A robust model-based tracker combining geometrical and color edge information

Antoine Petit, Eric Marchand, Keyvan Kanani

Abstract— **This paper focuses on the issue of estimating the complete 3D pose of the camera with respect to a potentially textureless object, through model-based tracking. We propose to robustly combine complementary geometrical and color edge-based features in the minimization process, and to integrate a multiple-hypotheses framework in the geometrical edge-based registration phase. In order to deal with complex 3D models, our method takes advantage of GPU acceleration. Promising results, outperforming classical state-of-art approaches, have been obtained for space robotics applications on various real and synthetic image sequences and using satellite mock-ups as targets.**

I. INTRODUCTION

Determining the complete 3D pose of the camera with respect to the object is a key requirement in many robotic applications involving 3D objects, especially in the case of autonomous, vision-based and uncooperative space rendezvous with space targets or debris [3], [15]. Based on the knowledge of the 3D model of the target, common approaches address this problem by using either texture [2] or edge features [5], [6], [10], [15]. Edge features offer a good invariance to illumination changes or image noise, conditions which can be encountered in space environments and are particularly suitable for poorly textured objects such as space objects. For such class of approaches, the pose computation is achieved by minimizing the distance between the projected edges of the 3D model and the corresponding edge features in the image, using weighted numerical nonlinear optimization techniques like Newton-Raphson or Levenberg-Marquardt. But though they have proven their efficiency, this technique requires an image extraction process which can involve outliers and, contrary to feature points which can be specifically described, suffer from having similar appearances. It can result in ambiguities between different edges, leading to tracking failures, particularly in the case of complex objects like satellites or space debris. Thus we propose a method to improve the accuracy and the robustness of 3D model-based tracking, while preserving reasonable computational costs.

A. Related works

In the recent literature could be distinguished three different kinds of approaches tackling this problem:

- One solution is to combine the information provided by edges with information provided by other features, such as interest points [16], [17], [19], color [13], or by additional sensors [8].
- Some researches have focused on the low-level robustness. To reject outliers in the edge matching process, methods like RANSAC [2], [4] or the use of M-Estimators such as the Tukey estimator [5], [19] are common trends to make the algorithm robust to occlusions and illumination variations. Also, instead of handling a single hypothesis for a potential edge in the image, multiple hypotheses are extracted and registered in the pose estimation [18], [19].
- Other studies have considered Bayesian filters such as Kalman filter [21] and more recently particle filters [4], [9], [18]. For such methods, a set of hypotheses on the camera pose is propagated with respect to a dynamic model. The pose is then estimated by evaluating the likelihood of the hypotheses in the image. In [18] the particle set is efficiently guided from edge low-level hypotheses. A limitation of these methods often lies in their execution time.

We propose, in the spirit of [13], to integrate geometrical and color features along edges in the pose estimation phase. The general idea is to combine in the criterion to be optimized a geometrical information provided by the distances between model and image edges with a denser color information through object/background color separation statistics along the model edges. A low-level multiple hypotheses edge matching process is also embedded in our framework. Like in our previous work [15], the model projection and model edge generation phase relies on the graphics process units (GPU) in order to handle complex 3D models, of any shape and to be reasonably time-consuming. We choose to restrict to a single nonlinear minimization in our pose estimation technique due to computational limits fixed by our application, but integrating our method into a particle filtering framework, as in [4], [18] would also improve performances.

The remainder of the paper is organized as follows. Section II presents the general pose estimation framework. Section III and IV respectively describe how the geometrical and color features are determined and combined. Finally some experimental results are provided in Section V.

II. COMBINING GEOMETRICAL AND COLOR EDGE-BASED FEATURES IN 3D MODEL-BASED TRACKING

Our problem is restricted to model-based tracking, using a 3D model of the target. The goal is to estimate the camera pose $\mathbf{r}$ by minimizing, with respect to $\mathbf{r}$, the error Δ between the observed data $\mathbf{s}^*$ and the current value $\mathbf{s}(\mathbf{r})$ of the same

A. Petit is with INRIA Rennes - Bretagne Atlantique, Lagadic Team, France, Antoine.Guillaume.Petit@inria.fr

E. Marchand is with Université de Rennes 1, IRISA, Lagadic Team, France, Eric.Marchand@irisa.fr

K. Kanani is with Astrium, Toulouse, France

features projected in the image according to the current pose:

$$\Delta(\mathbf{r}) = \sum_i \rho(s_i(\mathbf{r}) - s_i^*) \qquad (1)$$

where ρ is a robust estimator, which reduces the sensitivity to outliers. This is a non-linear minimization problem with respect to the pose parameters $\mathbf{r}$. We follow the Virtual Visual Servoing framework [5], similar to the Gauss-Newton approach. In this sense, we consider a robust control law which computes the virtual camera velocity skew $\mathbf{v}$ in order to minimize $\mathbf{s}(\mathbf{r}) - \mathbf{s}^*$:

$$\mathbf{v} = -\lambda(\mathbf{D}\mathbf{L_s})^+\mathbf{D}(\mathbf{s}(\mathbf{r}) - \mathbf{s}^*) \qquad (2)$$

where $\mathbf{L_s}^+$ is the pseudo inverse of $\mathbf{L_s}$, the interaction (or Jacobian) matrix of the feature vector $\mathbf{s}$, which links $\mathbf{v}$ to the velocity of the features in the image. λ is a proportional gain and $\mathbf{D}$ is a weighting matrix associated to the Tukey robust estimator. Finally, the new pose $\mathbf{r}_{k+1}$, represented by its homogeneous matrix ${}^{c_{k+1}}\mathbf{M}_o$, can be computed using the exponential map [11]:

$${}^{c_{k+1}}\mathbf{M}_o = {}^{c_{k+1}}\mathbf{M}_{c_k}\,{}^{c_k}\mathbf{M}_o = e^{-\mathbf{v}c_k}\;\mathbf{M}_o \qquad (3)$$

Our challenge is to combine geometrical edge-based features with a complementary type of features in order to overcome the limitations of classical edge-based approaches. Since we deal with potentially textureless 3D objects, combining this information with texture features would not be suitable. Besides, our idea is to avoid any image extraction or segmentation that would lead to outliers and mismatches and that would make some information lost. We propose to rely on denser and more accurate features. In this sense, we follow [13], for which color features are integrated with classical geometrical edge-based features. These features refer to the Contracting Curve Algorithm [7], which is designed to optimize the separation of color statistics collected on both sides of the projected edges of the 3D model. Δ can then be rewritten as:

$$\Delta = w^g\Delta^g + w^c\Delta^c \qquad (4)$$

Δ^g refers to the geometrical error function and Δ^c stands for the color-based one. w^g and w^c are weighting parameters. Both kinds of features rely on the projection of the 3D model, in the vicinity of the projected model edges.

III. Geometrical edge features

A. Model projection and generation of model edge points

As in our previous work [15], we propose to automatically manage the projection of the model and to determine the visible and prominent edges from the rendered scene, by considering the direct use of a complete model, which can be textured or not. By using the graphics process units (GPU) and a 3D rendering engine, we avoid any manual pre-processing.

For each acquired image I_{k+1}, the model is rendered with respect to the previous pose $\mathbf{r}_k$. The goal is to obtain a set of 3D points $\mathbf{X}_i$ that belong to target rims, edges and visible textures from the rendered scene. By processing the depth buffer through a Laplacian filter, we can determine the discontinuities which suit the geometrical appearance of the visible scene, resulting in a binary edge map. We have implemented the filtering computations on the GPU through shader programming, reducing computational time. In the case of a textured 3D model, we propose to combine the depth discontinuities with texture discontinuities. The rendered textures are passed through a Canny edge algorithm and the obtained edges are added to the ones generated from the depth buffer. We can sample this set of edge points along the x and y coordinates of the image in order to keep a reasonable number points $\mathbf{x}_i$. The 3D coordinates of the determined edge points in the scene are retrieved using the depth buffer and the pose used to project the model. Besides, the computation of both the edge and color based objective functions requires the orientation of the edge underlying a point $\mathbf{x}_i$. For the texture edges, it is done within the Canny algorithm on the rendered textures. For the depth edges, we compute the Sobel gradients along x and y on a gray-level image of the normal map of the scene, filtered using a Gaussian kernel, since the rendering phase can suffer from aliasing. These basic image processing steps are processed on the GPU, optimizing computations.

B. Feature computation and interaction matrix

The edge-based function Δ^g is computed in a similar way to [15]. From the model edge points we perform a 1D search along the normal of the underlying edge of each $\mathbf{x}_i(\mathbf{r}_k)$. A common approach is to choose the pixel with the maximum gradient as the matching edge point $\mathbf{x}'_i$ in the image. Once correspondences are established, we consider the distance between the projected 3D line $l_i(\mathbf{r})$ underlying the projected model edge point $\mathbf{x}_i(\mathbf{r})$ (projected from the 3D point $\mathbf{X}_i$) and the selected matching point $\mathbf{x}'_i$ in the image. Δ^g can be written as:

$$\Delta^g = \sum_i \rho^g(s_i^g(\mathbf{r}) - s_i^{g*}) = \sum_i \rho^g(d_\perp(l_i(\mathbf{r}), \mathbf{x}'_i))) \qquad (5)$$

with $s_i^g = d_\perp(l_i(\mathbf{r}), \mathbf{x}'_i))$, $s_i^{g*} = 0$ and ρ^g is a Tukey robust estimator. This function improves the approaches in [4], [13], [20], which consider the distance between $\mathbf{x}_i(\mathbf{r})$ and $\mathbf{x}'_i$ along the 2D normal vector to the edge underlying $\mathbf{x}_i(\mathbf{r}_k)$, determined at the model projection phase. A key requirement to our method is to compute the 3D equation of the line l_i in the world frame in order to perform its projection during the minimization process and to compute the interaction matrix $\mathbf{L}_{d_\perp}$, related to the point to line distance. This is addressed through the knowledge of the edge orientation during the rendering phase and the knowledge of the normal to the surface underlying l_i, retrieved with the rendered normal map. For the complete computation of $\mathbf{L}_{d_\perp}$, see [5].

C. Multiple-hypotheses framework

Regarding the geometrical edge registration process, a novel multiple-hypotheses solution is proposed to improve robustness. This approach extends the one presented in [15], [19] by taking advantage of some elements proposed by [18]. In [15], the idea was to consider and register different hypotheses corresponding to potential edges. They correspond to different local extrema of the gradient along the scan line. But the projected model edge points are treated independently, regardless their membership to primitives such as lines or particular curves. To overcome this issue, the idea is to cluster the model edge points into different primitives and to register different hypotheses consistently with these primitives. Here, we restrict to line primitives, for computational reasons.

a) Clustering model edge points into lines: from the edge map provided by the projection of the 3D model, a set of N_l 2D line segments $\{\mathbf{l}^i\}_{i=1}^{N_l}$ is extracted using a Hough line detector. A model edge point x_k for which the distance to the closest line is under a certain threshold is associated to this line. We obtain a set of clusters $\{\mathbf{C}^i\}_{i=1}^{N_l}$ of model edge points corresponding to the extracted lines $\{\mathbf{l}^i\}_{i=1}^{N_l}$.

b) Multiple-hypotheses registration: for each cluster $\mathbf{C}^i$, we process in a similar manner to [18]. For a point $\mathbf{x}_{i,j}$ in $\mathbf{C}^i$, we consider several edge hypotheses $\mathbf{x}'_{i,j,l}$ (see Figure 1). These candidates are then classified into k_i sets of points or classes $\{c^i_m\}_{m=1}^{k_i}$ using the *k-mean* algorithm, each c^i_m being represented by a mean line l^i_m, which best fits the points of c^i_m, and a corresponding weight w^i_m. w^i_m represents the likelihood of class c^i_m with respect to the others in $\mathbf{C}^i$.

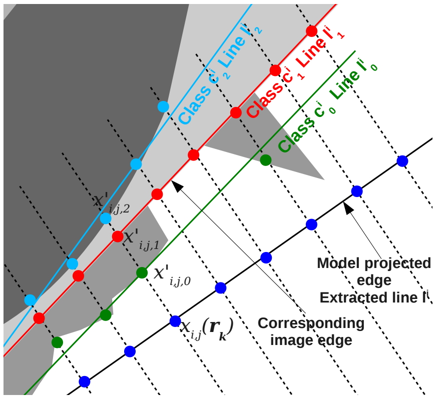

Fig. 1: Multiple hypotheses framework. Points $\mathbf{x}_{i,j}$ (blue dots) form the cluster $\mathbf{C}^i$ corresponding to the extracted line l^i. For a point $\mathbf{x}_{i,j}$, several hypotheses $\mathbf{x}'_{i,j,l}$ are registered, and are used to build classes c^i_m tied to lines l^i_m. The hypotheses in the class c^i_1 (red dots), which matches l^i, will have higher weights than the hypotheses of classes c^i_0 and c^i_2 (green and light blue dots), which correspond to clutter. Thus model edge points $\mathbf{x}_{i,j}$ will more likely converge towards the hypotheses of class c^i_1.

In [18], random weighted draws are then performed in order to get several hypotheses on the pose. Since it is time consuming, here, we simply use the weights w^i_m to determine the probability $\pi_{i,j,l}$ of a candidate $\mathbf{x}'_{i,j,l}$ to belong to a line. If $c^i_{m_l}$ denotes the class including $\mathbf{x}'_{i,j,l}$, we have:

$$\pi_{i,j,l} = p(c^i_{m_l} \cap \mathbf{x}'_{i,j,l}) = p(c^i_{m_l})p(\mathbf{x}'_{i,j,l} \mid c^i_{m_l}) \qquad (6)$$

with $p(c^i_{m_l}) \propto w^i_{m_l}$ and where the probability $p(\mathbf{x}'_{i,j,l} \mid c^i_{m_l})$ is related to the distance between $\mathbf{x}'_{i,j,l}$ and the mean line $l^i_{m_l}$ associated to $c^i_{m_l}$. The function corresponding to the points $\mathbf{x}_{i,j}$ clustered into the line classes $\{\mathbf{C}^i\}_{i=1}^{N_l}$ can be written as:

$$\Delta^g_0 = \sum_i \sum_j \rho^g_0(\min_l \pi_{i,j,l}(d_\perp(l_{i,j}(\mathbf{r}), \mathbf{x}'_{i,j})))) \qquad (7)$$

with $l_{i,j}$ the projected line, for pose $\mathbf{r}$, underlying $\mathbf{x}_{i,j}$. For the remaining points $\mathbf{x}_i$ which have not been classified into line clusters, we apply the multiple hypotheses approach proposed in [15], giving:

$$\Delta^g_1 = \sum_i \rho^g_1(\min_j d_\perp(l_i(\mathbf{r}), \mathbf{x}'_{i,j})) \qquad (8)$$

$$\Delta^g = \Delta^g_0 + \Delta^g_1 \qquad (9)$$

IV. Color features

The color-based function Δ^c is elaborated to characterize the separation between both sides of projected model edges, by relying on color information. In order to compute Δ^c, as in [13], we restrict ourself to silhouette edges, since it makes more sense than for crease or texture edges and it limits the computational burden.

The principle is to compute local color statistics (means and covariances) along the normal to the projected model silhouette edges, on both sides. Then for each pixel along the normal, we determine a residual representing the consistency of the pixel with these statistics, according to a fuzzy membership rule to each side. A first contribution we propose is to use a robust M-estimator in the computation of Δ^c. Another contribution consists in adding consistency with respect to the color statistics computed on the previous frame.

A. Computation of color local statistics

Given the set of projected silhouette model edge points $\mathbf{x}_i(\mathbf{r})$, determined from $\mathbf{X}_i$ (see Section III), we compute color statistics up to the 2^{nd} order, on both side of the edge (object O and background B) using $2D+1$ pixels along the edge normal $\mathbf{n}_i$, up to a distance L (see Figure 2). For the object side, we have:

$$\nu_i^{0,O} = \sum_{j=-D}^{D} \mu^O_{i,j} \qquad \boldsymbol{\nu}_i^{1,O} = \sum_{j=-D}^{D} \mu^O_{i,j}\mathbf{I}(\mathbf{y}_{i,j}) \qquad (10)$$

$$\boldsymbol{\nu}_i^{2,O} = \sum_{j=-D}^{D} \mu^O_{i,j}\mathbf{I}(\mathbf{y}_{i,j})\mathbf{I}(\mathbf{y}_{i,j})^T \qquad (11)$$

$\mathbf{y}_{i,j} = \mathbf{x}_i(\mathbf{r}) + L\overline{d}\mathbf{n}_i$ are the pixels located on both sides. $\overline{d} = \frac{j}{D}$ is the normalized signed distance to $\mathbf{x}_i(\mathbf{r})$. $\mathbf{I}(\mathbf{y}_{i,j})$ is the RGB color vector of pixel $\mathbf{y}_{i,j}$ and $\mu^O_{i,j}$ are local weights giving a higher confidence on the object side, close to the edge (see [7]). As in [13] these statistics are then blurred with respect to the other silhouette points, and normalized, to define RGB means $\overline{\mathbf{I}}^O_i$ and covariances $\overline{\mathbf{R}}^O_i$ for $\mathbf{x}_i(\mathbf{r})$:

$$\tilde{\nu_i}^{k,O} = \sum_j e^{-\lambda|i-j|}\nu_j^{k,O}, k = 0, 1, 2 \qquad (12)$$

$$\overline{\mathbf{I}}^O_i = \frac{\tilde{\boldsymbol{\nu}_i}^{1,O}}{\tilde{\nu_i}^{0,O}} \quad \text{and} \quad \overline{\mathbf{R}}^O_i = \frac{\tilde{\boldsymbol{\nu}_i}^{2,O}}{\tilde{\nu_i}^{0,O}} \qquad (13)$$

We proceed the same way for the background B.

B. Feature computation and interaction matrix

The consistency of observed color components of pixels $\mathbf{y}_{i,j}$ according to the computed color statistics are evaluated using a function $a(\overline{d})$ as a fuzzy membership rule to the object, with:

$$a(\overline{d}) = \frac{1}{2}(erf(\frac{\overline{d}}{\sqrt{2}\sigma} + 1), \overline{d} = -1..1 \qquad (14)$$

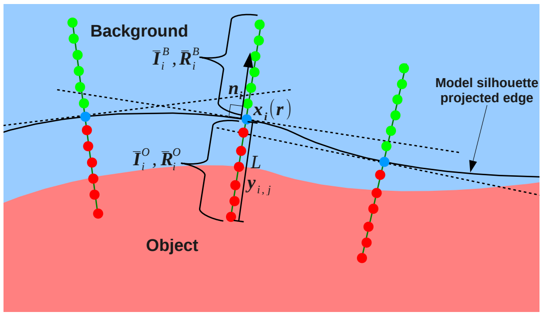
Background
$\bar{I}_i^B, \bar{R}_i^B$
n_i
$x_i(r)$
Model silhouette projected edge
$\bar{I}_i^O, \bar{R}_i^O$
L
$y_{i,j}$
Object

Fig. 2: Collection of local color statistics on both the background (B) and object (O) sides.

erf is the error function [1]. σ is a standard deviation defining the sharpness of the membership rule. Both object and background statistics can thus be mixed:

$$\hat{\mathbf{I}}_i(\mathbf{r}) = a(\bar{d}(\mathbf{r}))\bar{\mathbf{I}}_i^O + (1 - a(\bar{d}(\mathbf{r})))\bar{\mathbf{I}}_i^B \tag{15}$$

$$\hat{\mathbf{R}}_i(\mathbf{r}) = a(\bar{d}(\mathbf{r}))\bar{\mathbf{R}}_i^O + (1 - a(\bar{d}(\mathbf{r})))\bar{\mathbf{R}}_i^B \tag{16}$$

and the error $\mathbf{e}_{i,j}^c(\mathbf{r}) = \hat{\mathbf{I}}_i(\mathbf{r}) - \mathbf{I}(\mathbf{y}^{)}_{i,j}$ is normalized to define the color feature $s_{i,j}^c(\mathbf{r})$ as:

$$s_{i,j}^c(\mathbf{r}) = \sqrt{\mathbf{e}_{i,j}^c(\mathbf{r})^T \hat{\mathbf{R}}_i^{-1} \mathbf{e}_{i,j}^c(\mathbf{r})} \tag{17}$$

$\hat{\mathbf{I}}_i(\mathbf{r})$ represents a desired color value for the j^{th} pixel $\mathbf{y}_{i,j}$ on the normal $\mathbf{n}_i$, wether it is on the object O or background side B, with $j = D\bar{d}(\mathbf{r})$. The idea is to optimize the position $\bar{d}(\mathbf{r})$ of the membership rule a along the normal, so that the desired value $\hat{\mathbf{I}}_i(\mathbf{r})$ best matches the actual value $\mathbf{I}(\mathbf{y}_{i,j})$, minimizing $\mathbf{e}_{i,j}^c(\mathbf{r})$ and $s_{i,j}^c(\mathbf{r})$. The dependence of $\hat{\mathbf{R}}_i(\mathbf{r})$ on to the pose $\mathbf{r}$ is neglected to reduce computations.

In order to cope with possible outliers and to improve robustness, we propose to integrate a M-estimator in Δ^c, which becomes:

$$\begin{aligned} \Delta^c &= \sum_i \sum_j \rho^c(s_{i,j}^c(\mathbf{r}) - s_{i,j}^c{}^*) \\ &= \sum_i \sum_j \rho^c(\sqrt{\mathbf{e}_{i,j}^c(\mathbf{r})^T \hat{\mathbf{R}}_i^{-1} \mathbf{e}_{i,j}^c(\mathbf{r})}) \end{aligned} \tag{18}$$

with $s_{i,j}^c{}^* = 0$. As for Δ^g, we choose a Tukey estimator. The interaction matrix $\mathbf{L}_{s_{i,j}^c}$ can be computed as follows:

$$\mathbf{L}_{s_{i,j}^c} = \frac{\partial s_{i,j}^c(\mathbf{r})}{\partial \mathbf{r}} = \frac{1}{s_{i,j}^c}\left(\frac{\partial \mathbf{e}_{i,j}^c(\mathbf{r})}{\partial \mathbf{r}}\right)^T \hat{\mathbf{R}}_i^{-1} \mathbf{e}_{i,j}^c(\mathbf{r}) \tag{19}$$

and the interaction matrix $\mathbf{L}_{\mathbf{e}_{i,j}^c} = \frac{\partial \mathbf{e}_{i,j}^c(\mathbf{r})}{\partial \mathbf{r}}$ is computed as:

$$\mathbf{L}_{\mathbf{e}_{i,j}^c} = \frac{\partial \hat{\mathbf{I}}_i(\mathbf{r})}{\partial \mathbf{r}} = (\bar{\mathbf{I}}_i^O - \bar{\mathbf{I}}_i^B)\frac{\partial a(\bar{d}(\mathbf{r}))}{\partial d}\frac{\partial d}{\partial \mathbf{r}} \tag{20}$$

As in [4], [13], $\frac{\partial d}{\partial \mathbf{r}} = \frac{1}{L}\mathbf{n}_i^T \mathbf{L}_{\mathbf{x}_i}$ with $\mathbf{L}_{\mathbf{x}_i} = \frac{\partial \mathbf{x}_i(\mathbf{r})}{\partial \mathbf{r}}$ being the interaction matrix of a point, which is given by:

$$\mathbf{L}_{\mathbf{x}_i} = \mathbf{K}\begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & (1+y^2) & xy & -x \end{bmatrix} \tag{21}$$

with

$$\mathbf{K} = \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \tag{22}$$

the focal ratio parameters of the camera. (x, y) denotes the meter coordinates of the image point $\mathbf{x}_i$, and Z the depth of the corresponding 3D point.

C. Temporal consistency

For more accuracy, we introduce a temporal constraint to the objective function by considering the information of past frames. The idea is to integrate the color statistics computed on the previous frame ${}^P\mathbf{I}$ for the silhouette edge points $\mathbf{x}_i(\mathbf{r}_k)$ at the first iteration of the minimization process. $\mathbf{e}_{i,j}^c(\mathbf{r})$ becomes:

$$\mathbf{e}_{i,j}^c(\mathbf{r}) = \alpha\hat{\mathbf{I}}_i(\mathbf{r}) + \beta({}^P\hat{\mathbf{I}}_i(\mathbf{r})) - \mathbf{I}(\mathbf{y}_{i,j}) \tag{23}$$

with $\alpha + \beta = 1$, and we have:

$$\begin{aligned} \mathbf{L}_{\mathbf{e}_{i,j}^c} &= \alpha\frac{\partial \hat{\mathbf{I}}_{i,d}(\mathbf{r})}{\partial \mathbf{r}} + \beta\frac{\partial {}^P\hat{\mathbf{I}}_i(\mathbf{r})}{\partial \mathbf{r}} \\ &= (\alpha(\bar{\mathbf{I}}_i^O - \bar{\mathbf{I}}_i^B) + \beta({}^P\bar{\mathbf{I}}_i^O - {}^P\bar{\mathbf{I}}_i^B))\frac{\partial a(\bar{d}(\mathbf{r}))}{\partial \mathbf{r}} \end{aligned} \tag{24}$$

D. Combination with geometrical features

The combination of the geometrical features and color features $s_i^g(\mathbf{r})$ and $s_{i,j}^c(\mathbf{r})$ in the Virtual Visual Servoing framework is achieved by stacking these features into a global feature vector $\mathbf{s}$ and their corresponding interaction matrix into a global interaction matrix:

$$\begin{aligned} \mathbf{s} &= \begin{bmatrix} w^g s_1^g & \cdots & w^g s_{N_g}^g & w^c s_{1,1}^c & \cdots & w^c s_{N_s,2D}^c \end{bmatrix}^T \\ \mathbf{L}_s &= \begin{bmatrix} w^g \mathbf{L}_1^g & \cdots & w^g \mathbf{L}_{N_g}^g & w^c \mathbf{L}_{1,1}^c & \cdots & w^c \mathbf{L}_{N_s,2D}^c \end{bmatrix}^T \end{aligned} \tag{25}$$

with N_g the number of geometrical features. N_s refers to the number of model edge points belonging to the silhouette of the projected model, so that $N_c = 2DN_s$ accounts for the number of color features, with D the range along the normals to the edge points. $\mathbf{s}$ is a $N_g + N_c$ vector and $\mathbf{L}_s$ is a $(N_g + N_c) \times 6$ matrix. Regarding the weighting matrix $\mathbf{D}$, it is written as $\mathbf{D} = blockdiag(\mathbf{D^g}, \mathbf{D^c})$, where $\mathbf{D^g}$ and $\mathbf{D^c}$ are the weighting matrices associated to the robust estimators ρ^g and ρ^c.

V. Experimental results

In this section we validate the proposed method, both qualitatively on real images and qualitatively on synthetic images and the advantages of our contributions are verified.

A. Implementation

The rendering process of the 3D polygonal and textured model relies on OpenSceneGraph, which is flexible 3D rendering engine. As presented in Section II.B, we have considered shader programming for some image processing steps during the rendering and edge generation phases. This is done using OpenGL Shading Language (GLSL). The remainder of the algorithm has been implemented thanks to the C++ ViSP library [12]. Regarding hardware, an NVIDIA NVS 3100M graphic card has been used, along with a 2.8GHz Intel Core i7 CPU. For all the following tests, our algorithm has been initialized manually. Besides, since it is not available online, we have not implemented the algorithm of [13] exactly the same way as in the paper. Instead we have equivalently tested our new solution without the M-Estimators for both edge-based and color-based objective functions, without the multiple-hypotheses framework and without the temporal consistency for the color-based function.

B. *Results on synthetic images*

We have achieved a quantitative evaluation of our algorithm on synthetic images, using a realistic ray-tracing simulator developed by Astrium for space environments. We present the same sequence as in [15], which features a Spot satellite and which is provided with ground truth. For space debris removal concerns, we consider an arbitrary rotation for the target attitude and a chaser spacecraft is supposed to be located on a similar orbit, with a slightly different eccentricity in order to make the chaser fly around the target. We have investigated the performances of our algorithm comparatively to our former solution [15], which we denote as the Nominal Mode (*NM*) and to our implementation of the method presented in [13], denoted by *PM* (see provided video). The results can be seen on Figure 4 where the accuracy of rotation and translation components of an estimated camera pose $\hat{\mathbf{r}}$ with respects to the true pose $\mathbf{r}^*$ is determined throughout the sequence, through error plots on the pose parameters. For our new solution and for *NM*, the tracking is properly performed, as depicted on the image sequence on Figure 3. In terms of pose errors, the approach presented in this paper shows better performances, especially when the satellite is far, with low luminosity (between frame 1200 and 1500). With *PM*, the tracking fails, mainly due to the absence of a multiple hypothesis framework and to the absence of M-estimators for both edge-based and color-based functions.

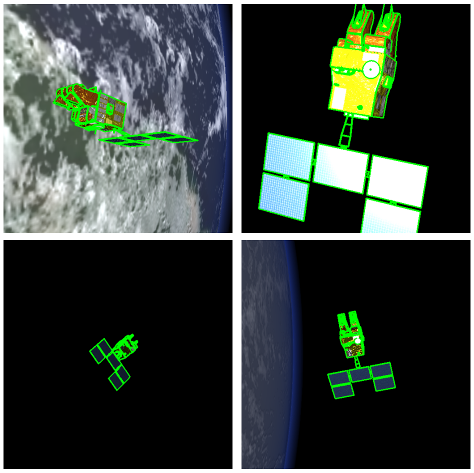

Fig. 3: Tracking for the Spot sequence with the proposed method.

We have also examined and verified the effectiveness and benefit of some of our contributions which are:

- Incorporating line primitives into our multiple-hypotheses framework for the edge-based registration process, what is described in Section III.B. This contribution is denoted by $C1$.
- Integrating the color-based objective function to the global function, denoted by $C2$.
- Temporal consistency for the color-based function ($C3$), presented in Section IV.C.

The results are represented on Table I by root mean square errors on the pose parameters between frame 1200 and 1500, which is the most challenging phase, to better enhance the advantages of the proposed methods. Execution times are also given (Table II).

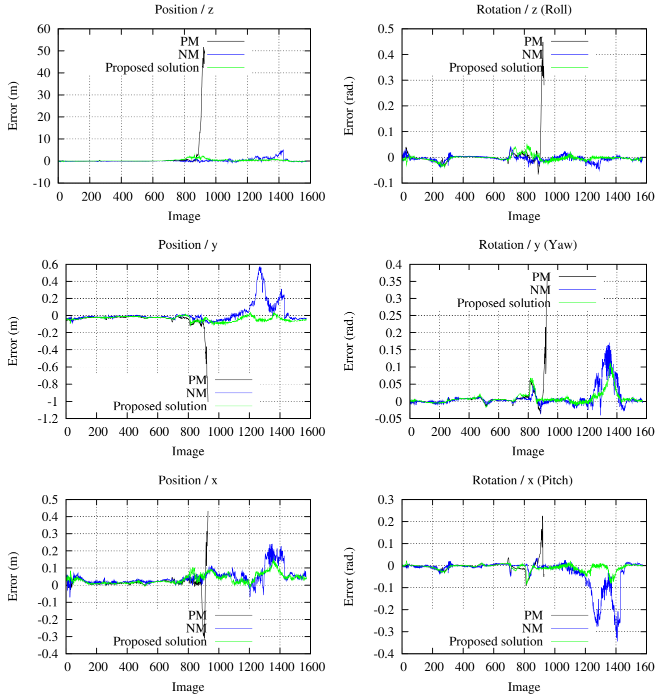

Fig. 4: Estimated camera pose parameters of the target over all the sequence, along with the ground truth, for the nominal mode (*NM*), the solution implemented from [13] (*PM*), and the proposed solution.

TABLE I: RMS errors for the Nominal Mode (*NM*), along with the different contributions (*C1*, *C2*, *C3*), for frames 1200-1500. t_x, t_y, t_z (in meters) and R_x, R_y, R_z (in radians) respectively refer to translation and rotation (Euler angles) parameters.

Mode	t_x	t_y	t_z	R_x	R_y	R_z
NM	0.118	0.238	1.771	0.158	0.069	0.016
NM, C1	0.108	0.230	1.537	0.145	0.061	0.015
NM, C2	0.082	0.114	0.517	0.066	0.037	0.017
NM, C2, C3	0.076	0.090	0.486	0.055	0.038	0.014
NM, C1, C2, C3	0.073	0.045	0.425	0.027	0.037	0.005

C. *Results on real images*

Soyuz sequence: this sequence shows the Soyuz TMA-03M undocking from the International Space Station (ISS). We also run on this sequence the Nominal Mode (*NM*) [15], the algorithm presented in [13] (*PM*) and the one described in this paper. As seen on Figures 5, the tracking is successfully achieved, whereas it tends to fail for both *NM* (Figure 6a) and *PM* (Figure 6b) modes.

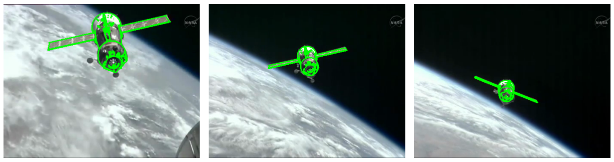

Fig. 5: Tracking for the Soyuz sequence with the new proposed solution

TABLE II: Mean execution times for frames 1200-1500.

Mode	Time (s)
NM	0.85
NM, C1	0.111
NM, C2	0.301
NM, C2, C3	0.306
NM, C1, C2, C3	0.344

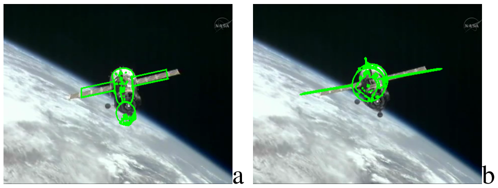

Fig. 6: Tracking for the Soyuz sequence with NM (a) and PM (b).

Mock-ups video sequences: two sequences are processed. In a similar way to our former work [14], the first one has been taken on the Lagadic robotic platform and Astrium provided a 1/50 mock-up of Amazonas-2, a telecom satellite. A six degrees of freedom robot has been used to simulate a space rendezvous, with a camera mounted on the end-effector of the robot, and enables to have regular and quite realistic movements. Let us however note that the specific dynamic of the chaser spacecraft is not considered in this paper. Sun illumination is also simulated by a spot light located around the scene. As the complete 3D model of the satellite shows differences with respect to the mock-up, it has been redesigned manually. Tracking results can be observed on Figure 7(a-c). The second sequences has been provided by Astrium and concerns a fly-around a mock-up of Envisat, an observation satellite which can be now considered as a space debris (Figure 7(d-f)).

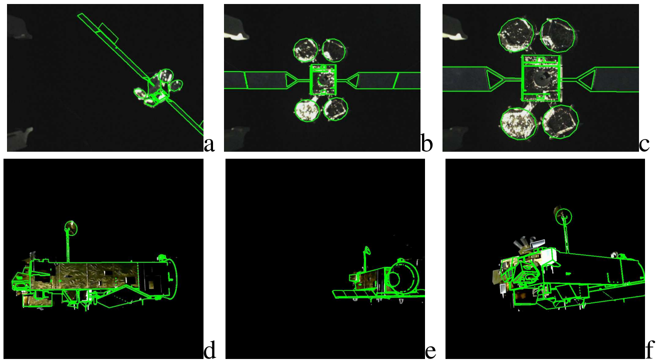

Fig. 7: Tracking results for the sequences involving Amazonas (a-c) and Envisat (d-f) mock-ups.

VI. Conclusion

In this paper we have presented a robust and hybrid approach of 3D visual model-based object tracking. The general idea was to combine in the global criterion to be minimized two complementary cues: a geometrical one, relying on distances between edge features, and a intensity-based one, relying on color features computed around silhouette edges. For robustness purposes, we employed a new multiple hypotheses framework taking advantage of line primitives, along with M-estimators for both objective functions, and we added temporal consistency for the color-based features. Our approach has been tested via various experiments, on both synthetic and real images, in which our contributions have shown notable results and improvements in terms of accuracy and robustness, with regards to state-of-the art approaches.

References

[1] M. Abramowitz. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,*. Dover Publications, 1974.

[2] G. Bleser, Y. Pastarmov, and D. Stricker. Real-time 3d camera tracking for industrial augmented reality applications. *Journal of WSCG*, pages 47–54, 2005.

[3] C. Bonnal, J.M. Ruault, and M.C. Desjean. Active debris removal: Recent progress and current trends. *Acta Astronautica*, 85:51–60, 2013.

[4] C. Choi and H. I. Christensen. Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *Int. J. Rob. Res.*, 31(4):498–519, April 2012.

[5] 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.

[6] 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.

[7] R. Hanek and M. Beetz. The contracting curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria. *Int. J. of Computer Vision*, 59(3):233–258, 2004.

[8] G. Klein and T. Drummond. Tightly integrated sensor fusion for robust visual tracking. 22(10):769–776, 2004.

[9] G. Klein and D. Murray. Full-3d edge tracking with a particle filter. In *Proc. British Machine Vision Conference, BMVC'06*, volume 3, pages 1119–1128, Edinburgh, September 2006.

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

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

[12] E. Marchand, F. Spindler, and F. Chaumette. ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52, December 2005.

[13] G. Panin, E. Roth, and A. Knoll. Robust contour-based object tracking integrating color and edge likelihoods. In *Proc. of the Vision, Modeling, and Visualization Conference 2008, VMV 2008*, pages 227–234, Konstanz, Germany, October 2008.

[14] A. Petit, E. Marchand, and K. Kanani. Vision-based space autonomous rendezvous : A case study. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'11*, pages 619–624, San Francisco, USA, September 2011.

[15] A. Petit, E. Marchand, and K. Kanani. Tracking complex targets for space rendezvous and debris removal applications. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'12*, pages 4483–4488, Vilamoura, Portugal, October 2012.

[16] M. Pressigout and E. Marchand. Real-time hybrid tracking using edge and texture information. *Int. Journal of Robotics Research, IJRR*, 26(7):689–713, July 2007.

[17] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE Int. Conf. on Computer Vision*, volume 2, pages 1508–1515, Beijing, China, 2005.

[18] C. Teulière, E. Marchand, and L. Eck. Using multiple hypothesis in model-based tracking. In *IEEE Int. Conf. on Robotics and Automation, ICRA'10*, pages 4559–4565, Anchorage, Alaska, May 2010.

[19] 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'04*, pages 48–57, Arlington, VA, November 2004.

[20] H. Wuest and D. Stricker. Tracking of industrial objects by using cad models. *Journal of Virtual Reality and Broadcasting*, 4(1), April 2007.

[21] Y. Yoon, A. Kosaka, and A. C. Kak. A new kalman-filter-based framework for fast and accurate visual tracking of rigid objects. *IEEE Trans. on Robotics*, 24(5):1238–1251, October 2008.