SATELLITE VISUAL TRACKING FOR PROXIMITY OPERATIONS IN SPACE

Manolis Lourakis and Xenophon Zabulis

Institute of Computer Science
Foundation for Research and Technology - Hellas (FORTH)
P.O.Box 1385, GR-711 10, Heraklion, Crete, Greece

ABSTRACT

Determining the pose of orbiting satellites is a fundamental prerequisite for supporting autonomous proximity operations in space. This work presents a model-based 3D tracking algorithm based on edges. The proposed tracker maintains a pose hypothesis that is propagated from frame to frame, using it first to render a depth image and then refining it according to partial matches established between depth and intensity edges. Edge matching relies on fast, local linear searches along the depth gradient direction. The tracker does not require any preprocessing of the 3D model nor does it make any assumptions regarding its characteristics, as is often the case for other approaches based on edges. It is also robust to parts of the tracked satellite being out of view, occluded, shadowed or visually undetected. Experimental results evaluating the accuracy of the tracker and comparing it against established techniques are also included.

Key words: Pose estimation; proximity navigation; ADR.

1. INTRODUCTION

Autonomous proximity operations in space refer to the ability of an unmanned satellite to conduct fine maneuvers close to other orbiting satellites with the purpose of rendezvous, inspection, assembly, re-supply, servicing or docking [16]. Of particular interest are active debris removal (ADR) missions, which aim to stabilize Earth's orbit space (esp. LEO) by deorbiting large non-functional satellites and launch vehicles [4]. A key enabling technology for such missions concerns automatic techniques for estimating the relative pose between a service and a target spacecraft [14]. Relative pose refers to the six position and attitude (orientation) parameters defining the geometric relationship between the two spacecraft. Estimates of relative pose and corresponding linear and angular velocities are to be used in closed-loop orbit control to guide the motion of the service spacecraft in order to safely approach and later navigate away from the target.

Existing techniques for pose estimation are characterized by the type of sensor they employ, their operating range and by whether the target spacecraft is cooperative, i.e. carries a set of easily detectable artificial markers (either retro-reflectors or visual fiducials) forming a known pattern that is tracked in order to achieve the estimation of pose. These classification criteria are intertwined since the type of sensor relates to the applied sensory processing algorithms and the required operating range determines the applicable sensor choices. Active LiDAR range-finders [9, 21] can provide accurate, high resolution distance measurements but have the drawbacks of substantial mass, power consumption and cost. They may also have low acquisition speed and angular resolution. On the other hand, infrared and visible light cameras are particularly attractive for estimating pose in space, owing to their compact size, passive mode of operation, simple hardware and low power consumption. Early space-demonstrated systems such as the Space Vision System (SVS) [29], the Trajectory Control System (TCS) [10] or the Advanced Video Guidance Sensor (AVGS) [22] operated on the assumption that the target spacecraft is cooperative. As such operational constraints are not acceptable in several important proximity navigation scenarios, for example the removal of space debris or the refueling of spacecraft in orbit, there is currently a trend towards systems that are appropriate for un-cooperative targets.

This paper reports our efforts in the context of ESA project HIPNOS (HIgh Performance avionics solutioN for advanced and cOmplex GNC Systems) [25], which studies computer vision algorithms and avionics architectures suitable for ADR. Our work focuses on developing vision-based algorithms for relative pose estimation that are amenable to hardware acceleration. Specifically, we are concerned with the problem of using a single passive camera to track the pose of an un-cooperative orbiting spacecraft at medium and close ranges, i.e. at distances below 100 m. Visual tracking in orbit must overcome challenges such as rapidly changing and often inadequate illumination, glare, hard shadows, tumbling targets, lack of strong texture, occasional background clutter, low capacity flight hardware, etc.

We propose a model-based, monocular tracking technique which relies on edges to yield accurate pose estimates with moderate computational overhead. Increased frame rate performance will be achieved by partitioning tracking on a FPGA/CPU system. The rest of the paper

is organized as follows. An overview of related work is provided in Sec. 2. The proposed tracking algorithm is presented in Section 3 and is compared with established methods in Section 4 using synthetically generated image sequences. The paper concludes in Section 5.

2. PREVIOUS WORK

Edges are defined by sharp changes in image intensities which originate from discontinuities in surface orientation, texture, depth or illumination. They are moderately robust against noise and illumination or viewpoint changes. Edges are also accurately and rapidly localized in images and can be extracted even from weakly textured objects, to which techniques relying on photometric local patch detectors and descriptors, e.g. [26], cannot be applied. Consequently, edges have been used as the primary feature type by several object tracking algorithms.

For instance, Lowe [28, 27] tracks an object by extracting line segments from its image contours and fitting them to a known object model. D'Amico et al. [12] adapt Lowe's approach to a spaceborne monocular vision-based navigation system. The RAPiD tracker proposed by Harris [18] operates in a reverse direction by first projecting the model using an approximate pose and then matching it with image edges. RAPiD relies on the assumption that the difference between the actual pose and its predicted estimate is small. Therefore, data association can be efficiently established using 1D local search for an image edge along the direction that is perpendicular to a predicted edge. To keep the computational complexity low, perpendicular matching is limited to a sparse set of predetermined control points. Control points are thus employed to sparsely measure the component of image motion that is perpendicular to an edge. Linearization about the current pose estimate allows each pair of orthogonally matched points to yield a linear constraint on the six parameters defining the incremental change in object pose. RAPiD is very efficient and was historically the first 3D tracker to successfully run at high frame rates on general purpose hardware. As a result, its paradigm of using a local search around a prior pose has been retained in subsequent trackers. Despite its effectiveness, however, the basic RAPiD algorithm lacks robustness to mismatches and occlusions and requires that control points and their visibility are provided externally.

Several authors have proposed improvements to the basic RAPiD algorithm. Armstrong and Zisserman [2], for example, address robustness by grouping control points using geometric primitives such as lines and conics and use RANSAC to identify and discard incorrect edge matches. Drummond and Cipolla [13] use a Lie group formalism to represent the linearized relationship between image motion and pose parameters. They define control points on object lines, the visibility of which is determined at runtime with binary space partition trees. Robustness is also attained by employing an M-estimator computed with iteratively reweighted least squares (IRLS) to estimate the pose parameters. Comport et al. [11] treat pose computation as the dual problem of 2D visual servoing and track points normal to the projections of object lines. M-estimation with IRLS is again used to obtain robustness.

A common thread in the works of [2, 13, 11] is that they assume simplified object models, in which all modeled edges must give rise to visible image edges that are sampled for defining the control points. Thus, models suitable for tracking must retain only the most prominent edges and should be primarily comprised of line segments. Such requirements are typically not met by CAD models represented by densely tessellated polygonal 3D meshes (cf. Fig. 2), clearly limiting the flexibility of the aforementioned tracking algorithms. To alleviate this, more recent approaches leverage the processing capabilities of modern GPUs in order to dynamically identify the edges of a model that are visible from a particular viewpoint. For example, Reinke et al. [32] propose a technique for hidden line removal which relies on rendering a CAD model to automatically extract visible edges that are used for tracking. Additionally, they recommend a random distance sampling strategy for defining the control points on visible edges. Petit et al. [31] also rely on rendering an object model to determine edge visibility but avoid any model line processing by extracting edges from discontinuities of the rendered depth image.

A shortcoming of purely edge-based methods is that unrelated edges might locally look very similar and thus can give rise to erroneous edge correspondences that will cause tracking to fail. To deal with this, certain works suggest maintaining multiple pose hypotheses, e.g. [23, 24, 35, 8]. In particular, we note that reference [8] which will latter be used for comparative evaluation, performs tracking in a particle filtering framework, using chamfer matching to form pose hypotheses and initialize particles that are subsequently evolved with standard edge-based tracking. Aiming to further increase robustness and reduce drift, techniques such as [36, 33, 7] which combine edge and point features have also been proposed. These define keyframes that are used for anchoring with the aid of point features.

This work puts forward a RAPiD-like tracking algorithm that can accommodate any triangle mesh model without pre-processing or manual intervention for determining the control points. This is achieved by using the object model in combination with rasterization rendering to produce a depth image. Rendering automatically handles self-occlusions, thus permitting control points to be defined on edges extracted from the depth image. The 6D pose is estimated by maintaining a single hypothesis which is evolved from frame to frame using robust regression techniques. Our approach is delineated next.

3. PROPOSED METHOD

A triangle mesh model for the tracked object and its pose in the initial frame are assumed to be externally provided.

At each subsequent frame, an approximate object pose, which might simply be the one estimated for the previous frame, is also assumed to be available. For each incoming image, Canny's algorithm [6] detects its strongest edges. Independently, the object model is projected on the image at the approximate pose, using rasterization rendering. Rendering relies on the approximate pose to yield a depth image (Sec. 3.1). Edges are then detected in the depth image and used to determine the control points by local, 1D searches that start at depth edge pixels and extend in directions parallel to the depth gradients. Specifically, a control point is defined for every match established between depth and intensity edges (Sec. 3.2). Each control point provides a linear constraint on the incremental change in pose (Sec. 3.3). The set of all such linear constraints is used in a robust regression framework which estimates pose after removing the influence of outliers (Sec. 3.4). This pose is used to repeat the pose estimation process for a new intensity frame.

The contributions of the method are threefold. First, control points are defined automatically by rendering unrestricted mesh models, without any manual preprocessing, and partially matching edges of different modalities. Second, outliers are handled with a combination of robust regression techniques that do not call for the definition of arbitrary outlier thresholds. Third, perpendicular matching is made more robust by maintaining multiple candidate matches for each control point. More details are provided in the following subsections.

3.1. Rendering

Given a triangle mesh model of an object and a camera pose, depth rendering simulates an image whose pixel values are distances rather than intensities. More specifically, every pixel in the rendered depth image contains the distance to the nearest point on the model's surface that projects on the pixel in question. Depth images are rendered with the aid of rasterization rendering [17]. Rasterization is essentially a technique to solve the visibility problem, which consists of being able to determine which parts of an object are visible to the camera, excluding those parts that are outside the field of view or are hidden due to self-occlusion. Rasterization projects mesh triangles onto the image by projecting their vertices, determining all image pixels that are covered by the projected triangle and computing the distance of the 3D triangle from each such pixel. To deal with the case of multiple triangles projecting on the same pixel, Z-buffering is employed to retain the projection of the triangle closest to the camera. Rasterization involves only geometry (i.e., no texturing/shading) and has been entirely implemented in software using the Möller-Trumbore algorithm for calculating the intersection of a ray and a triangle. Nevertheless, rendering can be hardware accelerated due to its high data parallelism.

3.2. Selection of Control Points

Most of the existing edge-based approaches, e.g. [2, 13, 11], require that the object to be tracked is modeled with a simple wireframe that consists of a small number of straight edges. In effect, the object's image contours should be described by piecewise linear segments. Control points are defined by sampling either image [2, 13] or model [11] edges. While such a choice ensures computational efficiency, it imposes strong constraints on acceptable models, thus limiting applicability. This is because detailed CAD object models that might be available need to be manually redesigned so that they are made suitable for tracking by retaining only their most salient edges. Furthermore, object models often include curved parts such as spheres or cylinders, which are not faithfully represented with straight edges. To deal with arbitrary 3D meshes, [7, 32] use wireframe rendering to determine which model edges are visible from a particular viewpoint, whereas [31] uses full depth rendering of the model and finds edges with the Laplacian operator as discontinuities in the depth image. In both cases, rendering requires a predicted object pose, which might simply be the object pose in the previous frame. In this work, we follow an approach similar to [31], but employ a more reliable alternative to detecting depth edges.

The Laplacian edge detector looks for zero crossings of the depth's second derivative, i.e. pixels where the Laplacian changes sign. As this produces many spurious edges, in practice edges are detected at pixels which in addition to being adjacent to zero crossings, have absolute Laplacian magnitudes differing by more than a threshold. However, absolute thresholding makes the detection of depth edges sensitive to the choice of threshold. In this work, we have employed [5], which primarily detects occluding (i.e., jump) edges. The algorithm operates in two passes, examining depth image pixels along rows and then columns. At each pass, it detects depth edges by comparing the difference in the depth values of neighboring pixels against a proportional threshold. The edge detector of [5] was further adapted to detect crease edges based on the magnitude of the contained angle of the normal vectors corresponding to neighboring depth pixels. More specifically, a crease edge is detected whenever the absolute value of the dot product between two neighboring normal vectors is less than a threshold. It is noted that this calculation is independent of the coordinate system used to express the normal vectors, which are estimated once when an object model is loaded. The jump and crease edgels resulting from the two applications of [5] are then merged into a single depth edge map. Control points are defined on the detected depth edge pixels, after matching them with intensity edge pixels. This process makes no assumptions about the shape of the object's contour and is explained next.

Given two edge maps and their corresponding gradient orientations quantized in eight 45° wide bins, edge matching concerns the establishment of perpendicular correspondences between edge pixels. This is achieved

by examining each edge pixel in the source (i.e., depth) edge map and moving along the gradient direction in the target (i.e., intensity) edge map, until either an edge pixel is found or a maximum distance from the starting pixel has been traced (see also Fig.1). To declare a match, the edge pixel found in the target map has to have a gradient orientation compatible with that of the source pixel (cf. edge polarity [18]). In accordance with the aperture problem, this procedure determines a partial rather than a full match, since local analysis can only provide the displacement component perpendicular to the edge. The search for a perpendicular match along the edge normal has linear rather than quadratic complexity, this being a crucial enabling factor for efficient performance. The search for corresponding edge pixels should be performed in both opposite orientations, as it is not possible to know in advance which side of the source edge the target edge has moved. In the case that matching candidates are found for both orientations, the one closest to the source pixel is retained. In all cases, the visited target pixels are determined with Bresenham's line drawing algorithm which involves integer coordinates only.

As described up to this point, edge matching seeks for the closest edge pixel in the target map along the gradient direction, similarly to previous edge-based trackers like [18, 2, 13, 11]. This can be improved by considering multiple candidate matches for each control point, as follows. During matching, all edge pixels from the target map along the gradient and up to a maximum distance are retained. A pose estimate is first computed by relying on the closest candidate match for each control point. The control points are then backprojected to the model using the depth image and then reprojected on the image with the estimated pose. The candidate match closest to each reprojected control point is then found and pose estimation is repeated, using the revised matches. After a new pose has been estimated, the process can be repeated. In practice, we have observed that one or two iterations are sufficient for the assignment of matches to stabilize.

3.3. Pose Constraints

This section relies on findings from [18]. Let $\mathcal{SE}(3)$ denote the special Euclidean group comprised of the six-parameter family of proper rotations and translations in the 3D Cartesian space. Assume an object coordinate system with its axes aligned with those of the camera coordinate system and its origin at $\mathbf{T} = (T_x, T_y, T_z)$ in camera coordinates. A control point's camera coordinates $\mathbf{M}$ and its object coordinates $\mathbf{P} = (P_x, P_y, P_z)$ are related with $\mathbf{M} = \mathbf{T} + \mathbf{P}$. Let $\mathbf{m}$ be this point's normalized image projection.[1] Note that the camera coordinates $\mathbf{M}$ can be determined by backprojecting $\mathbf{m}$ with the aid of the rendered depth image. Assume now that the object moves with a rigid motion $\xi \in \mathcal{SE}(3)$, consisting of an incremental rotation $\mathbf{R}$ and translation $\mathbf{t}$ which express the object's pose in the next camera frame with respect to the current one. The new camera coordinates of the control point are thus given by $\mathbf{M}' = \mathbf{RP} + \mathbf{T} + \mathbf{t}$. Assuming that motion ξ is infinitesimal, $\mathbf{R}$ can be approximated as $\mathbf{I} + [\omega]_\times$ with $\mathbf{I}$ being the 3×3 identity matrix and $[\omega]_\times$ the skew-symmetric matrix associated with the cross product:

$$[\omega]_\times = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \quad (1)$$

Then, $\mathbf{M}'$ can be approximated as

$$\mathbf{M}' \approx [\omega]_\times \mathbf{P} + \mathbf{M} + \mathbf{t}. \quad (2)$$

The normalized projection $\mathbf{m}'$ of $\mathbf{M}'$ on the image can be obtained by algebraically manipulating the right side of Eq. (2) and dividing the first two elements of the resulting vector expression with the third. Multiplying the numerators and denominators by $M_z - (\omega_x P_y - \omega_y P_x + t_z)$, expanding and ignoring second order terms, yields

$$\begin{array}{l} m'_x = m_x + (t_x + \omega_y P_z - \omega_z P_y - m_x(\omega_x P_y - \omega_y P_x + t_z))/(T_z + P_z) \\ m'_y = m_y + (t_y + \omega_z P_x - \omega_x P_z - m_y(\omega_x P_y - \omega_y P_x + t_z))/(T_z + P_z). \end{array} \quad (3)$$

The above expression is linear in the elements of $\xi = (\omega_x, \omega_y, \omega_z, t_x, t_y, t_z)^t$, therefore it can be written more compactly as

$$\mathbf{m}' = \mathbf{m} + \mathbf{A}\xi, \quad (4)$$

with $\mathbf{A}$ a 2×6 matrix defined by the coefficients of the elements of ξ from Eq. (3).

As shown in Fig. 1, let $\mathbf{n}$ be the unit vector perpendicular to the edge at $\mathbf{m}$. The distance between points $\mathbf{m}$ and $\mathbf{m}'$ in the direction of $\mathbf{n}$ is given by the projection of $\mathbf{m}' - \mathbf{m}$ on $\mathbf{n}$, i.e. $\mathbf{n}^t(\mathbf{m}' - \mathbf{m})$, and is approximately equal to d. Combined with Eq. (4), this yields $\mathbf{n}^t\mathbf{A}\xi = d$. Simply put, every control point provides one constraint on the motion ξ, therefore six such points determined by searching along different directions suffice to estimate ξ.

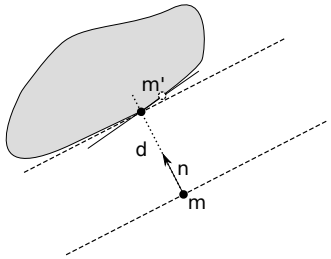

Figure 1. Illustration of perpendicular edge matching. Point $\mathbf{m}$ *is the predicted projection of a control point and* $\mathbf{m}'$ *the actual projection of this control point in the next frame. The edge through* $\mathbf{m}'$ *is approximately parallel to that through* $\mathbf{m}$ *(solid and dashed lines, resp.). Due to the aperture problem, the location of* $\mathbf{m}'$ *cannot be fully determined, so a linear search along* $\mathbf{n}$ *yields distance* d*.*

[1] A normalized image projection refers to a projection on an ideal pinhole camera, i.e. if $\mathbf{M} = (X, Y, Z)$ then $\mathbf{m} = (\frac{X}{Z}, \frac{Y}{Z})$. In other words, the effects of the camera intrinsics $\mathbf{K}$ on the projection have been removed.

If more than six constraints are available, ξ can be estimated in a least squares manner by solving

$$\hat{\xi} = \arg\min_{\xi} \sum_{i} (\mathbf{n}_i^t \mathbf{A}_i \xi - d_i)^2. \quad (5)$$

Finally, $\hat{\xi}$ is used to incrementally update the pose. For future reference, the differences $\mathbf{n}_i^t \mathbf{A}_i \xi - d_i$ between the observations and their fitted values are called residuals.

3.4. Robustification of Pose Estimation

The linearization about the current pose estimate detailed in the previous section allows each pair of orthogonally matched points to yield one linear constraint in the six parameters defining the incremental change in object pose. In practice, more than six matches will be established and some of them will be erroneous due to various sources of error. Erroneous matches will give rise to outlying constraints with large residuals, which will cause least squares to fail due to its lack of robustness. Therefore, despite its simplicity, Eq. (5) cannot be used directly. Instead, the pose refinement computation is carried out in a robust regression framework, which allows problematic measurements to be identified and discarded before corrupting the estimate. Robustness is achieved by employing the Least Median of Squares (LMedS) estimator [34], which substitutes the summation operation in the minimization of the squared residuals in Eq. (5) with the median:

$$\hat{\xi} = \arg\min_{\xi} \operatorname{med}_{i} (\mathbf{n}_i^t \mathbf{A}_i \xi - d_i)^2. \quad (6)$$

LMedS can tolerate up to 50% erroneous constraints, but unlike least squares, has no closed form solution. Thus, (6) is minimized by drawing a sufficient number of random sets of six constraints, estimating ξ from each and retaining the one corresponding to the minimum median residual. Despite its robustness, LMedS suffers from low precision. Therefore, the LMedS estimate of ξ is not used as such. Instead, an extra computation step uses it to identify as outliers those constraints that yield large residuals, then discards them and estimates ξ using least squares on the remaining inliers. RANSAC [15] could also be used in place of LMedS to achieve robustness. However, to discriminate between inliers and outliers, RANSAC requires that the standard deviation of valid residuals is externally supplied. LMedS, on the other hand, estimates this deviation as a byproduct, thus automatically adapts to the level of noise present in the constraints. We note that the median is computed with Quickselect [20], which avoids sorting and has an average complexity of $O(n)$.

An extra level of robustness is achieved by substituting the squared L_2 norm in the least squares minimization applied to the inliers with the p-th power of the L_p norm, $p < 2$:

$$\hat{\xi} = \arg\min_{\xi} \sum_{i} (\mathbf{n}_i^t \mathbf{A}_i \xi - d_i)^p. \quad (7)$$

The effect of the L_p norm is to down-weight large residuals by replacing their squares with their p-th power, which increase less steeply than quadratically. A value of 1.5 is chosen for p and the minimization in Eq. (7) is carried out with the IRLS algorithm after converting it to weighted least squares [3]. IRLS iterates over a set of weighted least squares problems, alternating between estimating the pose parameters and the weights, until convergence. We stress that IRLS is applied to the inliers of the LMedS algorithm and not to the set of all available constraints.

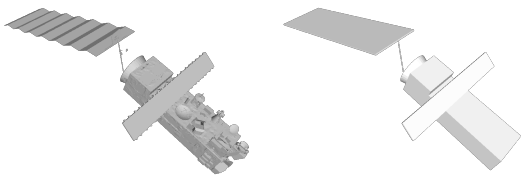

Figure 2. 3D models of ENVISAT used for tracking: detailed model made up of 30K faces (left) and simplified model with 1K faces (right).

To achieve more stable tracking and tolerate larger object motions, pose estimation is integrated with a linear Kalman filter with a constant velocity model. The filter is defined by six measurements and twelve state parameters. The measurements are the 6D pose estimated by tracking, whereas the state consists of the pose plus the linear and angular velocities. To attain a linear motion model, rotations are represented using Euler angles. At each iteration, the Kalman filter is used to predict the pose of the subsequent frame and drive rendering.

4. EXPERIMENTAL RESULTS

Results from a C implementation of the proposed tracker are presented in this section. The experiments rely on synthetic image sequences depicting ENVISAT, a defunct Earth observation satellite measuring 26 m $\times$ 10 m $\times$ 5 m (see Figs. 2 & 3). The sequences were rendered using the high detail mesh model shown on the left in Fig. 2, which consists of approximately 30K faces. They realistically simulate the motion of the satellite and the solar illumination, consisting of high resolution 1024×1024 images with a field of view equal to 40°. Each image is accompanied by accurate ground truth pose, facilitating the quantitative measurement of a tracker's performance. Sample images from the two ENVISAT sequences are shown in Fig. 3. A common characteristic of both test sequences is that they exhibit poor contrast due to insufficient illumination (cf. Fig. 3). To mitigate this, a low-complexity algorithm was used to enhance their contrast [1]. This pre-processing is applied prior to applying all tracking techniques compared below.

The metric used for measuring pose error concerns the average misalignment between the model's vertices at the true and estimated pose, as proposed in [19]. More

specifically, for the true pose $\{\mathbf{R}_g, \mathbf{t}_g\}$ and an estimate $\{\mathbf{R}_e, \mathbf{t}_e\}$, the alignment error is defined as

$$E = \frac{1}{N} \sum_{i=1}^{N} \|(\mathbf{R}_g x_i + \mathbf{t}_g) - (\mathbf{R}_e x_i + \mathbf{t}_e)\|_2, \quad (8)$$

where x_i denote the N mesh model vertices.

The proposed tracker was tested on the two ENVISAT sequences of Fig. 3. Furthermore, two recent model-based trackers were also applied to these sequences for comparison. Specifically, the first is the edge-based tracker of the ViSP library[2] [30] that corresponds to the implementation of [11] by its authors. The other is the EBT library[3], i.e. Choi's own implementation of the multi-hypotheses tracker [8]. Whereas the proposed method can employ the 3D model on the left in Fig. 2 directly, this is not the case for the method of [11]. As already pointed out, this is because [11] makes strong assumptions about the type of mesh edges that are permissible in the model of the object to be tracked, and assumes that the latter consists only of edges visible in the images. Thus, to enable a comparison, the simplified model on the right in Fig. 2 that conforms to the requirements of [11] was designed. On the other hand, the EBT tracker from [8] has no such restrictions and can work with the unmodified 30K faces detailed model.

The first sequence, a frame of which is shown on the top of Fig. 3, consists of 1152 frames. ENVISAT rotates out of plane and remains approximately at a constant distance from the camera, equal to about 55 meters. The top graph in Fig. 4 plots for each frame the alignment errors computed with Eq. (8) for the poses estimated by [11] with the simplified, 1K faces model and by [8] and the proposed method with the 30K faces model. The proposed method yields lower pose errors compared to both [11] and [8], and succeeds in tracking the target along the entire sequence. On the contrary, [11] loses track around frame 900 and [8] after 220, and yield much larger errors before they fail.

In the second sequence (cf. Fig. 3 bottom), which is 1735 frames long, the camera is initially at a distance of around 30 m from ENVISAT. While the satellite rotates, the camera gradually approaches it up to approximately 11 m. Owing to its large size, only some parts of ENVISAT are within the camera field of view in this range of distances. This sequence presents more challenging illumination which results in low contrast images, even after the enhancement by [1]. Tracked pose errors are shown in the bottom graph of Fig. 4, from which can be seen that the tracker of [11] is more accurate than the proposed one for frames between 100 and 400 but gradually becomes worse and fails completely after around frame 750. The EBT tracker of [8] produces sharper spikes and fails after frame 780.

Table 1 summarizes the average pose alignment errors for each algorithm, object model and sequence. The errors of the proposed tracker when employing the simplified model are also included for comparison. Expectedly, they are larger than those for the detailed model, since in this case, the control points determined by the proposed method do not always align well with the detected intensity edges, even for the correct poses. Still, the proposed tracker using the 1K faces model manages to track more frames with pose errors comparable to both [11, 8].

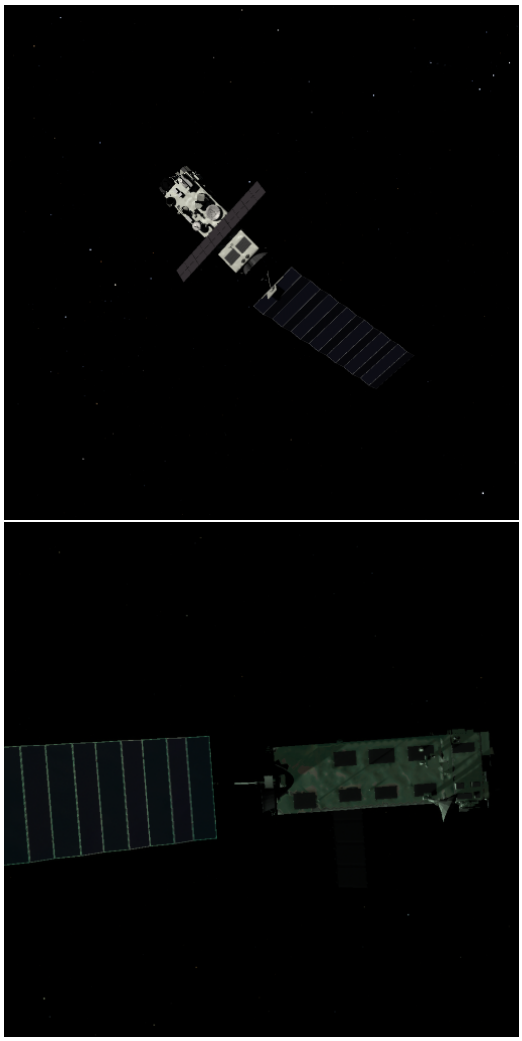

Figure 3. Initial frames from the far (top) and close (bottom) tracked sequences; notice their poor contrast.

Concerning execution speed, this depends on the apparent size of the tracked object in the images. Thus, the proposed tracker runs at about 10Hz on an Intel Core i7 CPU @ 3.60GHz for the far sequence and about 5Hz for the close one. These frame rates include the time spent loading compressed PNG images from disk and do not benefit from any hardware acceleration. Frame rate on limited capacity flight processors will be much lower, therefore ongoing work performs HW/SW co-design to accelerate the proposed tracker using space-grade FPGAs [25].

[2] https://visp.inria.fr/

[3] https://github.com/CognitiveRobotics/object_tracking_2D

Algo & model	Far sequence	#frames	Close sequence	#frames
ViSP 1K	56.11 (54.62)	901	19.63 (8.93)	752
EBT 30K	50.22 (77.64)	221	50.19 (81.60)	919
Ours 30K	16.74 (11.83)	1150	18.21 (4.94)	1735
Ours 1K	51.76 (44.32)	898	37.16 (31.09)	1598

Table 1. Average pose errors and standard deviations in cm along with tracking durations in frames for each sequence, object model and tracking algorithm. Errors were computed from the frames successfully tracked, different in each case.

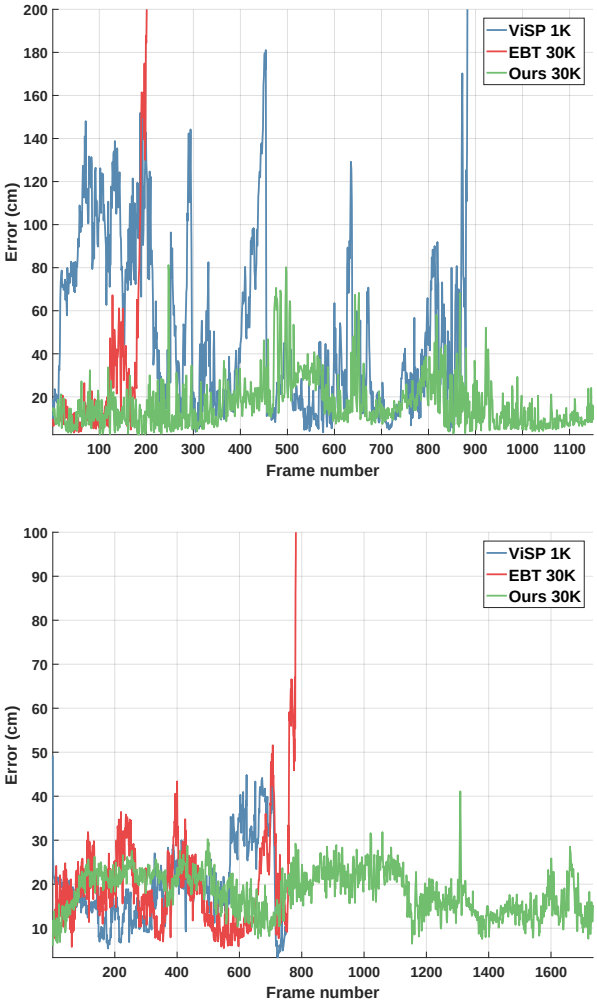

Figure 4. Errors for the poses estimated with ViSP [11], EBT [8] and the proposed tracker for the far (top) and close (bottom) ENVISAT sequences (cf. Fig. 3). The red and green graphs correspond to poses estimated by [8] and the proposed method using the detailed model from Fig. 2, while the blue graphs correspond to poses estimated with [11] using the simplified model from Fig. 2.

5. CONCLUSION

Relative localization of orbiting satellites is an important component of future autonomous proximity operations in space. Towards this goal, this work has suggested an approach for model-based tracking that relies on image edges. The developed tracker does not require any pre-processing of the 3D object model and makes no assumptions about its nature. It is also robust to common tracking nuisances, such as parts of the tracked object being out of view, shadowed, occluded or completely undetected. Tracking accuracy has been evaluated with the aid of simulated image sequences with known ground truth poses and has been shown to compare favorably to established tracking techniques. Future work will address the issue of bootstrapping the tracker by providing reliable estimates of the object's initial pose.

ACKNOWLEDGMENTS

This work was partially supported by the ESA "High Performance Avionics Solution for Advanced and Complex GNC Systems" (HIPNOS) study (ESA/ESTEC reference 4000117700/16/NL/LF).

REFERENCES

[1] Arici, T., Dikbas, S., and Altunbasak, Y. (2009). A histogram modification framework and its application for image contrast enhancement. *IEEE Trans. Image Processing*, 18(9):1921–1935.

[2] Armstrong, M. and Zisserman, A. (1995). Robust object tracking. In *Asian Conf. on Computer Vision*, volume I, pages 58–61.

[3] Bjorck, A. (2015). *Numerical Methods in Matrix Computations*. Texts in Applied Mathematics. Springer.

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

[5] Bose, L. and Richards, A. (2016). Fast depth edge detection and edge based RGB-D SLAM. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1323–1330.

[6] Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698.

[7] Choi, C. and Christensen, H. I. (2010). Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In *Proc. IEEE*

Intl. Conf. on Robotics and Automation (ICRA), pages 4048–4055.

[8] Choi, C. and Christensen, H. I. (2012). 3D textureless object detection and tracking: An edge-based approach. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3877–3884.

[9] Christian, J. and Cryan, S. (2013). A survey of LIDAR technology and its use in spacecraft relative navigation. In *AIAA Guidance, Navigation, and Control Conference*, Boston, MA.

[10] Clark, F., Spehar, P., Brazzel, J., and Hinkel, H. (2003). Laser-based relative navigation and guidance for space shuttle proximity operations. In *Advances in the Astronautical Sciences*, volume 113, pages 171–186.

[11] Comport, A. I., Marchand, É., Pressigout, M., and Chaumette, F. (2006). Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Trans. Vis. Comput. Graph.*, 12(4):615–628.

[12] D'Amico, S., Benn, M., and Jørgensen, J. (2014). Pose estimation of an uncooperative spacecraft from actual space imagery. *International Journal of Space Science and Engineering*, 2(2):171–189.

[13] Drummond, T. and Cipolla, R. (2002). Real-time visual tracking of complex structures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):932–946.

[14] English, C., Okouneva, G., Saint-Cyr, P., Choudhuri, A., and Luu, T. (2011). Real-time dynamic pose estimation systems in space: Lessons learned for system design and performance evaluation. *Intl. Journal of Intelligent Control and Systems*, 16(2):79–96.

[15] Fischler, M. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

[16] Flores-Abad, A., Ma, O., Pham, K., and Ulrich, S. (2014). A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences*, 68:1–26.

[17] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. (1990). *Computer Graphics: Principles and Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition.

[18] Harris, C. (1992). Tracking with rigid objects. In Blake, A. and Yuille, A., editors, *Active Vision*, pages 59–73. MIT Press, Cambridge, MA.

[19] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2012). Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conf. on Computer Vision*, pages 548–562. Springer.

[20] Hoare, C. A. R. (1961). Algorithm 65: Find. *Commun. ACM*, 4(7):321–322.

[21] Horaud, R., Hansard, M., Evangelidis, G., and Ménier, C. (2016). An overview of depth cameras and range scanners based on time-of-flight technologies. *Machine Vision and Applications*, 27(7):1005–1020.

[22] Howard, R., Heaton, A., Pinson, R., and Carrington, C. (2008). Orbital express advanced video guidance sensor. In *IEEE Aerospace Conference*, pages 1–10.

[23] Kemp, C. and Drummond, T. (2005). Dynamic measurement clustering to aid real time tracking. In *IEEE Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 1500–1507.

[24] Klein, G. and Murray, D. W. (2006). Full-3D edge tracking with a particle filter. In *Proc. British Machine Vision Conf. (BMVC)*, pages 1119–1128. British Machine Vision Association.

[25] Lentaris, G., Stratakos, I., Stamoulias, I., Maragos, K., Soudris, D., Lourakis, M., Zabulis, X., and Gonzalez-Arjona, D. (2017). Project HIPNOS: Case study of high performance avionics for active debris removal in space. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. To appear.

[26] Lourakis, M. and Zabulis, X. (2013). Model-based pose estimation for rigid objects. In *Intl. Conf. on Computer Vision Systems (ICVS)*, volume 7963 of *LNCS*, pages 83–92. Springer-Verlag.

[27] Lowe, D. (1991). Fitting Parameterized Three-Dimensional Models to Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(5):441–450.

[28] Lowe, D. (1992). Robust model-based motion tracking through the integration of search and estimation. *Intl. Journal of Computer Vision*, 8(2):113–122.

[29] MacLean, S. and Pinkney, H. (1993). Machine vision in space. *Canadian Aeronautics and Space Journal*, 39(2):63–77.

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

[31] Petit, A., Marchand, E., and Kanani, K. (2012). Tracking complex targets for space rendezvous and debris removal applications. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4483–4488.

[32] Reinke, S., Gutzeit, E., Mesing, B., and Vahl, M. (2012). Tracking technical objects in outdoor environment based on CAD models. In *Proc. Intl. Symp. on Advances in Visual Computing (ISVC)*, pages 437–446. Springer.

[33] Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Proc. IEEE Intl. Conf. on Computer Vision (ICCV)*, volume 2, pages 1508–1515. IEEE Computer Society.

[34] Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880.

[35] Teuliere, C., Marchand, E., and Eck, L. (2010). Using multiple hypothesis in model-based tracking. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4559–4565.

[36] Vacchetti, L., Lepetit, V., and Fua, P. (2004). Combining edge and texture information for real-time accurate 3D camera tracking. In *Proc. Intl. Symp. on Mixed and Augmented Reality (ISMAR)*, pages 48–57.