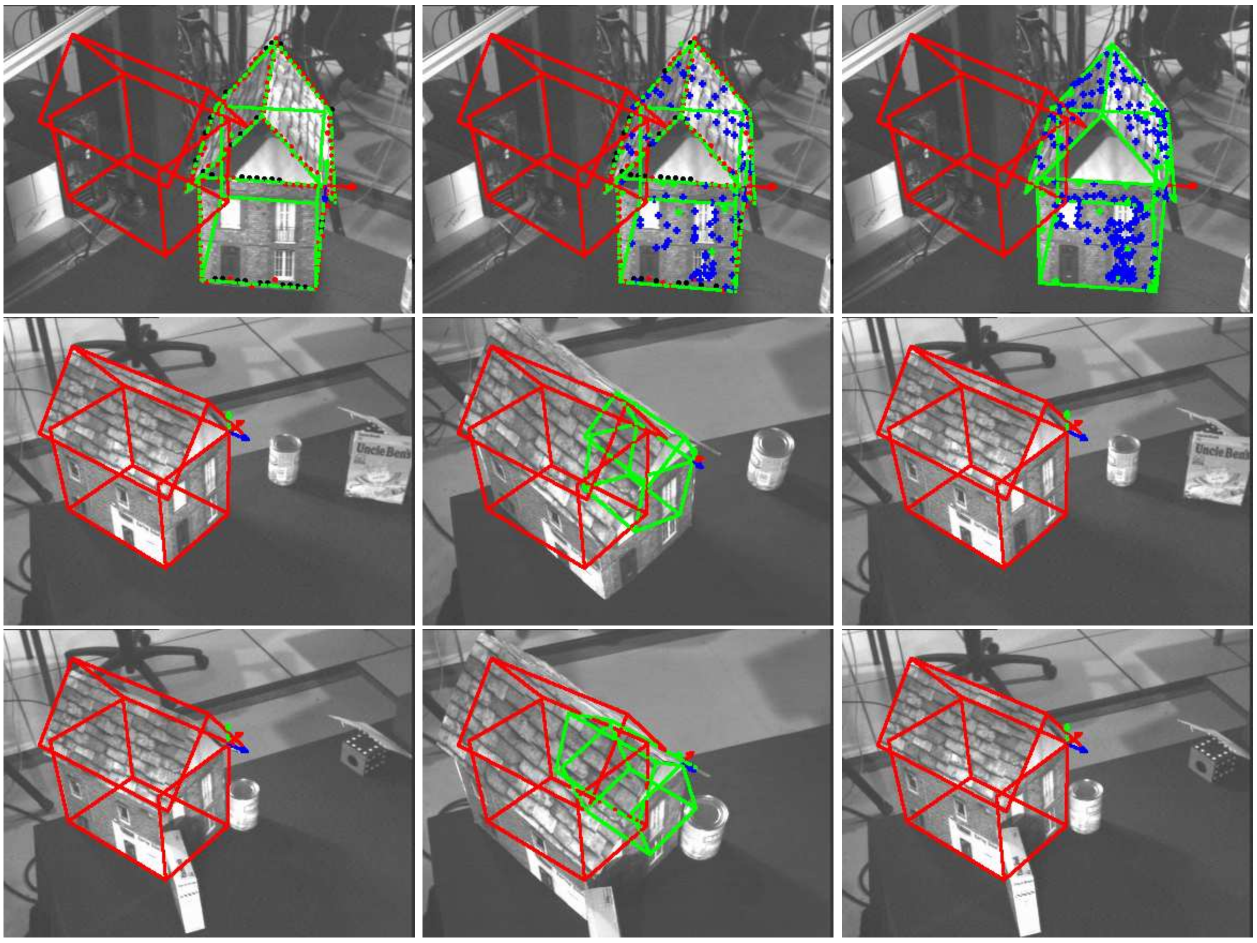

(a) (b) (c)

Figure 25: Second 2 1/2 D visual servoing experiment, Initial (first row) and final images (second row without occlusion, third one with occlusions) for (a) the edge-based tracker, (b) the texture-based one, (c) the hybrid one. The desired (resp current) position of the object in the image is given by the red (resp green) drawing. The hybrid tracker and the edge-based tracker succeed to track the object and achieve accurate positioning even if the object is not occluded. However when occlusions occurs, only the hybrid tracker enables to achieve the task with a good accuracy.

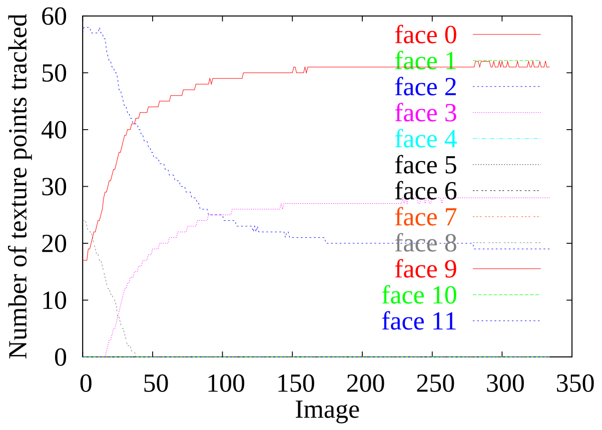


Figure 26: Second 2 1/2 D visual servoing experiment using the hybrid algorithm. Evolution of the visibility of the faces of the object. Appearance and disappearance of the faces do not disturb the tracker.

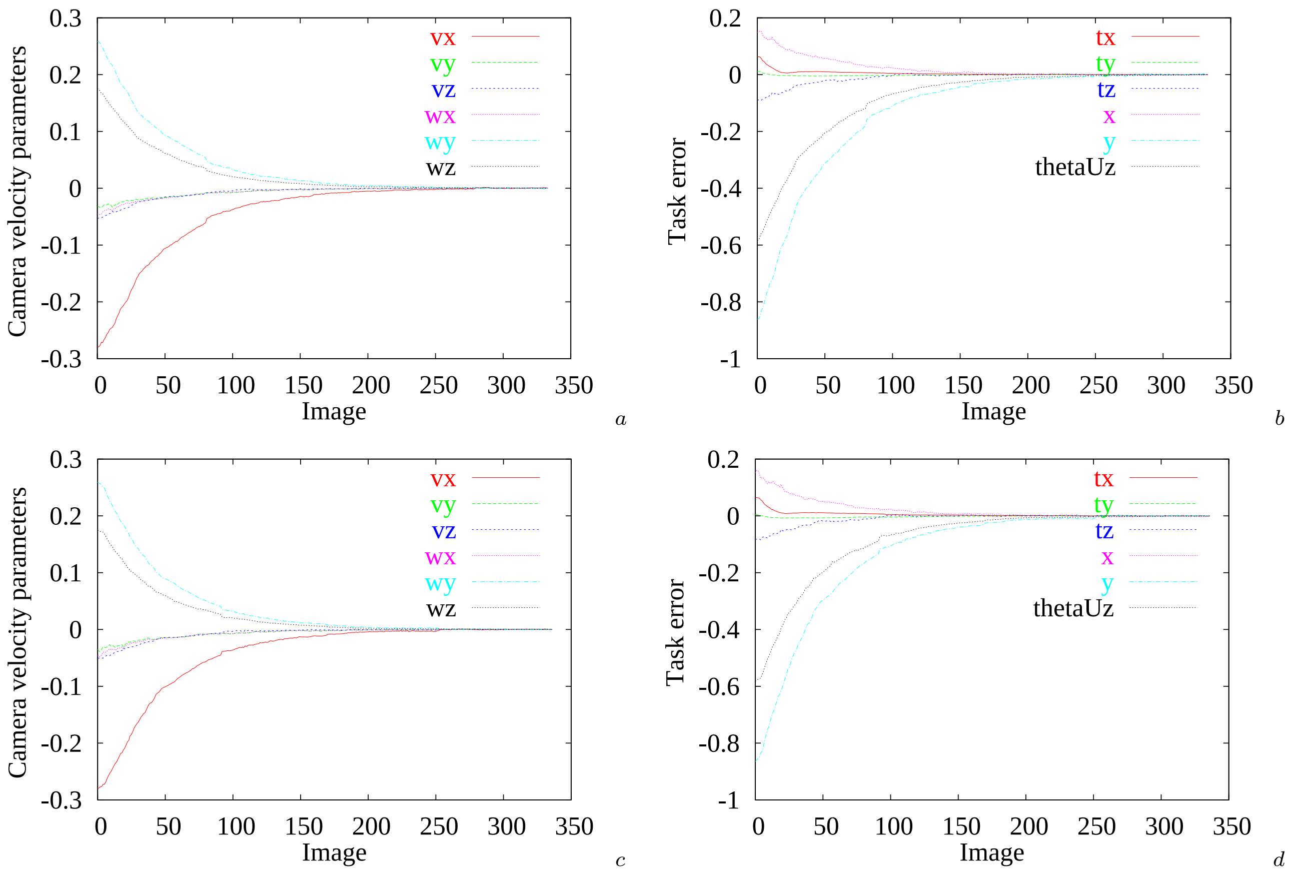


Figure 27: Second 2 1/2 D visual servoing experiment using the hybrid algorithm. (a) evolution of the camera velocity (m/s and rad/s) when no occlusion occurs, (b) evolution of the error when no occlusion occurs,(c) evolution of the camera velocity (m/s and rad/s) when occlusions occur, (d) evolution of the error when occlusions occur. The hybrid tracker enables a smooth robot motion and a good positioning, whether the object is occluded or not.

| Axes | $t_x$ | $t_y$ | $t_z$ | $r_x$ | $r_y$ | $r_z$ |
|---|---|---|---|---|---|---|
| Desired pose | -41.7 | 53.0 | -35.4 | 21.3 | 50.7 | 0.0 |
| Final pose with edge-based tracker without occlusions | -42.4 | 53.2 | -36.2 | 21.5 | 50.2 | 0.2 |
| Motion from this pose to the desired one | 1.1 | 0.1 | 0.7 | -0.3 | 0.5 | 0 |
| Final pose with hybrid tracker without occlusions | -41.7 | 53.5 | -35.5 | 21.3 | 50.5 | 0.1 |
| Motion from this pose to the desired one | 0.1 | -0.4 | 0 | 0 | 0.2 | 0 |
| Final pose with edge-based tracker with occlusions | -41.7 | 53.7 | -43.2 | 24.1 | 48.6 | -1.6 |
| Motion from this pose to the desired one | 2.8 | 1.8 | 8.1 | -1.6 | 1.5 | 1.8 |
| Final pose with hybrid tracker with occlusions | -41.3 | 53.3 | -35.4 | 21.3 | 50.8 | 0 |
| Motion from this pose to the desired one | -0.5 | -0.3 | 0 | 0 | -0.1 | 0 |

Figure 28: Second 2 1/2 D visual servoing experiment using the hybrid algorithm. Desired camera pose and the obtained one. $t_x$, $t_y$ and $t_z$ are the position parameters in cm and $r_x$, $r_y$ and $r_z$ are the orientation parameters in degrees. If no occlusion occurs, the edge-based and the hybrid trackers both enable an accurate positioning. However, only the hybrid one remains effective when occlusions occur.

and real-time tracking. M-estimators are added in the tracking process to enforce the robustness of the algorithm to occlusions, shadows, specularities and misleading backgrounds. The effectiveness of the proposed approach has been tested on various image sequences and within visual servoing positioning tasks.

We are now interested in extending this spatio-temporal tracking to texture lying on other non-planar structures to track a wider range of objects. As any improvement in the treatment of a kind of feature in the tracking process leads also to a better hybrid tracker, we also study a model of the textured plane to enforce the robustness to illumination changes.

## 7 Acknowledgement

The authors would like to thank Andrew Comport and Nicolas Mansard for their comments on this work.

## A Edge extraction

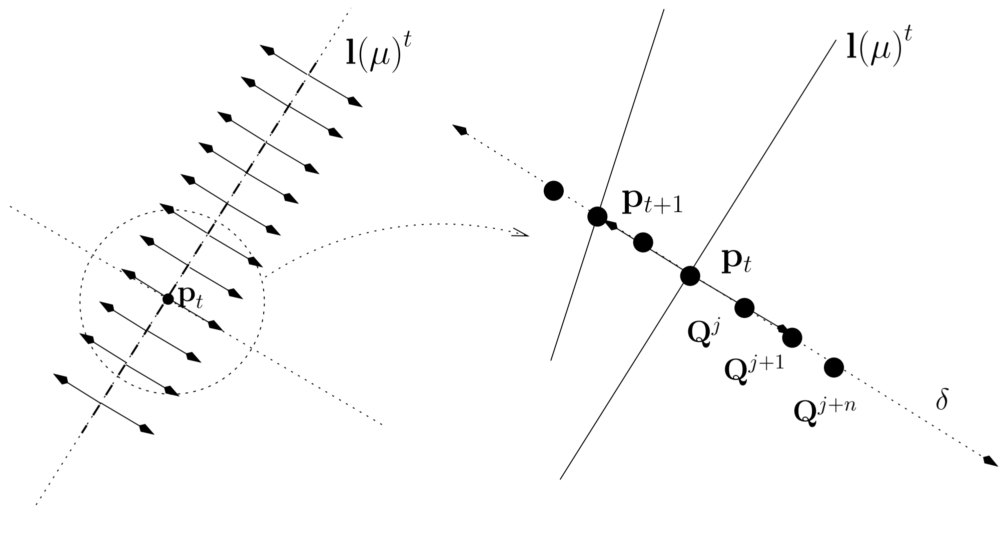
When dealing with low-level image processing, the contours are sampled at a regular distance. At these sample points a 1 dimensional search is performed to the normal of the contour for corresponding edges. An *oriented* gradient mask [6] is used to detect the presence of a similar contour. One of the advantages of this method is that it only searches for edges which are aligned in the same direction as the parent contour. An array of 180 masks is generated off-line which is indexed according to the contour angle. This is therefore implemented with convolution efficiency, and leads to real-time performance.

More precisely, the process [6] consists of searching for the corresponding point $\mathbf{p}_{t+1}$ in image $\mathbf{I}^{t+1}$ for each point $\mathbf{p}_t$ (see Figure 29). A 1D search interval $\{\mathbf{Q}_j, j \in [-J, J]\}$ is determined in the direction $\delta$ of the normal to the contour.For each point $P_i^t$ in the list $L^t$, and for every entire position $Q_j$, we compute a criterion corresponding to the square root of a log-likelihood ratio $\zeta^j$ [6]. The latter is nothing but the absolute sum of the convolution values, computed at $p^t$ and $Q_j$ respectively in images $I^t$ and $I^{t+1}$, using a pre-determined mask $M_\delta$ function of the orientation of the contour. Then the new position $\mathbf{p}_{t+1}$ is given by:

$$Q_j^* = \arg \max_{j \in [-J,J]} \zeta_j \text{ with } \zeta_j = \mid I^t_{\nu(p^t)} * M_\delta + I^{t+1}_{\nu(Q_j)} * M_\delta \mid \tag{47}$$

$\nu(.)$ is the neighborhood of the considered pixel. In this paper the neighborhood is limited to a $7 \times 7$ pixel mask. It should be noted that there is a trade-off to be made between real-time performance and mask stability. Likewise there is a trade-off to be made between the search distance, real-time performance while considering the maximum inter-frame movement of the object.

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



(a) (b)

| | | |
|---|---|---|
| 100 | 100 | 100 |
| 0 | 0 | 0 |
| −100 | −100 | −100 |

(c)

| | | |
|---|---|---|
| 0 | −100 | −100 |
| 100 | 0 | −100 |
| 100 | 100 | 0 |

(d)

Figure 29: Determining points position in the next image using the oriented gradient algorithm: (a) calculating the normal at sample points, (b) sampling along the normal and searching new similar contour (c-d) 2 out of 180 3x3 predetermined masks $\mathbf{M}_\delta$ (in practice 7x7 masks are used) (c) $180^o$ (d) $45^o$.

## B Case of line tracking for a homography estimation

$\mathbf{J}_{a_{\mu_t}}$, $\mathbf{J}_{b_{\mu_t}}$ and $\mathbf{J}_{c_{\mu_t}}$ are the respective Jacobian matrices of the lines coordinates $a_{\mu_t}$, $b_{\mu_t}$ and $c_{\mu_t}$. With the notations introduced in sections 2 and 3, they are such as :

$$\begin{aligned}
\mathbf{J}_{a_{\mu_t}} &= -\frac{1}{d}\begin{pmatrix} c_{a_1}u_a & c_{a_2}u_a & c_{a_3}u_a & c_{a_1}u_b \\ c_{a_2}u_b & c_{a_3}u_b & c_{a_1}u_c & c_{a_2}u_v & c_{a_2}u_v \end{pmatrix} \\
\mathbf{J}_{b_{\mu_t}} &= -\frac{1}{d}\begin{pmatrix} c_{b_1}u_a & c_{b_2}u_a & c_{b_3}u_a & c_{b_1}u_b \\ c_{b_2}u_b & c_{b_3}u_b & c_{b_1}u_c & c_{b_2}u_v & c_{b_2}u_v \end{pmatrix} \\
\mathbf{J}_{c_{\mu_t}} &= -\frac{1}{d}\begin{pmatrix} c_{c_1}u_a & c_{c_2}u_a & c_{c_3}u_a & c_{c_1}u_b \\ c_{c_2}u_b & c_{c_3}u_b & c_{c_1}u_c & c_{c_2}u_v & c_{c_2}u_v \end{pmatrix}
\end{aligned} \tag{48}$$

with :

$$\begin{aligned}
d &= det(\mathbf{H}) \\
\mathbf{c}_a &= \begin{pmatrix} c_{a_1} & c_{a_2} & c_{a_3} \end{pmatrix}^\top \\
&\quad \begin{pmatrix} (\mu_4\mu_8 - \mu_5\mu_7) & (\mu_5\mu_6 - \mu_3\mu_8) & (\mu_3\mu_7 - \mu_4\mu_6) \end{pmatrix}^\top \\
\mathbf{c}_b &= \begin{pmatrix} c_{b_1} & c_{b_2} & c_{b_3} \end{pmatrix}^\top \\
&\quad \begin{pmatrix} (\mu_2\mu_7 - \mu_1\mu_8) & (\mu_0\mu_8 - \mu_2\mu_6) & (\mu_1\mu_6 - \mu_0\mu_7) \end{pmatrix}^\top \\
\mathbf{c}_c &= \begin{pmatrix} c_{c_1} & c_{c_2} & c_{c_3} \end{pmatrix}^\top \\
&\quad \begin{pmatrix} (\mu_1\mu_5 - \mu_2\mu_4) & (\mu_2\mu_3 - \mu_0\mu_5) & (\mu_0\mu_4 - \mu_1\mu_3) \end{pmatrix}^\top \\
\mathbf{c} &= \begin{pmatrix} a_{\hat{\mu}_{t-1}} & b_{\hat{\mu}_{t-1}} & c_{\hat{\mu}_{t-1}} \end{pmatrix} \\
u_a &= \mathbf{c}.\mathbf{c}_a \\
u_b &= \mathbf{c}.\mathbf{c}_b \\
u_c &= \mathbf{c}.\mathbf{c}_c
\end{aligned} \tag{49}$$

Using these notations, the update of $\mathcal{C}_{\mu_t}$ along the sequence, *i.e.* of the coefficients $a_{\mu_t}$, $b_{\mu_t}$ and $c_{\mu_t}$, is given by :

$$a_{\mu_t} = \frac{u_a}{d}, \quad b_{\mu_t} = \frac{u_b}{d}, \quad c_{\mu_t} = \frac{u_c}{d}$$

Let note that $d \neq 0$ since it is the determinant of a homography.

## C Notations

Table 1 gives an overview of the different notation used in the paper.

| Notation | | Signification |
|---|---|---|
| $\mu_t$ | : | current 2D (homography) or 3D (pose) transformation for the image $\mathbf{I}_t$ |
| $\mathbf{s}$ | : | image features exploited to estimate the transformation |
| $\mathbf{s}^*$ | : | observed values of the image features in image $\mathbf{I}_t$ |
| $\mathbf{s}_{\mu_t}$ | : | current values of the image features in image $\mathbf{I}_t$ |
| $\mathbf{J}_x$ | : | Jacobian of a feature $x$ |
| $\mathbf{p}$ | : | image point |
| $\mathbf{p}_0$ | : | image point extracted in the reference image |
| $\mathbf{p}_{\mu_t}$ | : | projection of an image point according to $\mu_t$ |
| $\mathbf{p}_t$ | : | image point extracted in the current image $\mathbf{I}_t$ |
| $\partial_\perp(\mathbf{p}, \mathcal{C})$ | : | distance between a point $\mathbf{p}$ and a geometrical feature $\mathcal{C}$ |
| $\mathcal{C}_{\mu_t}$ | : | geometrical feature representing the contour according to $\mu_t$ |
| $I_0(\mathbf{p}_0^i)$ | : | grey level at location $\mathbf{p}_0^i$ in image $\mathbf{I}_0$ |
| $I_t(\mathbf{p}_{\mu_t}^i)$ | : | grey level at location $\mathbf{p}_{\mu_t}^i$ in image $\mathbf{I}_t$ |
| $\mathbf{K}$ | : | projective matrix obtained from the intrinsic camera parameters |
| ${}^{c_t}\mathbf{M}_w$ | : | camera pose matrix associated to image $\mathbf{I}_t$ |
| ${}^{c_t}\mathbf{R}_{c_0}$ | : | camera rotation matrix between the frames respectively associated to $\mathbf{I}_0$ and $\mathbf{I}_t$ |
| ${}^{c_t}\mathbf{t}_{c_0}$ | : | camera translation vector between the frames respectively associated to $\mathbf{I}_0$ and $\mathbf{I}_t$ |
| ${}^{c_t}\mathbf{H}_{c_0}$ | : | homography associated to a plane between the frames respectively associated to $\mathbf{I}_0$ and $\mathbf{I}_t$ |
| $\mathbf{n}_0$, $d_0$ | : | normal and distance to the origin of a plane expressed in the camera reference frame |

Table 1: Main notations used in the paper.

## References

[1] B. Bascle, P. Bouthemy, N. Deriche, and F. Meyer. Tracking complex primitives in an image sequence. In *Int. Conf. on Pattern Recognition, ICPR'94*, pages 426–431, Jerusalem, October 1994.

[2] S. Benhimane and E. Malis. Homography-based 2d visual tracking and servoing. *Int. Journal of Computer Vision*, 2007. Special IJCV/IJRR issue on vision for robots.

[3] M.-O. Berger. How to track efficiently piecewise curved contours with a view to reconstructing 3D objects. In *Int. Conf on Pattern Recognition, ICPR'94*, pages 32–36, Jerusalem, October 1994.

[4] A. Blake and M. Isard. *Active Contours*. Springer Verlag, April 1998.

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

[6] P. Bouthemy. A maximum likelihood framework for determining moving edges. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5):499–511, May 1989.

[7] T. Brox, B. Rosenhahn, D. Cremers, and H.-P. Seidel. High accuracy optical flow serves 3-D pose tracking: exploiting contour and flow based constraints. In A. Leonardis, H. Bischof, and A. Pinz, editors, *European Conf. on Computer Vision, ECCV'06*, volume 3952 of *LNCS*, pages 98–111, Graz, Austria, May 2006. Springer.

[8] J. Buenaposada and L. Baumela. Real-time tracking and estimation of plane pose. In *IAP Int. Conf. on Pattern Recognition, ICPR'02*, volume 2, pages 697–700, Québec, Canada, August 2002.

[9] F. Chaumette. Image moments: a general and useful set of features for visual servoing. *IEEE Trans. on Robotics*, 20(4):713–723, August 2004.

[10] N. Chiba and T. Kanade. A tracker for broken and closely-spaced lines. In *ISPRS Int. Society for Photogrammetry and Remote Sensing Conf.*, pages 676 – 683., Hakodate,Japan, 1998.

[11] A.I. Comport, E. Marchand, and F. Chaumette. Robust model-based tracking for robot vision. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'04*, volume 1, pages 692–697, Sendai, Japan, September 2004. (extended version version published in Advanced Robotics, 16(10):1097–1013, december 2005 (special issue on Selected paper from IROS'04).

[12] A.I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Trans. on Visualization and Computer Graphics*, 12(4):615–628, july 2006.

[13] N. Daucher, M. Dhome, J.T. Lapreste, and G. Rives. Modelled object pose estimation and tracking by monocular vision. In *British Machine Vision Conf., BMVC'93*, pages 249–258, Guildford, UK, September 1993.

[14] M. Dhome, M. Richetin, J.-T. Lapresté, and G. Rives. Determination of the attitude of 3D objects from a single perspective view. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989.

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

[16] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313–326, June 1992.

[17] A.W. Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(12-13):1145–1153, December 2003.

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

[19] M. Haag and H.H. Nagel. Combination of edge element and optical flow estimates for 3D-model-based vehicle tracking in traffic image sequences. *Int. Journal of Computer Vision*, 35(3):295–319, December 1999.

[20] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, October 1998.

[21] G. Hager and K. Toyama. The XVision system: A general-purpose substrate for portable real-time vision applications. *Computer Vision and Image Understanding*, 69(1):23–37, January 1998. Also Research Report Yale University.

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

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

[24] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, October 1996.

[25] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *ECCV'92*, pages 282–287, 1992.

[26] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *European Conf. on Computer Vision, ECCV'96, LNCS no. 1064, Springer-Verlag*, pages 343–356, Cambridge, UK, 1996.

[27] F. Jurie and M. Dhome. Read time 3D template matching. In *Int. Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 791–796, Hawai, December 2001.

[28] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):996–1000, July 2002.

[29] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *Int. Journal of Computer Vision*, 10(2):257–281, June 1993.

[30] D. Kragic and H. Christensen. Cue integration for visual servoing. *IEEE Trans. on Robotics and Automation*, 17(1):19–26, February 2001.

[31] V. Kyrki and D. Kragic. Integration of model-based and model-free cues for visual object tracking in 3d. In *IEEE Int. Conf. on Robotics and Automation, ICRA'05*, pages 1566–1572, Barcelona, Spain, April 2005.

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

[33] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. on Artificial Intelligence, IJCAI'81*, pages 674–679, 1981.

[34] Y. Ma, S. Soatto, J. Košecká, and S. Sastry. *An invitation to 3-D vision*. Springer, 2004.

[35] E. Marchand. Visp: A software environment for eye-in-hand visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'99*, volume 4, pages 3224–3229, Detroit, Michigan, Mai 1999.

[36] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2D-3D model-based approach. In *IEEE Int. Conf. on Computer Vision, ICCV'99*, volume 1, pages 262–268, Kerkira, Greece, September 1999.

[37] E. Marchand and F. Chaumette. Virtual visual servoing: a framework for real-time augmented reality. In G. Drettakis and H.-P. Seidel, editors, *EUROGRAPHICS'02 Conf. Proceeding*, volume 21(3) of *Computer Graphics Forum*, pages 289–298, Saarebrücken, Germany, September 2002.

[38] F. Martin and R. Horaud. Multiple camera tracking of rigid objects. *Int. Journal of Robotics Research*, 21(2):97–113, February 2002. (INRIA RR-4268, september 2001).

[39] L. Masson, F. Jurie, and M. Dhome. Contour/texture approach for visual tracking. In *13th Scandinavian Conf. on Image Analysis, SCIA 2003*, volume 2749 of *Lecture Notes in Computer Science*, pages 661–668. Springer, 2003.

[40] P. Meer, C. V. Stewart, and D. E. Tyler. Robust computer vision: An interdisciplinary challenge. *Computer Vision and Image Understanding: CVIU*, 78(1):1–7, 2000.

[41] J.-M. Odobez and P. Bouthemy. Robust multiresolution estimation of parametric motion models. *Journal of Visual Communication and Image Representation*, 6(4):348–365, December 1995.

[42] L. Piegl and W. Tiller. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., 1997.

[43] C. Rasmussen and G. Hager. Joint probabilistic techniques for tracking multi-part objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(6):560–576, 2001.

[44] J. Shi and C. Tomasi. Good features to track. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'94*, pages 593–600, Seattle, Washington, June 1994.

[45] G. Taylor and L. Kleeman. Fusion of multimodal visual cues for model-based object tracking. In *Australasian Conference on Robotics and Automation (ACRA2003)*, Brisbane,Australia, December 2003.

[46] M. Tonko and H.H. Nagel. Model-based stereo-tracking of non-polyhedral objects for automatic disassembly experiments. *Int. Journal of Computer Vision*, 37(1):99–118, June 2000.

[47] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *ACM/IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR'2004*, volume 2, pages 48–57, Arlington, Va, November 2004.

[48] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(10):1385–1391, October 2004.

[49] M. Vincze. Robust tracking of ellipses at frame rate. *Pattern Recognition*, 34(2):487 – 498, February 2001.

## Contents