Tracking complex targets for space rendezvous and debris removal applications

Antoine Petit, Eric Marchand, Keyvan Kanani

***Abstract*— In the context of autonomous rendezvous and space debris removal, visual model-based tracking can be particularly suited. Some classical methods achieve the tracking by relying on the alignment of projected lines of the 3D model with edges detected in the image. However, processing complete 3D models of complex objects, of any shape, presents several limitations, and is not always suitable for real-time applications. This paper proposes an approach to avoid these shortcomings. It takes advantage of GPU acceleration and 3D rendering. From the rendered model, visible edges are extracted, from both depth and texture discontinuities. Correspondences with image edges are found thanks to a 1D search along the edge normals. Our approach addresses the pose estimation task as the full scale nonlinear minimization of a distance to a line. A multiple hypothesis solution is also proposed, improving tracking robustness. Our method has been evaluated on both synthetic images (provided with ground truth) and real images.**

I. Introduction

A. Context : active removal of heavy space debris

The active removal of heavy space debris (typically larger than 1000kg) has been identified as a key development to control the growth in the debris population and to limit the risk for active satellites. In that context, Astrium has been working on optimization and implementation of sensors and navigation solutions onboard a Debris Removal Vehicle with the main objective to ensure high safety proximity maneuvers. In particular, special attention has been paid to the design of autonomous, vision-based navigation solutions for uncooperative rendezvous with space debris. In this paper we focus on an algorithm which achieves 3D tracking of complex target and allows fine estimation of chaser states with respect to a target spacecraft or debris.

This tracking algorithm has been tested on real images, such as Soyuz-TMA rendezvous with ISS and Atlantis space shuttle pitch maneuver to demonstrate its robustness on real data. A test campaign on simulated images has also been carried out to quantitatively show the performances and robustness of both tracking and associated navigation.

B. 3D model based tracking

In order to estimate the pose of a camera with respect to a specific scene, common model-based approaches use either point [2], edge features [9], [4], [3] or a combination of both [17], [15]. Edge features offer a good invariance to illumination changes or image noise and are particularly suitable with poorly textured scenes, whether the scene is in industrial, outdoor or indoor environments. 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, extracted thanks to a 1D search for gradient maxima along the model edge normals. Weighted numerical nonlinear optimization techniques like Newton-Raphson or Levenberg-Marquardt are usually implemented for the minimization. To reject outliers, methods like RANSAC [2] or the use of M-Estimators such as the Tuckey estimator [17], [3] are common trends to make the algorithm robust to occlusions and illumination variations. But the robustness deteriorates when ambiguities between different edges occur, especially between geometrical and texture edges of the scene. One way to address this issue has been to fuse the information of edge features with information given by particular key-points [13] or by other sensors [7]. Other solutions have considered multiple hypothesis for potential edge-locations in the image [17], [16].

Regarding implementation purposes, most of these approaches process 3D models which are made-up of segments. But achieving the model projection in the image has limitations and some problems appear when dealing with objects made of cylindrical, spherical, curved or complex shapes. Furthermore, provided complete polygonal models for complex objects can be too heavy and need to be manually redesigned to keep the most relevant edges of the scene and to make the algorithm computationally efficient.

A first challenge of the solution proposed in this paper is to process a complete polygonal 3D model, in order to track the object in the image and to estimate the camera pose. In this sense, the whole information from the geometrical shape of any kind of scene can be used and a heavy phase of a manual redesign of the model is completely avoided. Our method relies on the use of the graphics process units (GPU) and of a 3D rendering engine. This allows to automatically manage the projection of the model and to determine the visible and prominent edges from the rendered scene. Such method has also been considered in [18], [14]. An advantage of these technique is to automatically handle the hidden face removal process and to implicitly handle auto occlusions. A second challenge is to improve the robustness by combining both depth and texture edges and by including multiple hypothesis in the edge matching process.

The remainder of the paper is organized as follows. Section II presents the common approaches in 3D model-based tracking and gives an overview of the proposed

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

Keyvan Kanani is with Astrium, Toulouse, France

method. Section III explains how complex 3D models can be handled efficiently, Section IV details the tracking and pose estimation processes. Some experimental results are given in Section V.

II. Classical 3D model based tracking approaches

Our problem is restricted to model-based tracking, using a 3D model of the target. The purpose is to compute the camera pose which provides the best alignment between edges of the projected model and edges extracted in the image.

Such approaches have proved to be very efficient and various authors have proposed different formulations of the problem (eg, [9], [4], [3]). Although one can find some differences in these various solutions, the main idea is the following. Given a new image, the 3D model of the scene or the target is projected in the image according to the estimated previous camera pose $\mathbf{r}$. Each projected line $l_i(\mathbf{r}) = pr(L_i, \mathbf{r})$ of the model is then sampled, leading to a set of 2D points $\{\mathbf{x}_{i,j}\}$. Then from each sample point $\mathbf{x}_{i,j}$ a 1D search along the normal of the projected edge is performed to find a corresponding point $\mathbf{x}'_{i,j}$ in the image.

In order to compute the new pose, the distances between points $\mathbf{x}'_{i,j}$ and the projected lines l_i are minimized with respect to the following criteria [3] :

$$\Delta = \sum_i \sum_j \rho(d_\perp(l_i(\mathbf{r}), \mathbf{x}'_{i,j})) \quad (1)$$

where $d_\perp(l_i(\mathbf{r}), \mathbf{x}'_{i,j})$ is the distance between a point $\mathbf{x}'_{i,j}$ and the corresponding line $l_i(\mathbf{r})$ projected in the image from a pose $\mathbf{r}$. ρ is a robust estimator, which reduces the sensitivity to outliers. This is a non-linear minimization process with respect to the pose parameters $\mathbf{r}$. In [3], the minimization process follows the Virtual Visual Servoing framework, similar to the Gauss-Newton approach.

One of the drawbacks of these methods is that the 3D model is usually made of segments, which implies dealing with simple objects or manually pre-processing the CAD model.The approach presented in this paper considers the direct use of a complete but non necessarily polygonal model, which can be textured or untextured.

Considering complex shape targets leads to forget the notion of 3D sharp edges as in [3] or in our previous work [12] and to deal only with 3D points that belongs indifferently to sharp edges or to the "occlusion boundaries" or rims [8]. Two issues have then to be considered: complex model projection and 3D points selection.

The tracking algorithm in our method is structured as follows:

- Projection of the detailed model with respect to the pose $\mathbf{r}_k$ computed for the previous image I_k. To achieve this process we rely on the graphics libraries (openGL) that allows to perform quickly this projection regardless the complexity of the model, thanks to the use of the GPU.
- From this projection we generate 3D measurement points $\mathbf{X}_i$ by extracting edges from the depth and texture discontinuities of the rendered model.
- We then search for corresponding edge points $\mathbf{x}'_i$ in image I_{k+1}. To allow a better robustness, we propose a multiple hypothesis version of the algorithm.
- The last step is to estimate the pose $\mathbf{r}_{k+1}$ which minimizes the errors $d(\mathbf{x}_i, \mathbf{x}'_i)$ between the points extracted from the image $\mathbf{x}'_i$ and the projection of the selected 3D points $\mathbf{x}_i(\mathbf{r}) = pr(\mathbf{X}_i, \mathbf{r})$, with the following criteria:

$$\Delta = \sum_i \rho(d(\mathbf{x}_i(\mathbf{r}), \mathbf{x}'_i)) \quad (2)$$

III. Generation of 3D measurement points

As in [18], [14], at each acquired image I_{k+1}, the model is rendered and projected using the openGL rendering engine (which takes advantage of the computer GPU), with respect to the previous pose $\mathbf{r}_k$.

Our goal is to obtain a set of 3D points $\mathbf{X}_i$ that belong to target rims, edges and visible textures from the rendered depth buffer and textured scene. Our approach follows [18] and is related to the techniques of silhouette generation of polygonal models described in [6].

A. Edge extraction using depth discontinuities

From the depth or Z-buffer, which corresponds to the depth values of the scene according to the camera location at each pixel point (Figure 1(a)), we can determine the discontinuities which suit the geometrical appearance of the scene.

Therefore, we apply a second order differential operator, such as a Laplacian filter, to these computed Z values, resulting in a binary edge map of the visible scene ((Figure 1(b)). In our approach, we have implemented the filtering computations on the GPU through shader programming, resulting in a much lower computational time, due to the parallel structure of the process.

B. Edge extraction using texture discontinuities

In case of highly textured scenes, geometrical edges are not sufficient and ambiguities with texture edges arise, which results in false matching and thus local minima during the pose estimation process. An improvement of our method is then to combine the depth discontinuities with texture discontinuities. The rendered textures of the 3D model are thus processed by a classical Canny edge algorithm and the obtained edges are added to the ones generated from the depth buffer.

C. Generation of 3D measurement points

Given the edge map of the complete scene, the 3D coordinates of the edge points in the scene can be computed thanks to the Z-buffer and the pose used to project the model. As dealing with the whole edge map can be computationally intensive, we can sample it along x and y coordinates of the image in order to keep a reasonable number of these edge measurement points.

Besides, the tracking phase (see Section IV.A) requires the orientation of the edge underlying a measurement point $\mathbf{x}_i$. For the texture edges, it is done within the Canny algorithm

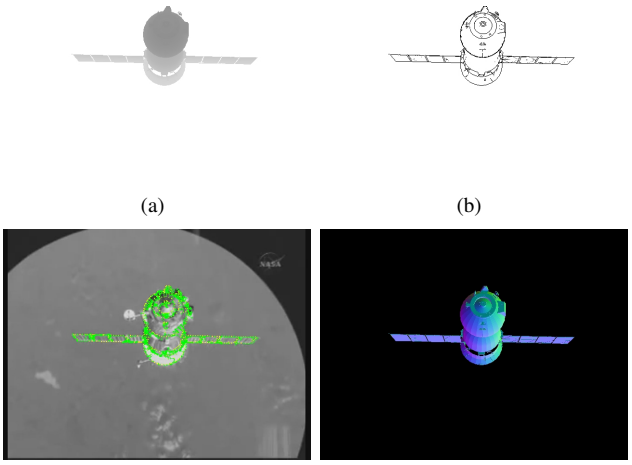


Fig. 1: On (a) is represented the z-buffer of the rendered 3D model using Ogre3D, from which the edge map is computed (b).This edge map is then sampled to extract measurement points, reprojected on the current image and from them a 1D search along the edge normal is performed to find a matching edge point in the acquired image (c). (d) shows the normal map of the scene.

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. The normal map of the scene associates the [R,G,B] value of each pixel location to the coordinates in the world frame of the normal to the corresponding surface in the scene (see Figure 1(d)). As the rendering phase can suffer from aliasing, the gray-level image of the normal map is filtered using a Gaussian kernel before computing its Sobel gradients. These basic image processing steps, as well as the retrieval of the normal map, are also processed on the GPU by composing vertex and fragment shaders, significantly optimizing computations.

IV. Pose Estimation and Low Level Tracking

A. Tracking from edge measurement points

The measurement points are then processed to track corresponding edges in the image. In a similar manner to [17], [3], [18], we perform a 1D search along the normal of the underlying edge (Figure 2 and Figure 1(c)) 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.

B. Minimization of a distance to a line

Once correspondences are established, the goal is then to estimate the new pose $\mathbf{r}_{k+1}$ that realigns the measurement points with their matching observed image points $\mathbf{x}'_i$. This task is addressed by minimizing the errors $d(\mathbf{x}_i(\mathbf{r}), \mathbf{x}'_i)$. As in [4], [3], [18], our approach considers the distance between the projected 3D line $l_i(\mathbf{r})$ underlying the projected measurement 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 (see Figure 2). Criteria (2) becomes:

$$\Delta = \sum_i \rho(d_\perp(l_i(\mathbf{r}), \mathbf{x}'_i))) \tag{3}$$

where ρ is a robust estimator used to reject outliers (Tuckey estimator), and $d_\perp(l_i(\mathbf{r}), \mathbf{x}'_i)$ is the distance between the point $\mathbf{x}'_i$ and the corresponding line $l_i(\mathbf{r})$. It has to be noted that for sharp edges the 3D point $\mathbf{X}_i(\mathbf{r})$ is not modified when we modify $\mathbf{r}$. This is no longueur the case for points $\mathbf{X}_i(\mathbf{r})$ that belong to an occlusion rim. Nevertheless, since the camera motion between two successive images is very small, this approximation has no impact on the efficiency of the approach.

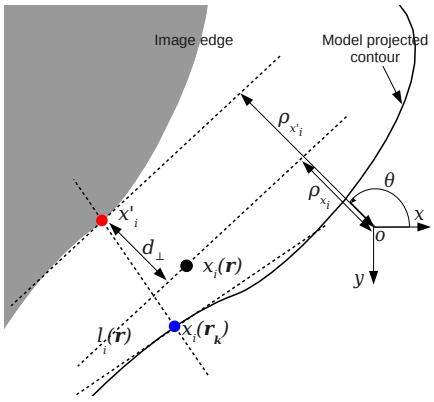


Fig. 2: Moving edge principle: from the initial pose $\mathbf{r}_k$, 1D search along the projected contour underlying the measurement point. Distance of a point $\mathbf{x}'_i$ to a corresponding line $l_i(\mathbf{r})$ within the minimization process.

The optimization technique is similar to the virtual visual servoing framework described in [3], which relates this optimization problem to a visual servoing issue. A key requirement 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. We achieve this by first expressing the polar coordinates $(\rho_{\mathbf{x}_i}, \theta_{\mathbf{x}_i})$ of $l_i(\mathbf{r}_k)$ in the image, with $\rho_{\mathbf{x}_i}$ the distance between the projected line $l_i(\mathbf{r}_k)$ and the center of the image and $\theta_{\mathbf{x}_i}$ the angle between the image frame and the line (Figure 2):

$$x \cos\theta_{\mathbf{x}_i} + y \sin\theta_{\mathbf{x}_i} = \rho_{\mathbf{x}_i}, \forall (x, y) \in l_i(\mathbf{r}_k) \tag{4}$$

From the model rendering phase, we know $\theta_{\mathbf{x}_i}$ from the gradient computations detailed in Section III.C and we can compute $\rho_{\mathbf{x}_i}$, since $\mathbf{x}_i(\mathbf{r}_k) \in l_i(\mathbf{r}_k)$. Then, thanks to the normal map (see Figure 1(d)) the equation of the model surface underlying l_i is retrieved, and it is finally straightforward to obtain the 3D equation of l_i.

l_i can thus be projected with respect to the pose $\mathbf{r}$, updating $\rho_{\mathbf{x}_i}$ and $\theta_{\mathbf{x}_i}$, and the error $d_\perp(l_i(\mathbf{r}), \mathbf{x}'_i)$, which can be derived as follows:

$$d_\perp(l_i(\mathbf{r}), \mathbf{x}'_i) = \rho_{\mathbf{x}_i} - \rho_{\mathbf{x}'_i}. \tag{5}$$

In the sense of the virtual visual servoing framework [3], the robust control law that moves the velocity skew $\mathbf{v}$ of the

virtual camera in order to minimize $d_\perp(l_i(\mathbf{r}), \mathbf{x}'_i)$ is given by:

$$\mathbf{v} = -\lambda(\mathbf{D}\mathbf{L}_{d_\perp})^+ \mathbf{D} d_\perp(l_i(\mathbf{r}), \mathbf{x}'_i) \quad (6)$$

where $\mathbf{L}^+_{d_\perp}$ is the pseudo inverse of $\mathbf{L}_{d_\perp}$, the interaction (or Jacobian) matrix of the feature $d_\perp(l_i(\mathbf{r}), \mathbf{x}'_i)$, which links $\mathbf{v}$ to the velocity of the features in the image (see [5] for its complete derivation, in the case of a distance to a line). λ is a proportional gain that ensures an exponential decrease of the error and $\mathbf{D}$ is a weighting matrix associated to the Tuckey 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 [10]:

$${}^{c_{k+1}}\mathbf{M}_o = {}^{c_{k+1}}\mathbf{M}_{c_k} {}^{c_k}\mathbf{M}_o = e^{-\mathbf{v}} {}^{c_k}\mathbf{M}_o \quad (7)$$

C. Multiple hypothesis solution

In order to improve the robustness of the pose estimation and to avoid problems due to ambiguities between edges, it is possible to consider and register different hypothesis corresponding to potential edges. They correspond to different local extrema of the gradient along the scan line. As in [17], [16], we choose the hypothesis which has the closest distance to the projected 3D line l_i during the minimization process. The cost function becomes :

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

where points $\mathbf{x}'_{i,j}$ are the selected candidates for each measurement point $\mathbf{x}_i$.

V. Experimental results

A. Implementation

The rendering process of the 3D polygonal and textured model relies on OGRE (Object-oriented Graphics Rendering Engine), which is a scene-oriented, flexible 3D rendering engine [1]. The library avoids to explicitly use the underlying system libraries (Direct3D or OpenGL). As presented in Section III, 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), supported by OGRE, which enables classic shading techniques such as composition of successive shaders. The remainder of the algorithm has been implemented thanks to the ViSP library [11]. Regarding hardware, an NVIDIA NVS 3100M graphic card has been used, along with a 2.8GHz Intel Core i7 CPU.

For tests procedures, we have performed experiments on both real and synthetic image sequences. They consist in evaluations of the algorithm described in the previous sections, for the texture and untextured cases and for the single and multiple hypothesis solutions.

B. Tests on real images

The first example deals with the tracking of the Soyuz TMA-12 spacecraft during its rendezvous phase with the International Space Station (ISS). This manned flight is intended to transport crew members of the ISS. With its three different modules, the main body of the spacecraft is of complex shape, with curved and fuzzy edges, but the solar arrays tend to facilitate the tracking. Here the tracking procedure consists in only using the depth discontinuities in the rendered scene to generate the visible and prominent edges (see Figure 1(d)), and the single hypothesis solution has shown to be sufficient for this sequence. Despite the uncertainties over the camera internal parameters, the 3D model consistency, the cluttered background and the low quality of the original video, the tracking is successfully achieved (see Figures 5(a)- 5(d) and provided video).

The second example concerns the Atlantis Space shuttle performing a pitch maneuver for its rendezvous with the ISS, for the STS-135 mission. An untextured 3D model of the spacecraft has been processed for the tracking, together with the multiple hypothesis registration process and a Kalman filter with a constant velocity model on the pose parameters. This target also presents a complex shape, with cylindrical and curved parts, such as the fuselage or the engines. Figures 5(e)- 5(h) shows the tracking properly performed over the sequence, with a robustness to some illumination changes (see provided video). The multiple hypothesis solution and the Kalman filter are necessary to handle the shuttle flip in the image (around Figure 5(g)). The estimated camera pose parameters are represented on Figure 5(i).

C. Tests on synthetic images

The evaluation has been performed using a realistic ray-tracing simulator developed by Astrium for space environments, and which enables to simulate rendezvous or approaches with spacecrafts, satellites or space debris. We focus here on the case of the Spot satellite family. Spot are earth observation satellites whose orbit is approximately polar, circular, sun-synchronous, at an altitude of around 830 kilometers, and with an inclination of 98.7 degrees (see Figure 3). 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 (Figure 4). The chaser is also equipped with a camera filming the target and a spot light to lighten the target, especially during sun eclipses.

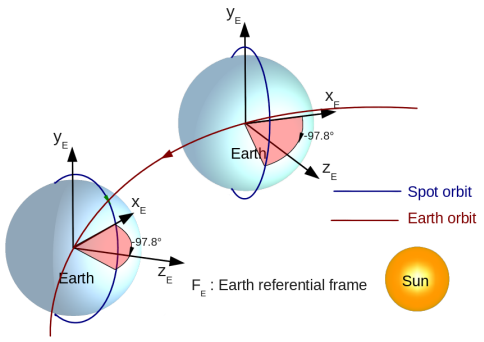
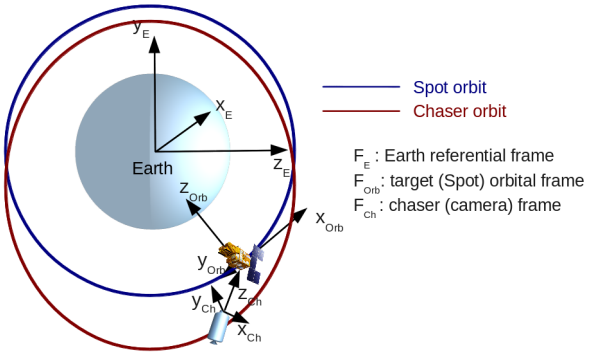


Fig. 3: Spot sun-synchronous orbit.

As it can be observed on Figure 6, the tracking presents



(a)

Fig. 4: Chaser and target (Spot) orbits in the Earth reference frame.

good performances throughout the whole sequence. In order to quantify the different results, we separately evaluate the accuracy of rotation and translation components of an estimated camera pose $\hat{\mathbf{r}}$ with respects to the true pose $\mathbf{r}^*$, as we can be provided with Ground Truth. The results for the single (SH) and multiple hypothesis (MH) approaches, relying on both depth and texture edges, and with or without Kalman filtering are represented on Figure 7. They show that when using multiple hypothesis along with the Kalman filter, translation and rotation errors can be kept small, especially when the target is far from the chaser with low luminosity (see Figure 6(d)), and when the solar panels flip in the image (see Figure 6(c)), leading to some local minima for the single hypothesis solution.

D. Computational costs

Thanks to the implementation of several image processing phases on the GPU using shaders, the execution time for these phases has been considerably reduced. With this improvement, the whole algorithm can be processed at a 17 fps framerate for the Atlantis sequence, at 15 fps for the Soyuz sequence, and at 12 fps for the Spot synthetic sequence when relying on depth edges. The multiple hypothesis solution, since we are using 3 candidates per measurement point, does not affect much computations. Including texture information makes the process costlier, since more points are considered, with a 7 fps framerate for the Spot sequence.

VI. Conclusion

This paper presents a generic tracking and pose estimation method suited for complex, textured or untextured objects in deep space environments, for space rendezvous and space debris removal purposes. It relies on the generation of visible and salient edges from the 3D complete polygonal model, projected and rendered using a rendering engine. Information from both geometrical and texture discontinuities are used. This technique avoids the heavy processing of 3D lines out of a 3D line model, which is problematic when dealing with curved forms and computationally costly. For pose estimation, the method is similar to the classical approaches as it consists in the realignment of the generated model edges with edges detected in the image.

From the tests carried out on real images, our approach qualitatively presents satisfactory results. Thanks to the realistic image simulator, performances can be measured, showing the relevance of the approach. The implementation proves to be computationally efficient, especially through the processing of some phases on the GPU. As ambiguities between edges can occur in detailed scenes, leading to local minima and unstable tracking when using a single registration technique, considering multiple hypothesis improves the robustness of our approach.

Further works would aim at implementing a detection strategy in order to initialize the frame-by-frame tracking. It would also rely on the 3D model of the target.

References

[1] Ogre3d, open source 3d graphics engine. www.ogre3d.org.

[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] 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.

[4] 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.

[5] 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.

[6] T. Isenberg, B Freudenberg, S. Schlechtweg, and T. Strothotte. A developer guide to silhouette algorithms for polygonal models. *IEEE Comput. Graph. Appl.*, 23(4):28–37, 2003.

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

[8] J.J. Koenderink. Solid shape. MIT Press, 1990.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] G. Reitmayr and T. Drummond. Going out: robust model-based tracking for outdoor augmented reality. In *IEEE/ACM Int. Symp. on Mixed and Augmented Reality, ISMAR'2006*, pages 109–118, Santa Barbara, CA, October 2006.

[15] 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.

[16] 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.

[17] 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.

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

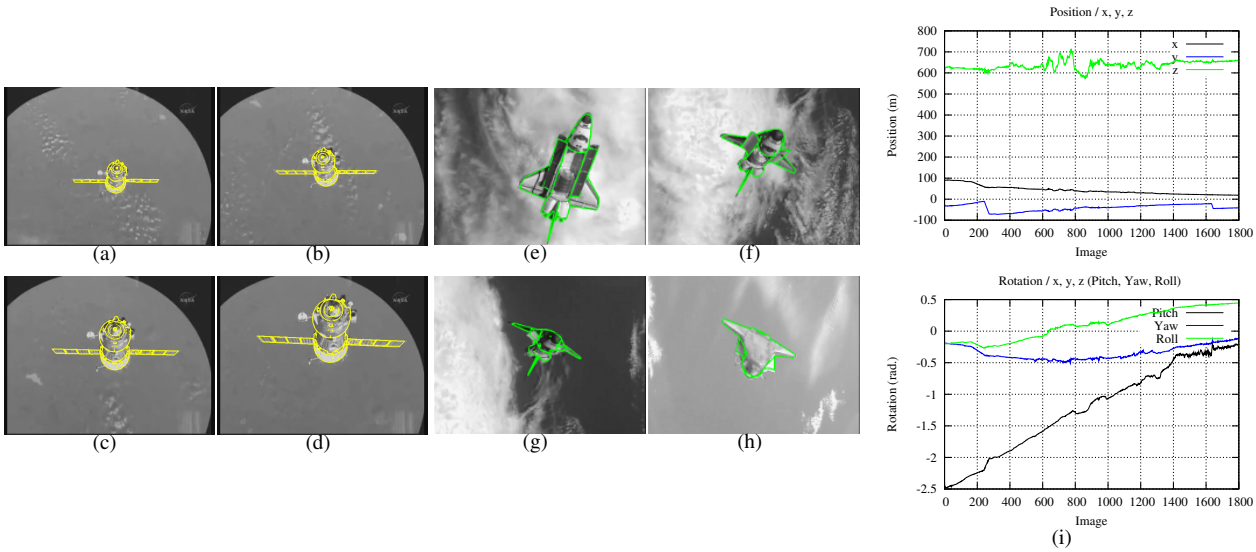
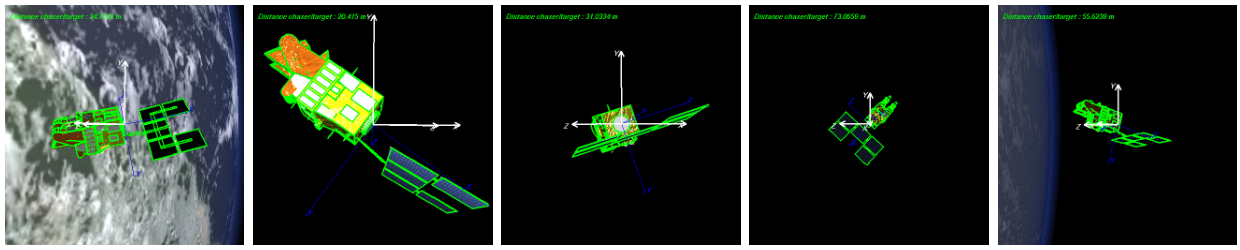


Fig. 5: Tracking relying on depth edges for the Soyuz ((a)-(d)) and Atlantis sequences ((e)-(h)). Pose parameters for the Atlantis sequence are represented on (i).



(a) Image 1 (b) Image 599 (c) Image 754 (d) Image 1099 (e) Image 1439

Fig. 6: Tracking relying on depth and texture edges, with multiple hypothesis and Kalman filtering. The tracking is properly performed throughout the sequence.

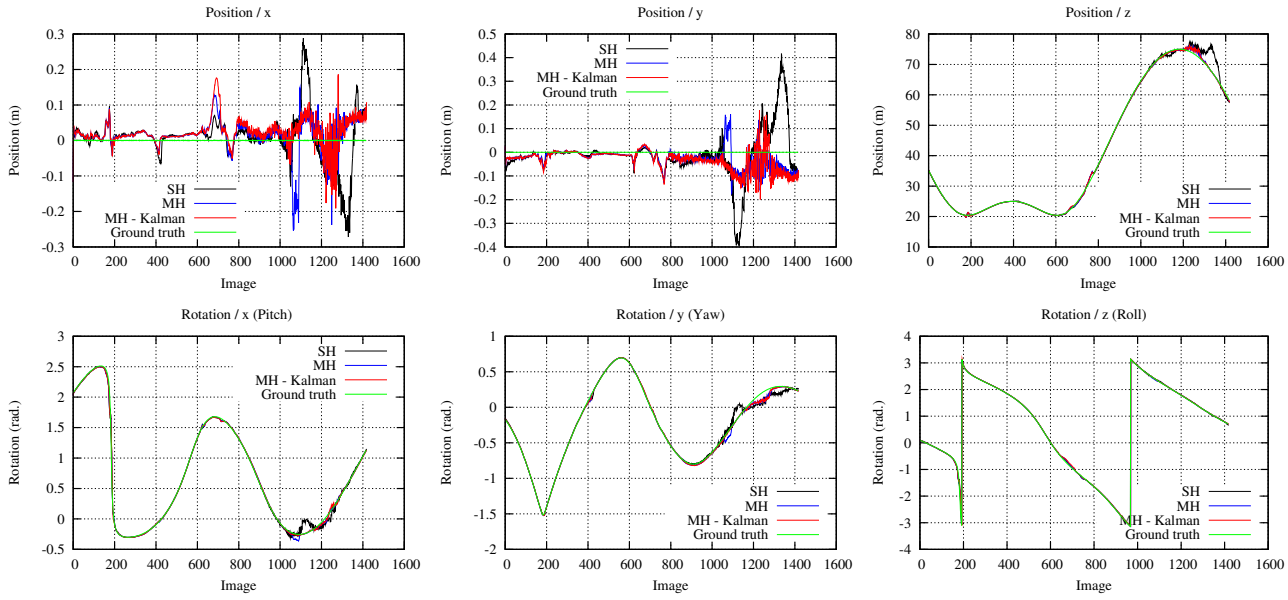


Fig. 7: Estimated camera pose parameters of the target over all the sequence, along with the ground truth, for the single and multiple hypothesis solutions, together with a Kalman filter or not.