Real-time 3D Model-Based Tracking: Combining Edge and Texture Information

Muriel Pressigout
IRISA-Université de Rennes 1
Campus de Beaulieu, 35042 Rennes Cedex, France
Email: Muriel.Pressigout@irisa.fr

Éric Marchand
IRISA-INRIA Rennes
Campus de Beaulieu, 35042 Rennes Cedex, France
Email: Eric.Marchand@irisa.fr

Abstract— **This paper proposes a real-time, robust and efficient 3D model-based tracking algorithm. A non linear minimization approach is used to register 2D and 3D cues for monocular 3D tracking. The integration of texture information in a more classical non-linear edge-based pose computation highly increases the reliability of more conventional edge-based 3D tracker. Robustness is enforced by integrating a M-estimator into the minimization process via an iteratively re-weighted least squares implementation. 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 large motions and textured environments.**

I. INTRODUCTION

This paper addresses the problem of robust real-time model-based tracking of 3D objects by integrating texture information in an edge-based process. This fundamental vision problem has applications in many domains such as, but not restricted to, robotics, industrial applications, medical imaging, augmented reality,. . . Most of the available tracking techniques can be divided into two main classes: feature-based and model-based. The former approach focuses on tracking 2D features such as geometrical primitives (points, segments, circles,. . .), object contours [13], regions of interest [10]. . . The latter explicitly uses a model of the tracked objects. This can be a CAD model [2], [3], [5], [6], [16], [18] or a 2D template of the object. This second class of methods usually provides a more robust solution. Indeed, the main advantage of the model-based methods is that the knowledge about the scene (the implicit 3D information) allows improvement of robustness and performance.

Classically, the problem is solved using registration techniques that allow alignment of 2D image data and a 3D model. Relying only on edge information provides good results when tracking sharp edges even if there are illumination changes. However, it can lead to jittering and even to erroneous pose estimation if the environment or the object is highly textured.

Texture information is widely used to track an object in an image sequence. Contrarily to edge-based trackers, it is well adapted to textured objects and does usually not 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 [25] or region of interest tracking [9], [15], [1]. Points or regions of interest can also be used within a 3D model-based tracking as reported in [14], [27]. Nevertheless these approach may become inappropriate on poorly textured objects. Furthermore they usually lack of precision if scale changes.

As one can note, model-based trackers can be mainly divided in two groups, the edge-based ones and the textured-based one. Both have complementary advantages and drawbacks. The idea is then to integrate both approaches in the same process.

The most classical approached to consider 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) [17], [26]. Let note that some methods rely on a particle filtering as [13] or PMHT [23]. Although these approaches are appealing and widely used, there exists other approaches that does not rely on probabilistic framework and though the objectives are similar, the theoretical aspect are very different and can hardly be compared. In [27] the proposed model-based approach considers both 2D-3D matching against a keyframe as in a classical model-based approach but considering multiple hypothesises for the edge tracking and 2D-2D temporal matching. The work of [21] extend the tracker of [14] to integrates contour information. Motion and edges may be used together to improve tracking results as in [8], [20]. The texture information has also been exploited in [24] to find the projected contour of a 3D object.

The framework presented in this paper also fuses a classical model-based approach based on the edge extraction [4] and a temporal matching relying on the texture analysis [9] in a similar manner as done in [22] for 2D tracking. Indeed, estimating both pose and camera displacement introduces an implicit spatio-temporal constraint the simple model-based tracker lacks of. The merging is however handled in a different way than in [27] or [21] and does not require points of interest extraction in each image. In this paper, pose and camera displacement computation is formulated in terms of a full scale non-linear optimization. This general framework is used to create an image feature based system which is capable of treating complex scenes in *real-time*. To improve robustness, a robust estimation based on M-estimator [12] is integrated in a robust control law.

In the remainder of this paper, Section II presents the principle of the approach and the Section III how to perform a camera pose or displacement estimation. The details of the integration of the camera displacement estimation in the pose computation process are given in Section IV. In order to validate this approach the tracker is tested on several realistic image sequences as well as input to a 2 1/2 D visual servoing experiments.

II. MODEL-BASED TRACKING BY A NON-LINEAR MINIMIZATION

The fundamental principle of the proposed approach is to integrate a camera displacement estimation based on the texture information in a more classical camera pose computation process that relies on edge-based features. This is achieved by minimizing a non-linear criterion. This Section is dedicated to the description of this general framework.

The approach consists of estimating the real camera pose $^{c}\mathbf{M}_o$ or displacement by minimizing the error Δ between the observed data $\mathbf{s}^*$ and the current value $\mathbf{s}$ of the same features computed using the model according to the current pose/displacement:

$$\Delta = \sum_{i=1}^{N} \rho\big(s_i(\mathbf{r}) - s_i^*\big)^2, \tag{1}$$

where $\rho(u)$ is a robust function [12] introduced in the objective function in order to reduce the sensitivity to outliers (M-estimation) and $\mathbf{r}$ is a vector-based representation of the pose $^{c}\mathbf{M}_o$.

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 0 is then defined as:

$$\mathbf{e} = \mathbf{D}(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*), \tag{2}$$

where $\mathbf{D}$ is a diagonal weighting matrix given by $\mathbf{D} = diag(w_1, \ldots, w_k)$. The weights w_i reflect the confidence in each feature and their computation is based on M-estimators and is described in [3], [5].

A virtual camera, defined by its position $\mathbf{r}$ in the object frame, can be virtually moved in order to minimize this error. At convergence the position of the virtual camera will be aligned with the real camera pose. This can be achieved by considering a simple control law given by:

$$\mathbf{v} = -\lambda(\widehat{\mathbf{D}\mathbf{L_s}})^+\mathbf{D}\big(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*\big), \tag{3}$$

where $\mathbf{v}$ is the velocity screw of the virtual camera and $\mathbf{L_s}$ is the interaction matrix or image Jacobian related to $\mathbf{s}$ and defined such as $\dot{\mathbf{s}} = \mathbf{L_s}\mathbf{v}$. Rodrigues' formula is then used to map the velocity vector $\mathbf{v}$ to its corresponding instantaneous displacement allowing the pose to be updated. To apply the update to the displacement between the object and camera, the exponential map is applied using homogeneous matrices $^{c}\mathbf{M}_o$ that describe the pose resulting in:

$$^{c}\mathbf{M}_o^{t+1} = {}^{c}\mathbf{M}_o^{t}\, e^{\mathbf{v}} \tag{4}$$

where t denotes the number of iteration of the minimization process.

Depending on the nature of the features, this approach can solve a pose computation problem or a camera displacement one. 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. The next Section is dedicated to the choice of visual features and to their role in the proposed method.

III. FEATURES CHOICE: HYBRID TRACKER

Any kind of features can be considered within the proposed control law as soon as it is possible to compute its corresponding interaction matrix $\mathbf{L_s}$. The combination of different features is 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:

$$\begin{pmatrix} \dot{\mathbf{s}}_1 \\ \vdots \\ \dot{\mathbf{s}}_n \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{s1} \\ \vdots \\ \mathbf{L}_{sn} \end{pmatrix} \mathbf{v} \tag{5}$$

The redundancy yields a more accurate result with the computation of the pseudo-inverse of $\mathbf{L_s}$ as given in (3).

The first Subsection is dedicated to the edge-based features used in a classical pose computation. The following one presents the texture-based features for the displacement estimation.

A. Edge-based features for a pose computation

The use of edge-based features enables to perform a pose computation as in a classical model-based tracker [3], [5], [18]. To illustrate the principle, consider the case of an object with various 3D features $^{o}\mathbf{P}$ (for instance, $^{o}\mathbf{P}$ are the 3D coordinates of these features in the object frame). The approach consists of estimating the real pose by minimizing the following error Δ between the observed data $\mathbf{s}^*$ (the position of a set of features in the image in the pose computation case) and the position $\mathbf{s}$ of the same features computed by forward-projection according to the current pose:

$$\Delta = \sum_{i=1}^{N} \rho\big(pr_\xi(\mathbf{r}, {}^{o}\mathbf{P}_i) - s_i^*\big)^2, \tag{6}$$

where $pr_\xi(\mathbf{r}, {}^{o}\mathbf{P})$ is the projection model according to the intrinsic parameters ξ and camera pose $\mathbf{r}$. At convergence, the virtual camera reaches the pose $\widehat{\mathbf{r}}$ which minimizes the error Δ ($\widehat{\mathbf{r}}$ will be the real camera's pose).

In the case of an edge-based model-based tracker, this framework allows to deal with different kinds of geometrical features and the derivation of the corresponding interaction matrices (or Jacobian) can be found in [3], [5], [18]. In this paper, we consider features corresponding to a distance between the forward-projection of the contours of a CAD model and local point features obtained from a 1D search along the normal to the contour. In this case the desired values of these distances are equal to zero. The assumption is made that the contours of the object in the image can be described as piecewise linear segments. All distances are then treated according to their corresponding segment. This process is described in [4].

The edge-based model-based tracker corresponds to a classical method for pose computation. It is fast, efficient, robust

to illumination changes. However, it is mainly a mono image process which leads to some issues. If the geometrical features can not be accurately extracted without any ambiguity, the tracker may lack of precision. As a consequence, it may be sensitive to the texture of the object or the background and prone to jittering.

B. Texture-based features for a camera displacement computation

The idea is then to introduce in the tracker a spatio-temporal constraint in order to correct the drawbacks of the edge-based tracker presented in the previous paragraph. This is achieved by a camera displacement estimation based on the image intensity matching between two images and incorporated in the same framework.

Assuming two images 1 and 2, our goal will be to estimate the camera pose $^2\mathbf{M}_o$ considering the pose in the previous image $^1\mathbf{M}_o$. Since $^2\mathbf{M}_o = {}^2\mathbf{M}_1\,{}^1\mathbf{M}_o$ updating the displacement $^2\mathbf{M}_1$ is equivalent to update the pose $^2\mathbf{M}_o$ as long as the pose is known in the first image.

Whereas for pose estimation the goal is to minimize the error between the features observed in the image and their forward projection onto the image plane, for camera displacement estimation the idea is to minimize the error between the grey level value at the location position $(\mathbf{p}_1)$ in the first image $\mathbf{I}_1$ and the one observed in the second image $\mathbf{I}_2$ at the location of the corresponding features transfered from $\mathbf{I}_1$ in the $\mathbf{I}_2$ through a 2D transformation $^2tr_1$ which relies on the camera displacement $^2\mathbf{M}_1$ and the geometrical multi-view constraints. For such features, equation (1) is then given by:

$$\Delta = \sum_{i=1}^{N} \rho\big(I_1(\mathbf{p}_{1_i}) - I_2({}^2tr_1(\mathbf{p}_{1_i}))\big)^2, \qquad (7)$$

where N is the number of considered pixels. At convergence, the virtual camera has realized the displacement $\widehat{^2\mathbf{M}_1}$ which minimizes this error. The details (the 2D transformation, the interaction matrix,...) are presented in the next Section.

Such a process enables the integration of the texture-based features for the camera displacement estimation in the control law used for the pose computation process following (5).

IV. INTEGRATING THE CAMERA DISPLACEMENT ESTIMATION IN THE POSE COMPUTATION

This Section presents the details of the integration of the displacement estimation in the pose computation process. The resulting algorithm is from now called the hybrid tracker. The points to be investigated are the 2D transformation and the interaction matrix associated to (7). The texture model and details about data processing are also given.

A. Point transfer

The geometry of a multi-view system (or of a moving camera) introduce very strong constraints in feature location in the different view. In the general case, the point transfer can be achieved considering the epipolar geometry and the essential or fundamental matrices (see, for example, [11]). In

this paper we restrict ourselves to the less general case where point transfer can be achieved using an homography. Since any kind of 3D motion must be considered, this means that the texture lies on a plane in the 3D space. In that case, a point $\mathbf{p}_1$ in image $\mathbf{I}_1$ expressed in homogeneous coordinates $\mathbf{p}_1 = ({}^1u, {}^1v, {}^1w)$, is transfered in image $\mathbf{I}_2$ as a point $\mathbf{p}_2$ by:

$$\mathbf{p}_2 = {}^2tr_1(\mathbf{p}_1) = \alpha\mathbf{K}^{-1}\,{}^2\mathbf{H}_1\mathbf{K}\mathbf{p}_1, \qquad (8)$$

where $\mathbf{K}$ is the intrinsic camera parameters matrix and $^2\mathbf{H}_1$ is an homography (defined up to scale factor α) that defines the transformation in meter coordinates between the images acquired by the camera at pose 1 and 2. Once a camera displacement is generated, the homography $^2\mathbf{H}_1$ is given by:

$$^2\mathbf{H}_1 = ({}^2\mathbf{R}_1 + \frac{^2\mathbf{t}_1}{^1d}{}^1\mathbf{n}^\top), \qquad (9)$$

where $^1\mathbf{n}$ and ^1d are the normal and distance to the origin of the reference plane expressed in camera 1 frame. $^2\mathbf{R}_1$ and $^2\mathbf{t}_1$ are respectively the rotation matrix and the translation vector between the two camera frames. We finally get $\mathbf{p}_2 = {}^2tr_1(\mathbf{p}_1) = ({}^2u, {}^2v, {}^2w)$ that is used for the next iteration of the minimization process.

B. Interaction matrix

The interaction matrix $\mathbf{L}_{I(\mathbf{p}_2)}$ is the interaction matrix that links the variation of the grey level value to the camera motion. It is given by [9]:

$$\mathbf{L}_{I(\mathbf{p}_2)} = \frac{\partial I(\mathbf{p}_2)}{\partial \mathbf{r}} = \nabla_{\mathbf{x}}\mathbf{I}_2^\top(\mathbf{p}_2)\frac{\partial \mathbf{p}_2}{\partial \mathbf{r}}, \qquad (10)$$

where $\nabla_{\mathbf{x}}\mathbf{I}_2(\mathbf{y})$ is the spatial image gradient of the image $\mathbf{I}_2$ at the location $\mathbf{y}$ and $\frac{\partial \mathbf{p}_2}{\partial \mathbf{r}} = \mathbf{L}_{\mathbf{p}_2}$ is the interaction matrix of an image point expressed in pixel coordinates. $\mathbf{L}_{\mathbf{p}_2}$ is given by :

$$\mathbf{L}_{\mathbf{p}_2} = \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \cdot$$
$$\begin{pmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & (1+y^2) & -xy & -x \end{pmatrix} \qquad (11)$$

f_x and f_y are the focal ratio of the camera and (x, y) denotes the meter coordinates of the image point $\mathbf{p}_2$ whose pixel location in the image is given by $(\frac{^2u}{^2w}, \frac{^2v}{^2w})$. The depth information Z is computed at each iteration: $1/Z = \frac{^1d - {}^2\mathbf{t}_1^\top\,{}^1\mathbf{n}}{(^2\mathbf{R}_1\,{}^1\mathbf{n})^\top [x,y,1]^\top}$.

C. The object model

The displacement estimation has been presented for two images $\mathbf{I}_1$ and $\mathbf{I}_2$. In practice, $\mathbf{I}_2$ is the current image for which the camera pose has to be estimated and $\mathbf{I}_1$ is a reference image of the tracked plane. There is a reference image for each plane π_i with texture to track on the object. 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 (9) and the depth computation.

D. Outlier detection

Since grey level is sampled on Harris points, 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, it is classical (as in robust motion estimation algorithm) to perform a normalization of the vector Δ which is used to compute the weight w_i of the matrix $\mathbf{D}$. This normalization is given by:

$$\Delta' = \left(\dots, \frac{I_1(\mathbf{p}_{1_i}) - I_2(^2tr_1(\mathbf{p}_{1_i}))}{\| \nabla I_1(\mathbf{p}_{1_i}) \|}, \dots \right) \quad (12)$$

Furthermore, the global illumination in the reference images may be different when the tracking is performed. To enforce the M-estimation process, the grey level mean of a reference image is adapted when the associated plane becomes visible.

E. Merging edge-based and texture-based information

As already said, any kind of features can be considered within the proposed framework using (5). If both edge-based and texture-based features and their associated interaction matrices are stacked as in our hybrid tracker, a scaling [7] must be performed to take into account the different unit of the considered cue (meter and pixel intensity). Indeed, the error associated with a texture point (grey level value) and the one associated with the edge locations (point-to-contour distance) are of a different order of magnitude. Therefore, the set of the errors associated with an edge-based feature (resp. a grey level value) is normalized such as these values belong to the interval $[-1; 1]$.

V. EXPERIMENTS AND RESULTS

This Section presents some tracking results where our hybrid tracker is compared to the edge-based one and the texture-based one. These two latter trackers use in the minimization process only the kind of feature associated with. The three first experiments are show object tracking in video sequences. In the fourth experiment tracking is performed during a positioning task by visual servoing. Thanks to the rejection of the outliers, the output of the hybrid tracker is at least as good as the output of the best single-cue tracker.

In all the reported experiments, the edge locations and texture points used in the minimization process are displayed in the first image (blue crosses for the grey level sample locations and red crosses for the edge locations). In the next images, only the forward-projection of the model for a given pose is displayed in green.

A. Rice box sequence

In the considered image sequence, tracking the rice box is a very complex task since the object achieve a complete rotation. If the tracking begins to drift, it may be difficult to rectify the error since the features to be tracked change. The object contours are permanently partially occluded by the hands or hardly visible: the edge-based tracker ends to lose the object (see Figure 4a). The texture-based tracker also fails to track the object quite quickly (see Figure 4b). However the hybrid tracker enables to track the object correctly (see Figure 4c).

The camera pose parameters evolution is shown in Figure 1a and the evolution of the number of grey level samples used in the control law per face in Figure 1b. 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 output can not be perfectly regular. Figure 1c shows an example of specularities the tracker has to deal with. The grey level samples in the concerned area are considered as outliers by the M-estimators. 240 points are tracked in each case, the hybrid tracker runs at an average frame rate of 25 Hz (see Figure 1d).
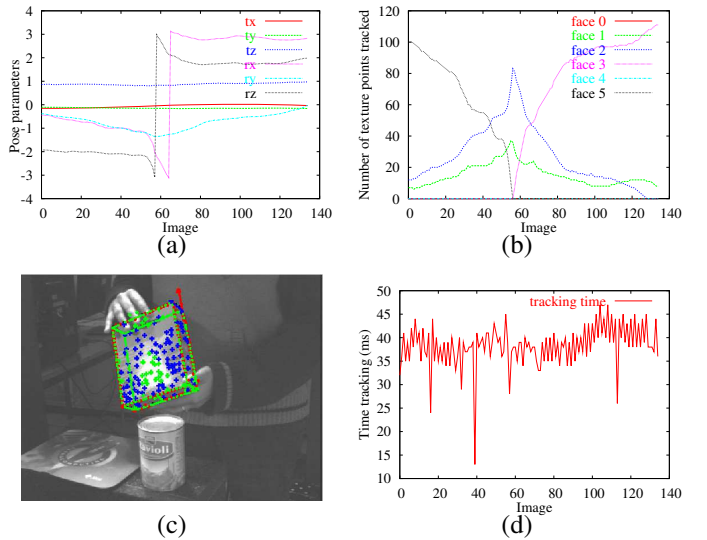

Fig. 1. Rice box sequence. (a) camera pose parameters, (b) evolution of the number of grey level samples per face used in the control. The hybrid approach succeeds a tracking without jittering, which is illustrated by the smoothness of these curves. (c): example of specularity. The outliers are displayed in green and the inliers in blue for the grey level samples or red for the edge locations. (d): evolution of the time tracking

B. Cylinder sequence

Tracking texture on plane does not restrict this approach to boxes tracking: the new object to track is a cylinder. This is also a quite difficult test since there are misleading edges for the edge-based tracker that fails quickly (see Figure 5a). Even if there is only one plane, the texture-based tracker is quite robust (see Figure 5b). However, as its accuracy depends on the pose computed for the reference image, it is less accurate than the hybrid one (see Figure 5c). In this latter case, the edge helps the tracker to better fit the object. Let us note that the texture is required in this experiment to estimate correctly the 6 parameters of the pose. Indeed, when using only the edge-based features in the tracking process of a cylinder, the rotation along the cylinder axis is not taken into account. Let us note that tracking is achieved at 30Hz.

C. Visual servoing experiment

The hybrid tracker has been used successfully in 2 1/2 D visual servoing experiments [19]. Figure 6 presents an example of such an application. A desired position of the object in the image is given and the camera mounted on a 6 d.o.f robot moves from an initial position to the desired one by

minimizing the error between the desired position of the object in the image (red box) and the current one (green box). The precision of the tracking is a key point of the success or failure of such a task. Only the hybrid tracker succeeds to track the object and achieve an accurate position
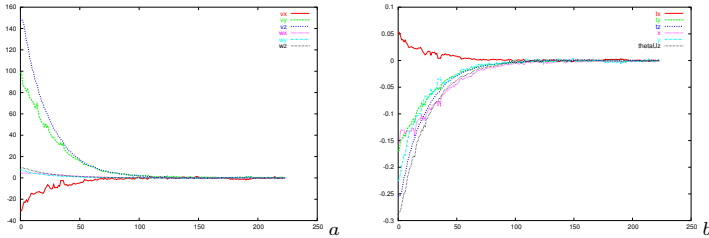

Fig. 2. Visual servoing experiment using the hybrid algorithm. (a) Evolution of the camera velocity (mm/s and deg/s) (b) Evolution of the error.

pose	t_x	t_y	t_z	r_x	r_y	r_z
desired	560.9	507.9	110.1	10.7	42.9	0.0
obtained	553.6	506.0	107.7	10.2	42.8	0.9

Fig. 3. 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 millimeters and r_x, r_y and r_z are the orientation parameters in degrees.

In Figure 2(a), the evolution of the camera velocity is given and Figure 2(b) shows the task error decreasing. This leads to a precise positioning: the desired pose and the obtained one are given in Figure 3. The error in the positioning is below 1 centimeter for the position parameters and 1 degree for the orientation ones.

VI. CONCLUSION AND PERSPECTIVES

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 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 sequence and within visual servoing positioning tasks.

We are now interested in extending this spatio-temporal tracking to texture lying on 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 multi-scale model of the textured plane to enforce the robustness to scale changes.

REFERENCES

[1] S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *IEEE/RSJ Int. Conf. on Intelligent Robots Systems*, Sendai, Japan, October 2004.

[2] K. Berk Yesin and B. Nelson. Robust cad model-based visual tracking for 3d microassembly using image space potentials. In *IEEE Int. Conf. on Robotics and Automation, ICRA'04*, pages 1868–1873, New Orleans, USA, April 2004.

[3] 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.

[4] 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*, 2006.

[5] 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.

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

[7] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, 1981.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

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

[16] H. Kollnig and H.-H. Nagel. 3D Pose Estimation by Directly Matching Polyhedral Models to Gray Value Gradients. *Int. Journal of Computer Vision*, 23(3):283–302, July 1997.

[17] 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.

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

[19] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 D visual servoing. *IEEE Trans. on Robotics and Automation*, 15(2):238–250, April 1999.

[20] 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.

[21] 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.

[22] M. Pressigout and E. Marchand. Real-time planar structure tracking for visual servoing: a contour and texture approach. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'05*, volume 2, pages 1701–1706, Edmonton, Canada, aug 2005.

[23] 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.

[24] A. Shahrokni, T. Drummond, and P. Fua. Texture boundary detection for real-time tracking. In *European Conf. on Computer Vision, ECCV'04, LNCS 3022*, volume 2, pages 566–577, Prague, Czech Republic, May 2004.

[25] 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.

[26] 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.

[27] 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.

Fig. 4. 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.
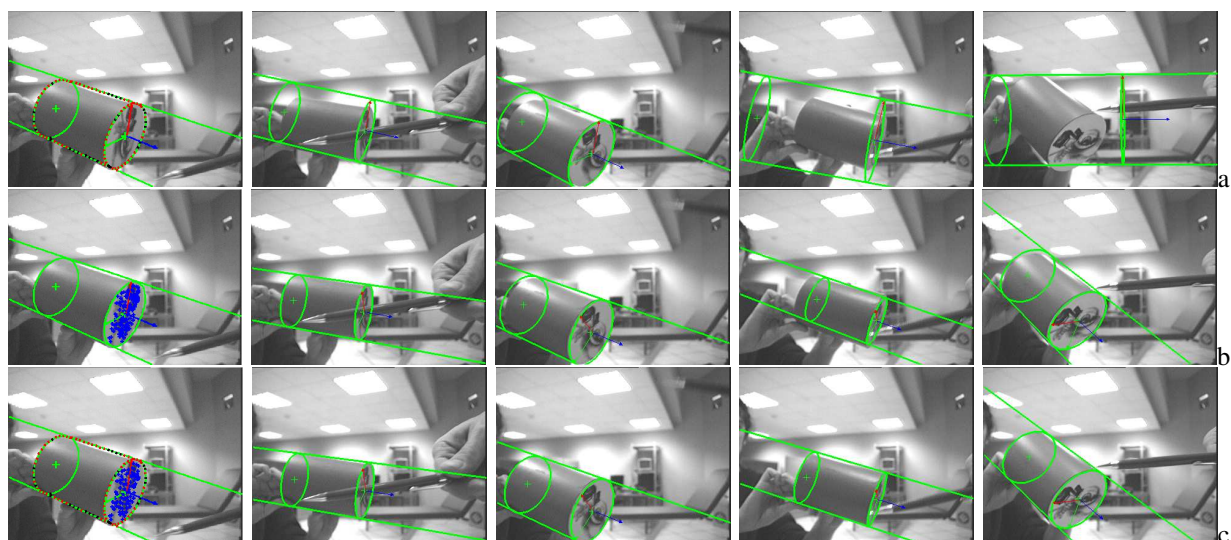

Fig. 5. Cylinder sequence. Images for (a): the edge-based tracker, (b): the texture-based one, (c): the hybrid one. The hybrid tracker is the better one, despite the specularities and the misleading environment. For the initial images, the grey level samples are represented by blue crosses and the edge location by red points if an edge is detected or a black one otherwise
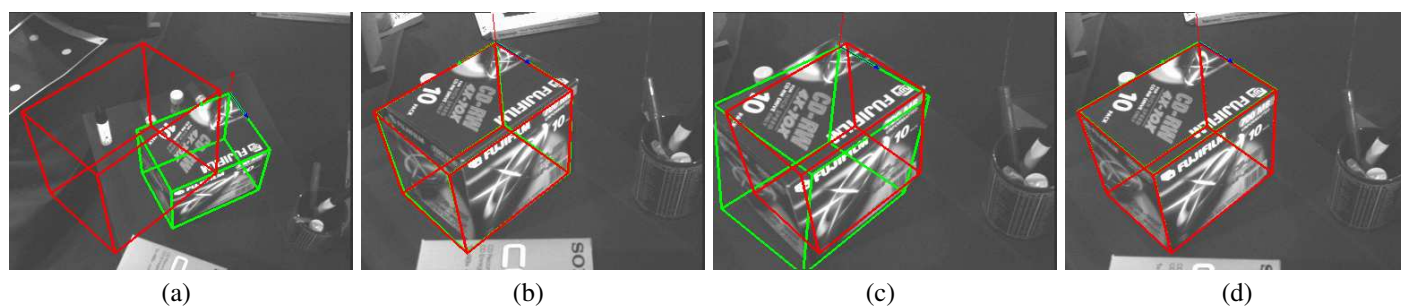

| (a) | (b) | (c) | (d) |

Fig. 6. Visual servoing experiment, (a) Initial image. Final images for (b): the edge-based tracker, (c): the texture-based one, (d): 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.