AIAA SciTech Forum, 2018 AIAA Guidance, Navigation, and Control Conference, 8–12 January 2018, Kissimmee, Florida
https://doi.org/10.2514/6.2018-2100

Multi-View Monocular Pose Estimation for Spacecraft Relative Navigation

Duarte Rondao* and Nabil Aouf†

Cranfield University, Shrivenham, Swindon, SN6 8LA, United Kingdom

This paper presents a method of estimating the pose of a non-cooperative target for spacecraft rendezvous applications employing exclusively a monocular camera and a three-dimensional model of the target. This model is used to build an offline database of pre-rendered keyframes with known poses. An online stage solves the model-to-image registration problem by matching two-dimensional point and edge features from the camera to the database. We apply our method to retrieve the motion of the now inoperational satellite ENVISAT. The combination of both feature types is shown to produce a robust pose solution even for large displacements respective to the keyframes which does not rely on real-time rendering, making it attractive for autonomous systems applications.

I. Introduction

A space rendezvous mission is defined as an orbital manoeuvre during which a chaser vehicle gets connected to, or approaches to a very close distance, a target spacecraft or space object in general. It is regarded as a highly complex and challenging operational process.[1] A *non-cooperative rendezvous* (NCRV), in particular, refers to a rendezvous with a space object that does not provide effective cooperative information: the target does not provide any aids for rendezvous sensors, its orientation is not controlled, and its motion may not be known accurately. This type of mission architectures demands autonomous operations due to the delays inherent to the transmission of signals between the Earth and the orbits in question. The safety of a NCRV typically requires high reactivity to unknown disturbances, which is difficult to achieve with a closed loop from the Earth. In terms of this autonomy, control and guidance algorithms have gained the required maturity to be implemented in real-world applications, whereas robust navigation solutions, in contrast, are still to be proposed to meet current and future challenging missions,[2] opening the door to the development of capable relative strategies for visual navigation.

An example of a mission architecture involving NCRV is *active debris removal* (ADR). The importance of ADR stands on the fact that orbital debris poses a threat to current and future spacecraft orbiting the Earth; given the increasing growth of the space industry and its recent extension towards the private sector, this results in more launches, which in turn contribute to an unbounded growth in space junk. ADR therefore has the goal of capturing and de-orbiting space debris to successfully provide a stopping power to this trend. To capture the target, the chaser spacecraft bearing the ADR mechanism requires a precise knowledge of its relative motion. Active remote sensing devices such as Lidar and time-of-flight sensors have been proposed for this purpose; however, they are often characterized by high mass, power consumption, and cost, preventing their wide usage. On the contrary, *passive sensors* such as digital cameras provide a lower costing and lighter alternative when combined with adequate image processing algorithms due to their reduced dimensions and mass production.

For relative motion estimation, monocular camera-based systems extract two-dimensional *features* from the target image. When information about its three-dimensional structure is known such that a computer-aided design (CAD) of the target is available (which is common for man-made satellites) the motion can be evaluated by *matching* the 2D features from the camera frame with the 3D reference points from the model. Then, the rotation and translation (i.e. the *pose*) of the object with respect to the camera coordinate system

*PhD Student, Centre for Electronic Warfare, Information and Cyber, Defence Academy of the UK, AIAA Student Member.

†Professor of Autonomous Systems, Centre for Electronic Warfare, Information and Cyber, Defence Academy of the UK.

that yield the best alignment between these matches can be extracted, effectively solving the *model-to-image registration problem.* A commonly used type of features are *interest points*, or corners. In 1988, Harris and Stephens mathematically formalised Moravec's work[3] to assemble their well-known corner detection algorithm.[4] A fixed-size image patch around the extracted point can then be used to detect similar regions in other images in a template-matching process, or tracked over a temporal sequence using optical flow[5] constraints. More recently, with his Scale-Invariant Feature Transform (SIFT),[6] Lowe pioneered the creation of a new class of feature detectors that encompass a variable-sized region centred around the interest point, seldom called *blobs.* This method embodies scale-space filtering and orientation assignment to make the features *invariant to perspective changes.* Other algorithms that improve SIFT's computational time while maintaining much of its accuracy have since been developed.[7–9] From the detected blob, a *descriptor vector* can then be computed to match the features. For extensive amounts of features, descriptor matching can be a costly process in terms of computation, although a novel class of methods has emerged that uses a binary string data representation,[10–12] reducing the required time for the correspondence process. Other types of features may be considered, such as *edges.* Whereas traditionally images can be processed to yield object contours[13] and then subject to a voting process to extract straight lines,[14] more sophisticated approaches showcasing false detection control, scale-space extraction, edge start and endpoint detection, and faster performance are now available.[15,16]

With respect to model-based relative navigation, two main approaches can be followed:[17] *tracking* and *detection.* In the first method, the pose estimate is initialised and subsequently propagated in time by tracking features from frame to frame. An example of relative pose estimation by tracking is the work by Comport et. al.,[18] which uses edge features from industrial CAD models in a virtual visual servoing application. The camera motion between frames is assumed to be small so that sampled 3D control points from the model are reprojected on the image plane using a predicted pose, followed by a one-dimensional search to find the corresponding edge on the feature space. The found edge can then be tracked to the following frame, and more control points can be generated on-the-go using rendering techniques. Kelsey et. al., and currently Petit et. al., have expanded this method and applied it to spacecraft relative navigation using edge,[19] colour,[20] and interest point[21] information, where control points are rendered using the aid of a graphics processing unit (GPU) in the latter work. However, rendering a 3D model in real-time is a time-consuming task which may prove dire for on-board autonomous space systems, where processing power is rationed for each subsystem and GPU capabilities are not available.

For this paper, we instead propose a relative navigation solution by detection: 2D image feature points and edges are matched to a database of 2D model features extracted from pre-rendered views termed *keyframes*, and whose 3D location on the model's surface can be pinpointed. The advantage is that this database can be built in an offline training phase; this is the reasoning behind the work of Vacchetti et. al.[22] In addition, the latest advances on feature detection and description provide an efficient framework for large baseline matching and real-time performance. Similar approaches have been applied to the space domain: Shang et. al.[23] match detected line segments with ones from synthetic model views; Post et. al.[24] use feature points, but depend on triangulating tracked features to estimate their 3D coordinates to match with the model points; whereas our method is based purely on 2D-2D feature matching.

The structure of this paper is organised as follows: Section II describes the image processing algorithms used to detect and register features, Section III discusses the problem of pose estimation, namely the frames of reference involved and the adopted solution to compute the rigid transform between them, Section IV analyses the methodology followed for the implementation of the proposed algorithm, and lastly Sections V and VI showcase the results obtained in a simulated environment and present the attained conclusions, respectively.

II. Image Processing

The model-to-image registration problem can traditionally be solved exclusively using 2D-3D point correspondences; in this case the problem is more concretely termed the *perspective-n-point* (PnP) problem, for which there exists an extensive number of algorithms capable of solving it.[25–27] As shall be seen, however, point-based features are not free from drawbacks, and a relative motion solution may benefit from combining different types of features. For the purpose of this work, we combine point with edge features to estimate the pose. This section discusses the computer vision algorithms utilised to process them.

A. Feature Point Detection in Scale-Space

Although invariant to rotation, corner detectors such as Harris'[4] employ a fixed window size which makes interest point detection sensitive to scale changes. The SIFT[6] algorithm makes use of scale-space filtering to tackle this issue. A difference of Gaussians (DoG) is used to approximate the Laplacian of Gaussians (LoG); it is obtained by computing the difference between two Gaussian blurs of the same image with different standard deviations separated by a constant factor, i.e. σ and $k\sigma$. Successive blurrings are performed until the last layer is transformed with a value of twice the initial σ. Once a complete octave is processed, this layer is down-sampled by a factor of 2, marking the start of the following octave. Once all the DoGs are found, the resulting structure is searched for extrema in space ($\mathbf{x}$) and scale (σ): each sample point is compared to its eight neighbours in the current image and nine neighbours in the scale. It is selected as a potential feature if it is either larger or smaller than all of them.

As a further refinement, each potential feature is subjected to a rejection process based on a contrast threshold value. Additionally, in order to reject edges, a process similar to the Harris corner detector is employed by computing the 2×2 Hessian matrix $\mathbf{H}$ of the difference image D at the location and scale of the interest point

$$\mathbf{H}(\mathbf{x}, \sigma) = \begin{bmatrix} D_{xx}(\mathbf{x}, \sigma) & D_{xy}(\mathbf{x}, \sigma) \\ D_{yx}(\mathbf{x}, \sigma) & D_{yy}(\mathbf{x}, \sigma) \end{bmatrix}, \tag{1}$$

and subjecting its ratio of principal curvatures to an edge threshold. The quantities D_{xx}, etc., are the second-order derivatives of D, estimated by taking differences of neighbouring sample points.

Although accurate, this method is characterised by a considerable computational burden, which is particularly crucial for platforms used in autonomous real-time applications. The Speeded Up Robust Features (SURF)[7] algorithm was developed by Bay et. al. with this motivation in mind. We alternatively detect point features using the *Fast-Hessian detector*, introduced by the authors in their original paper, which aims to provide a computationally faster version of SIFT's DoG detector. The Fast-Hessian makes use of a further approximation of the LoG by using box filters, which can be evaluated swiftly and independently of size using integral images. The box filters are used to compute approximations to the derivatives D_{xx}, etc. For instance, 9×9 box filters are approximations for Gaussian second order derivatives with $\sigma = 1.2$. These approximations are consequently used to produce an estimation of the determinant of $\mathbf{H}$, which is used in a thresholding process to cull weak features.

B. Binary Feature Point Description

With the advent of algorithms such as SIFT and SURF, the point detection paradigm shifted towards the registration of a variable-sized blob around that point. The idea is that these surrounding regions could provide distinctive enough information to be used for feature matching, and should therefore be encoded into floating-point descriptor vectors. Then, by computing the Euclidean distance between the descriptors, a correspondence between features can be developed.

For this study, we rely instead on encoding information into *binary strings*, using the Fast Retina Keypoint (FREAK) descriptor,[12] which takes inspiration in the design of the human retina. The method adopts the retinal sampling grid as the sampling pattern for pixel intensity comparisons; this pattern resembles a circular geometry where the density of points drops exponentially from the centre outwards, mimicking the spatial distribution of ganglion cells in the eye. These are segmented into four different areas, which is believed to result in a body resource optimization, where a higher resolution is captured in the fovea (inner-most circle), while lower acuity images are formed in the perifovea (outer-most circle). To match this biological model, the algorithm uses different kernel sizes for the Gaussian smoothing of every sample point in each receptive field, where these overlap for added redundancy leading to increased discriminative power. To determine which pairs of pixels to compare, Alahi et. al. defend that a coarse-to-fine pair selection yields the largest variance and uncorrelation between pairs, i.e. the first selected pairs compare sampling points in the outer circles and the last pairs compare points in the inner circles. This is interestingly consistent with modern understanding of the retina, where the perifoveal fields are first used to estimate the location of a point of interest and the validation is then performed with the densely distributed foveal receptive fields. Effectively, to describe a (even static) scene, the eye moves around with discontinuous individual movements called saccades. As such, FREAK emulates this process by parsing the computed descriptor in a way that the first

16 bytes represent coarse information, which is applied as a triage in the matching process. This way, a cascade of comparisons is performed, accelerating the procedure even further. For rotation-invariance, the orientation of the feature is estimated using local gradients.

Using binary descriptors is advantageous as feature matching can be performed with resort to the Hamming distance, which provides better performance with respect to the Euclidean distance test used with floating point descriptors: it consists only in applying the exclusive or (XOR) logical operator followed by a bit count.

C. Edge Detection with False Positive Control

Whereas recent research has directed an effort towards creating point feature descriptors invariant to perspective changes and scene conditions, these are not free from weak spots. Effectively, these features are not always impervious to illumination changes, and matching failure due to partial occlusion is an issue. In contrast, edge features are typically less distinctive but carry an extra degree of robustness by showing stability towards such conditions.[17]

We perform the detection of edge features using the Edge Drawing Lines (EDL) detector.[16] The algorithm is divided into three main steps. Firstly, the *edge drawing* method is applied: the grayscale input frame is filtered to remove noise and the gradient magnitude and orientation are computed at each pixel; peaks in this gradient map are marked as *anchors* due to their high probability of being edge elements; anchors are then connected by drawing edges between them. Secondly, line segments are extracted from the generated anchor chains using a least squares line fitting method. Lastly, the segments are subject to a *validation process*: for each line segment, the gradient orientation is computed for each pixel along it to assess the number of aligned pixels. The number of false alarms for the segment is then evaluated as

$$\mathrm{NFA}(n,k) = W^2H^2 \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}, \tag{2}$$

where n is the segment's length, k is the number of aligned pixels, $p = 0.125$ is the accuracy of the line direction (the alignment computation is discretised into 8 bins), W, H are the width and height of the image, respectively, and the brackets signify the binomial coefficient. The line segment is accepted as valid if $\mathrm{NFA} \leq 1$.

In their paper, the authors test the EDL detector against classic detection algorithms that start by computing a binary edge map, using for instance Canny's algorithm,[13] and that subsequently compute the Hough transform[14] to extract lines. They show that the present algorithm results in the extraction of more accurate, well-localized edges with considerably less false detections, while simultaneously reducing the computational effort.

III. Pose Estimation

In this section, we first start by defining the frames of reference, or coordinate systems, involved in the pose estimation problem. The fundamental transforms of image projection are covered. Lastly, the implemented pose estimation algorithm combining point and edge features is described.

A. Frames of Reference and Image Formation

Given a frame of reference $\mathcal{W}$ attached to the target (also called world coordinates) and the camera frame of reference $\mathcal{C}$ connected to the chaser, we are interested in finding the rigid transformation, i.e. the 3 × 3 rotation matrix $\mathbf{R}_W^C \equiv \mathbf{R}$ and the 3 × 1 translation vector $\mathbf{t}_W^C \equiv \mathbf{t}$, that maps a point $\mathbf{p}^W$ expressed in the $\mathcal{W}-$frame to one expressed in $\mathcal{C}$, $\mathbf{p}^C$. Both frames of reference are depicted in Figure 1. This information is often concatenated into a *pose matrix* $\mathbf{T}_W^C \equiv \mathbf{T}$:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{3}$$

where $\mathbf{0}$ is a null matrix of appropriate dimensions. For this purpose, we adopt the *pinhole camera model*:[28] the origin of the $\mathcal{C}-$frame coincides with the camera's optical centre, contained in the focal plane; a distance

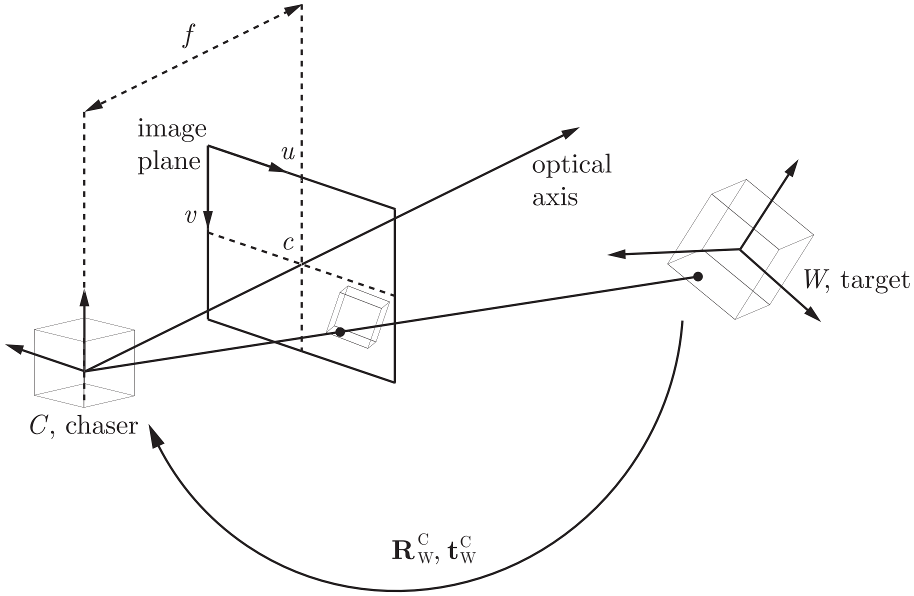

f
image plane
u
v
c
optical axis
W, target
C, chaser
$\mathbf{R}_{\mathrm{W}}^{\mathrm{C}}, \mathbf{t}_{\mathrm{W}}^{\mathrm{C}}$

Figure 1. Camera and target frames of reference.

f equal to the focal length separates this plane from the image plane, perpendicular to the camera's optical axis, where the 2D images are formed.

The intrinsic camera (or calibration) matrix, given by

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{bmatrix}, \tag{4}$$

is used to obtain the relation between a 3D point $\mathbf{p}^C = [p_x^C \, p_y^C \, p_z^C]^T$ in the $\mathcal{C}-$frame and a 2D point $\mathbf{p} = [u \, v]^T$ in the image plane expressed in pixel coordinates as

$$g\left(\mathbf{p}^C\right) = \begin{bmatrix} c_x + f_x p_x^C / p_z^C \\ c_y + f_y p_y^C / p_z^C \end{bmatrix}, \tag{5}$$

where f_x, f_y are the focal length components for the sensor's x, y dimensions, respectively, and c_x, c_y are pixel coordinates of the optical centre c. Thus, a point in the target frame of reference and a point in pixel coordinates are related through the full *reprojection equation*:

$$\mathbf{p} = h\left(\mathbf{p}^W, \mathbf{R}, \mathbf{t}\right) = g\left(\mathbf{R}\mathbf{p}^W + \mathbf{t}\right). \tag{6}$$

Note that the use of Eq. (6) implies that the *depth information* of the 3D point is lost.

B. M-Estimator Minimisation

The relative pose can alternatively be described by a *minimal-dimension* 6×1 vector $\mathbf{v}_W^C \equiv \mathbf{v}$ given by

$$\mathbf{v} = \begin{bmatrix} \mathbf{t} \\ \boldsymbol{\omega} \end{bmatrix}, \tag{7}$$

where $\mathbf{t}$ is the relative translation and $\boldsymbol{\omega}_W^C \equiv \boldsymbol{\omega}$ is the relative rotation, or Rodrigues, vector; it can be visualised as a vector representing an axis of rotation whose magnitude describes the angular displacement. Using this minimal representation, we estimate the pose by minimising the *reprojection error* between the reprojected model features and the detected features:

$$
\begin{aligned}
\widehat{\mathbf{v}} &= \underset{\mathbf{v}}{\operatorname{argmin}}\, \Delta\left(\mathbf{v}\right) \\
&= \underset{\mathbf{v}}{\operatorname{argmin}} \sum_{i=1}^{N} \rho\left(\sigma^{-1} r_i(\mathbf{v})\right), \qquad (8)
\end{aligned}
$$

where $\rho(\cdot)$ is a robust loss function, $r_i\left(\mathbf{v}\right) \equiv r_i$ is the residual for the $i-$th match, σ^2 is the variance associated with the measurements, and N is the total number of matches. Eq. (8) is a generalisation of the least squares method, which is unstable in the presence of *outliers* (i.e. erroneous correspondences between features); by choosing an *M-estimator* $\rho(\cdot)$ with a bounded influence function $\psi(x) \equiv d\rho(x)/dx$, the solution can be made robust to the presence of these outliers. In this sense, we call $\widehat{\mathbf{v}}$ an *M-estimate* of $\mathbf{v}$.[29] For the purpose of this work, we take $\rho(\cdot) = \rho_{\text{Tuk}}(\cdot)$ as the Tukey function. We approximate the scale $\sigma = \widehat{\sigma}$ for each set of correspondences with an estimate of the mean absolute deviation:

$$
\widehat{\sigma} = \frac{1}{\Phi^{-1}(0.75)} \sqrt{\underset{i \in N}{\operatorname{median}}\, r_i^2}, \qquad (9)
$$

where Φ^{-1} is the inverse of the cumulative normal distribution.

The use of M-estimation leads to what is known as an *iteratively reweighted least squares* (IRLS) problem, which can be solved with any of the classical least squares techniques, with the additional step of calculating weighing factors $w_i = w(x_i) \equiv \psi(x_i)/x_i$ using the current estimate of $\mathbf{v}_k$ at time $t = t_k$ in the computation of the new estimate of $\mathbf{v}_{k+1}$. We adopt a Levenberg-Marquardt (LM) scheme for the IRLS minimisation, for which the solution to Eq. (8) is given by

$$
\widehat{\mathbf{v}}_{k+1} = \delta\mathbf{v} \boxplus \widehat{\mathbf{v}}_k, \qquad (10a)
$$

$$
\delta\mathbf{v} = -\left(\mathbf{J}^T\mathbf{W}\mathbf{J} + \mu\mathbf{I}\right)^{-1} \mathbf{J}^T\mathbf{W}\mathbf{r}, \qquad (10b)
$$

where $\mathbf{J}$ is the Jacobian matrix, $\mathbf{W}$ is the diagonal matrix containing the weights w_i, $\mathbf{r}$ is the vector of scale-normalised residuals, μ is the LM weight, and $\mathbf{I}$ is an identity matrix of appropriate dimensions. The operator "$\boxplus$" denotes pose composition; in effect, the reason behind Eq. (10a) is that the pose vector $\mathbf{v}$ is a parametrisation of a manifold with structure $\mathbf{SO}(3) \times \mathbb{R}^3$, i.e. the *special Euclidean group*, $\mathbf{SE}(3)$. If the common Euclidean addition operator were to be used to update the pose estimate in Eq. (10a), there would be no guarantee that the resulting vector represents a pose. Instead, we carry out the optimization directly on the manifold while keeping a pose vector parametrisation by considering $\delta\mathbf{v}$ as the increment in the linearisation of the manifold around $\mathbf{v}$, and update the solution using its exponential map as:[30]

$$
\widehat{\mathbf{T}}_{k+1} = \mathrm{e}^{\delta\mathbf{v}} \widehat{\mathbf{T}}_k. \qquad (11)
$$

1. Point-based features

By computing the corresponding feature descriptors of each $\mathbf{p}_i$, we match them to features in one of the views from the database, for which the $\mathcal{W}-$coordinates $\mathbf{p}_i^W$ have been registered offline. Using the obtained set of 2D-3D correspondences, we minimise the function

$$
\Delta_p = \frac{1}{N_p} \sum_{i=1}^{N_p} \rho_p\left(r_i^p\right), \qquad (12)
$$

where $\rho_p(\cdot)$ is the Tukey M-estimator associated to the point features, $r_i^p = \sigma_p^{-1} d\left(\mathbf{p}_i, \mathbf{p}_i^W\right)$ with $d = h\left(\mathbf{p}_i^W, \mathbf{v}\right) - \mathbf{p}_i$, and N_p is the number of point matches. Each block of the $2N_p \times 6$ Jacobian matrix, $\mathbf{J}_p$, corresponding to the residual of each match can be shown to be given by[30]

$$\begin{aligned}\mathbf{J}_p^{r_i} &= \sigma_p^{-1}\frac{\partial g(\mathrm{e}^{\delta\mathbf{v}} \boxplus \mathbf{v} \boxplus \mathbf{p}_i^W)}{\partial(\delta\mathbf{v})} \\ &= \sigma_p^{-1}\begin{bmatrix} f_x/p_{z,i}^C & 0 & -f_x p_{x,i}^C/(p_{z,i}^C)^2 & -f_x p_{x,i}^C p_{y,i}^C/(p_{z,i}^C)^2 & f_x\left(1+(p_{x,i}^C)^2/(p_{z,i}^C)^2\right) & -f_x p_{y,i}^C/p_{z,i}^C \\ 0 & f_y/p_{x,i}^C & -f_y p_{y,i}^C/(p_{x,i}^C)^2 & -f_y\left(1+(p_{y,i}^C)^2/(p_{z,i}^C)^2\right) & f_y p_{x,i}^C p_{y,i}^C/(p_{z,i}^C)^2 & f_y p_{x,i}^C/p_{y,i}^C \end{bmatrix},\end{aligned} \tag{13}$$

where the function $g(\cdot)$ refers to Eq. (5), f_x and f_y refer to Eq. (4), and $\mathbf{p}_i^W, \mathbf{p}_i^C$ are the 3D point coordinates of the $i-$th match in the $\mathcal{W}-$ and $\mathcal{C}-$frames, respectively.

2. Edge-based features

Registered model edges are sampled to a discrete number of 3D points, which are then reprojected onto the image plane using Eq. (6), yielding the edge-based minimisation function

$$\Delta_e = \frac{1}{N_e}\sum_{i=1}^{N_e} \rho_e\left(r_i^e\right), \tag{14}$$

where $\rho_e(\cdot)$ is the Tukey M-estimator associated to the edge features, $r_i^e = \sigma_e^{-1} d_\perp\left(\mathbf{p}_i, \mathbf{p}_i^W\right)$ with $d_\perp = \mathbf{n}_i^T d\left(\mathbf{p}_i, \mathbf{p}_i^W\right)$, $\mathbf{n}_i$ is the normal of the $i-$th sampled edge point, and N_e is the number of sampled edge point matches. Note that Eq. (14) considers the normal distance between the detected and the reprojected edge points. This is because the edge matching algorithm performs a search along the line that passes through the sampled point i and is perpendicular to its projection, defined by its neighbours, for the closest pixel in the detected lines; hence the inclusion of the term $\mathbf{n}_i$.

The computation of the edge Jacobian matrix $\mathbf{J}_e$ has a similar form to Eq. (13), save the inclusion of the normal:

$$\mathbf{J}_e^{r_i} = \sigma_e^{-1}\mathbf{n}_i^T\frac{\partial g(\mathrm{e}^{\delta\mathbf{v}} \boxplus \mathbf{v} \boxplus \mathbf{p}_i^W)}{\partial(\delta\mathbf{v})}, \tag{15}$$

where the partial derivative has been evaluated therein.

3. Combining point- and edge-based features

As stated in Ref. 21, the IRLS framework provides a straightforward mechanism to couple different types of features for the estimation of the relative pose. The function to minimise becomes

$$\Delta = \alpha_p\Delta_p + \alpha_e\Delta_e, \tag{16}$$

where α_p, α_e are weighing factors that measure the contribution of each feature type. To compute these weights, we follow the method of Zou et. al.[31] which states that a larger number of features and a smaller residual vector should contribute more to the estimated solution. This allows us to combine the strong points of each feature type. We thus assign

$$\alpha_p = \frac{N_p}{\sqrt{\Delta_p}}\exp\left(-\Delta_p\right), \tag{17a}$$

$$\alpha_e = \frac{N_e}{\sqrt{\Delta_e}}\exp\left(-\Delta_e\right), \tag{17b}$$

followed by a normalisation:

$$\alpha_p \leftarrow \frac{\alpha_p}{\alpha_p + \alpha_e}, \tag{18a}$$

$$\alpha_e \leftarrow 1 - \alpha_p. \tag{18b}$$

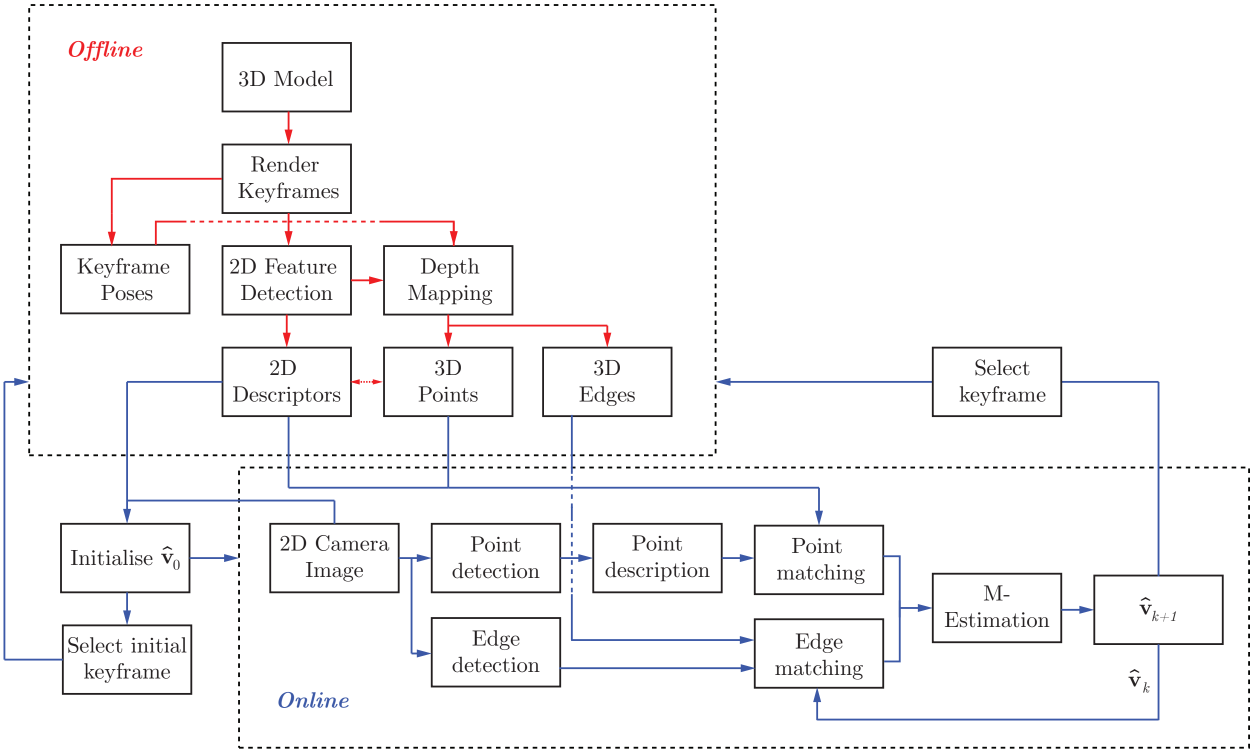


Figure 2. Architecture for the designed pose estimation solution.

Lastly, we set $\lambda_p = \alpha_p/N_p, \lambda_e = \alpha_e/N_e$ and a global residual vector and Jacobian can be defined, respectively, by a weighed stacking of the ones from each feature type as

$$\mathbf{r} = \begin{bmatrix} \sqrt{\lambda_p}\mathbf{r}_p^T & \sqrt{\lambda_e}\mathbf{r}_e^T \end{bmatrix}^T, \tag{19a}$$

$$\mathbf{J} = \begin{bmatrix} \sqrt{\lambda_p}\mathbf{J}_p^T & \sqrt{\lambda_e}\mathbf{J}_e^T \end{bmatrix}^T. \tag{19b}$$

A global weighing matrix is formed by arranging the respective matrices from each feature type, $\mathbf{W}_p$ and $\mathbf{W}_e$, in a block diagonal:

$$\mathbf{W} = \text{blockdiag}\left[\mathbf{W}_p, \ \mathbf{W}_e\right]. \tag{20}$$

Using the computed $\mathbf{r}$, $\mathbf{J}$, and $\mathbf{W}$, Eqs. (10) are used to solve Eq. (8) for the pose.

IV. Methodology

The procedure for the adopted pose estimation architecture is now presented. This architecture can be branched into two main aspects: the first one is the creation of an offline database using the CAD model, whereas the second one is an online stage which compares information between the buffer images and this database to yield the solution. Figure 2 is a schematic depicting the structure of the methodology.

A. Offline Database Creation

1. View-sphere sampling

The process of building the database starts off with the rendering of different views of the target's model. To accomplish this, we adopt the concept of the *view-sphere*, often used in the context of object detection.[32,33] We first define a sphere centred on the $W-$frame, and hence on the target model, with a minimum radius

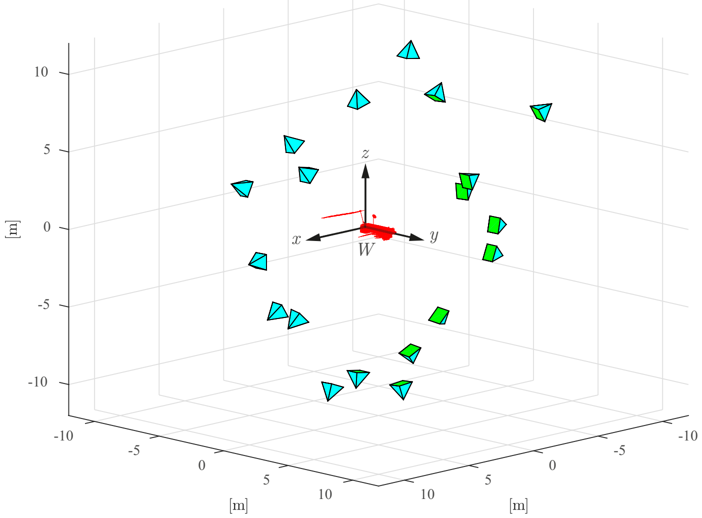


Figure 3. Visual representation of the employed multi-view sampling for training. Different viewpoints are obtained by changing the camera's (blue) elevation, azimuth, and distance with respect to the target (red) in the $W-$frame. The camera's front face is distinguishable (green).

such that it encases the whole model. Then, we position the camera at a certain point on the surface of the view-sphere, while pointing at the origin of the $W-$frame, and render a 2D keyframe from that viewpoint. By changing the elevation and azimuth of the camera through a set of discrete values on the view-sphere, as well as the sphere's radius itself, we are able to render a collection of keyframes that ensure the target is covered from multiple perspectives. Figure 3 illustrates the sampled views of the target for the present work.

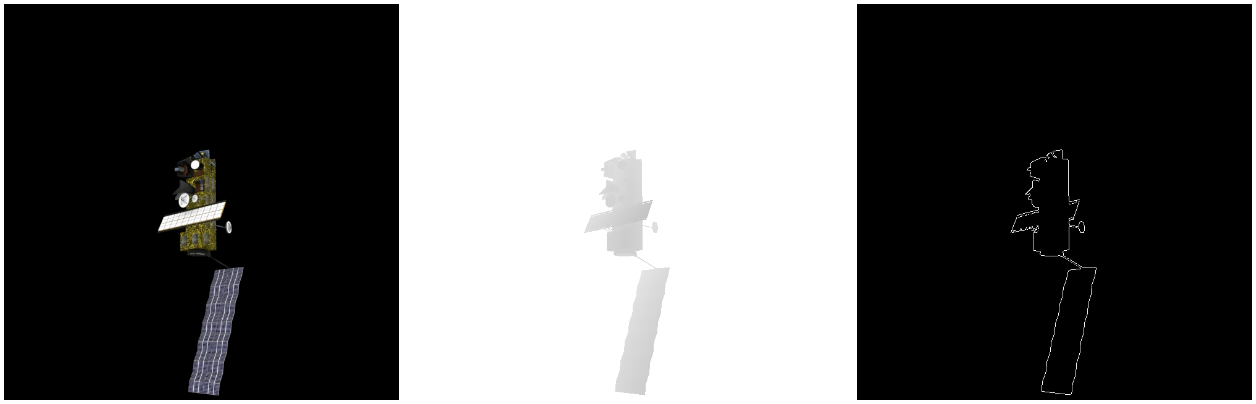
The pose $\mathbf{R}, \mathbf{t}$ for each keyframe is derived using the CAD software. Feature points and edges are extracted from each view using the image processing algorithms. In the case of feature points, the corresponding descriptor is computed using FREAK.

2. Data training

There are several ways to extract the 3D information required to train the images from the database. A simple method consists in back-projecting each point and edge extrema onto the surface of the CAD model. We may achieve this by first computing the ray that passes through each 2D point $\mathbf{p}_i$ by inverting Eq. (6). Then, since each face in the CAD mesh is defined as a triangle, the corresponding 3D model points $\mathbf{p}_i^W$ to each 2D feature can be found using a ray-triangle intersection algorithm, such as the one in Ref. 34.

However, this method is not free from drawbacks. Despite the assumption that it is meant to be used offline, the most simple ray-triangle intersection algorithms are often computationally costly as they require every mesh triangle to be tested. This is particularly impactful when dealing with complex CAD models. Another drawback is that the edge registration might fail for some cases, as these features are located on the boundary of the CAD model's 2D projection.

An alternative approach is the use of *depth mapping*, i.e. the generation of training images containing encoded information relating to the distance of the scene objects with respect to the camera viewpoint. For each textured training image in the database, using the same CAD software we can simply export the Z-buffer output for the same $\mathbf{R}, \mathbf{t}, \mathbf{K}$ to obtain the corresponding depth map. Since the depth data is encoded in the image's intensity values, this means that for a 2D feature detected at image plane coordinate $\mathbf{p}$, the scale of the corresponding 3D point with respect to the $C-$frame origin is found by accessing the same coordinates on the depth map. An image from the database and the corresponding generated depth map are represented in Figures 4(a) and 4(b), respectively.



(a) Textured image

(b) Depth map

(c) Extracted edges

Figure 4. Offline training. For each textured training image (a), an equivalent depth map is rendered (b), here normalised to an 8-bit depth image for visualisation where darker tones indicate a nearer surface, allowing for the 3D registration of detected features. The true contours are easily obtained from the latter (c).

An added benefit of using depth maps is their adequacy for the extraction of the target's edges. Since they are devoid of texture, the simple Canny's algorithm can be applied to detect the true 2D contours without noise (see Figure 4(c)), which improves the accuracy of the online edge matching process.

B. Online Pose Estimation

1. Nominal estimation

The online pose estimation loop consists of the following steps: (i) points and edges features are detected in the current camera image; (ii) the features are matched to features from the selected model keyframe; (ii-a) in the case of feature points, we register the two closest descriptor matches and subject them to a nearest-neighbour distance ratio (NNDR) test, i.e. the matching of the descriptors $\mathbf{s}_i$ and $\mathbf{s}_j$ is accepted if

$$\frac{d_{\mathrm{Ham}}(\mathbf{s}_i, \mathbf{s}_j)}{d_{\mathrm{Ham}}(\mathbf{s}_i, \mathbf{s}_k)} < \mu_{\mathrm{NNDR}}, \tag{21}$$

where $d_{\mathrm{Ham}}(\cdot, \cdot)$ is the Hamming distance, $\mathbf{s}_j, \mathbf{s}_k$ are the 1^{st} and 2^{st} nearest neighbours to $\mathbf{s}_i$ and μ_{NNDR} is a ratio from 0 to 1; (ii-b) in the case of edges, each template segment is sampled into 3D points which are then reprojected using the pose estimate from the previous time-step, and each point is tested for a potential match by searching for the closest detected edge along a normal search path with an empirically defined length; (iii) the obtained set of 2D-3D feature correspondences is used in the LM minimisation sub-loop of Eqs. (10) to solve the IRLS problem. The sub-loop is initialised with the previous pose estimate $\widehat{\mathbf{v}}_k$ and outputs a current estimate $\widehat{\mathbf{v}}_{k+1}$; (iv) the reference keyframe is selected such that the Euclidean distance between $\widehat{\mathbf{v}}_{k+1}$ and the pose registered to the keyframe in the offline stage is minimised. The loop returns to point (i) and is repeated for the new acquired image.

2. Initialisation

In order to launch the nominal estimation loop, we first recover an initial estimate $\widehat{\mathbf{v}}_0$ of the pose vector and select a model keyframe from the database for matching. To perform this, we carry out a search by matching the detected point features in the initial frame to every model view in the database using the descriptors. By using a PnP algorithm that does not require an initialisation, we can compute an approximate, initial pose hypothesis for each set of correspondences. For the current work, we use the EPnP[25] method in combination with RANSAC[35] to simultaneously obtain a pose estimate and reject outlying matches. From the approximation obtained for $\widehat{\mathbf{v}}_0$, the initial selected keyframe is chosen analogously to point (iv) in the nominal estimation procedure.

Notwithstanding a fitting approximation to obtain $\widehat{\mathbf{v}}_0$, it must be realised that each descriptor is a multidimensional vector and, consequently, solving the nearest-neighbour problem over the whole database

Table 1. Simulated camera properties.

Parameter	Value
Resolution [px × px]	640×640
Focal length [mm]	16.5
Field-of-view [deg × deg]	44×44
Measurement rate [Hz]	10

simultaneously using a standard brute-force algorithm is not adequate for real-time processing. In order to overcome this hurdle, we construct a *hierarchical $k-$means tree*[36] with all of the feature point descriptors in the database. First, a branching factor k that defines the number of clusters at each level of the hierarchy is selected. Then, the set of descriptors is grouped into k clusters using a standard $k-$means algorithm, cutting the tree such that their variance is minimised. Lastly, each sub-cluster is recursively clustered until a lower bound is reached. While this represents an approximation to the exact brute-force searching, it can be performed in a fraction of the computation time, being particularly useful when there is large inter-frame motion.

3. Reset

In order to prevent the degradation of the M-estimation solution, we monitor its associated translation and rotation covariances and apply a reset if they exceed a threshold. This reset consists in generating a pose estimate from the current 2D-3D point feature matches with EPnP and RANSAC. The new solution is only accepted if it yields a given minimum number of inliers, after which the M-estimation is resumed; otherwise the reset is rejected and a new one is attempted after a cool-down period, e.g. after a fixed number of frames.

V. Simulations

For the scope of this research work, an experimental setup was devised to test the methodology from Section IV.

A. Setup

For our analysis, a three-dimensional CAD model of the former remote sensing satellite ENVISAT is used to represent the target object. ENVISAT is a complex object, being formed by several modules, namely a solar panel array, a synthetic aperture radar (SAR), and several antennae, among others, connected to a main body unit which is covered by multi-layer insulation (MLI). Each model part is textured differently and therefore looks and reacts to illumination differently. A chaser spacecraft is assumed to have a body-mounted camera capable of collecting images in the visible wavelength, with the properties presented in Table 1. The chaser is taken to be observing the target, which is at a fixed hold point while rotating about its $y-$axis at a constant rate of $\dot{\phi} = 5\,\mathrm{deg\,s^{-1}}$ relative to the former; the sequence lasts 72 s, representing a full revolution along this axis. A total of 19 keyframes are used to build the database: these are generated according to Subsection IV-A (see Figure 3). All keyframes are rendered at a resolution of 640 px × 640 px, the same as the on-board camera's resolution (see Table 1). Images of the camera sequence and keyframes are simulated using the open-source 3D computer graphics software Blender.[a] The pose estimation framework was coded in the C++ programming language. The OpenCV[b] library, version 3.0, was used for computer vision and image processing related functions. The native implementations for the Fast-Hessian feature point detector, FREAK feature point descriptor, and EDL line detector were used. In the initialisation stage we make use of the Fast Library for Approximate Nearest Neighbours (FLANN) for the hierarchical clustering.[37]

[a] `http://www.blender.org/`

[b] `http://opencv.org`

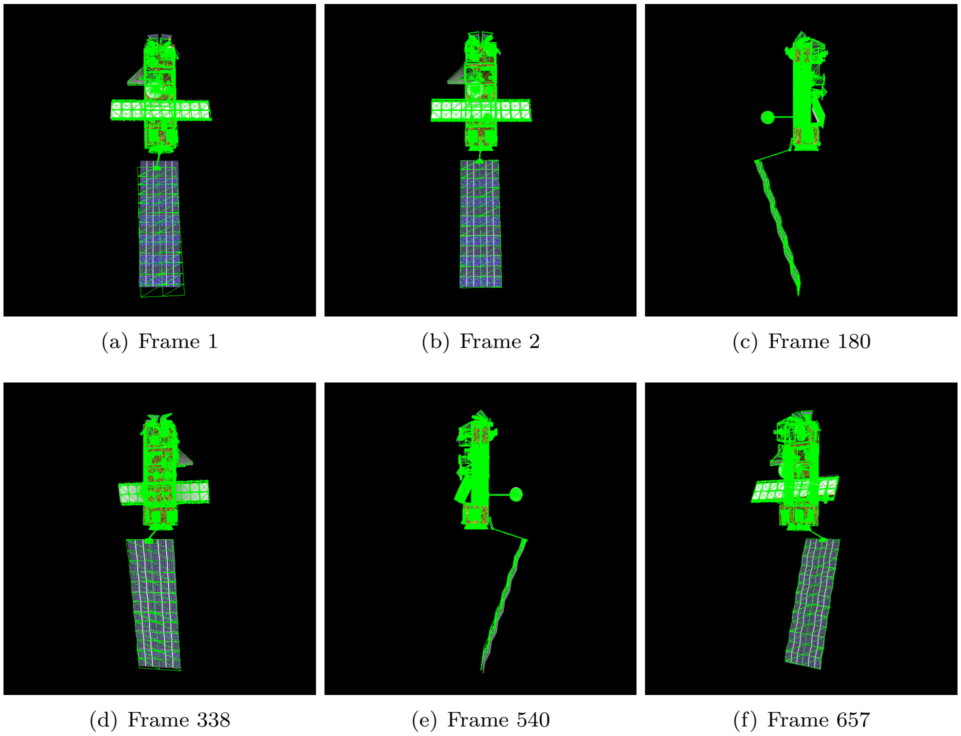

(a) Frame 1
(b) Frame 2
(c) Frame 180
(d) Frame 338
(e) Frame 540
(f) Frame 657

Figure 5. Model mesh (green) reprojected onto the camera image frames using the estimated pose for the considered sequence. Figure (a) represents the pose initialisation with hierarchical clustering, Figure (b) shows immediate convergence of the solution for frame 2. The pose estimate remains robust throughout Figures (c-f).

All simulations are carried out with a setup using an Intel® Core™ i7-6700 @ 3.40 GHz × 8 core central processing unit (CPU), 16 GB RAM system.

B. Results

The results for the estimated relative pose of the target are portrayed in Figures 5 and 6. It can be seen that most pose vector elements are in close agreement with the ground truth for the majority of the sequence. The largest errors in the translation and rotation simultaneously can be observed in two neighbourhoods centred on frames 180 and 540; effectively, these correspond to the periods when the target completes 90 deg and 270 deg rotations about its $y-$axis, respectively, showing the greatest degree of self-occlusion in the sequence. This results in the projected surface area of the target reaching a minimum, thus impacting the pose estimate. Figure 7 shows the estimation error for the sequence. The translation is accurate up to 0.25 m and the largest error is observed for the $z-$axis, highlighting the challenges of depth recovery in monocular applications. With respect to rotational motion, the error is kept under 8 deg whereas the ones about the $x-$ and $y-$axes bear the largest magnitude; these are the axes corresponding to out-of-plane rotation.

Both the translation as well as the rotation error maxima occur around frame 470, as the spacecraft nears the three-quarter turn, whereas the second and third largest error spikes take place near frames 240 and 560, after the quarter and the three-quarter turns are carried out, respectively. Moreover, the uncertainty of the M-estimation solution, obtained from the covariance matrix of $\delta\mathbf{v}$, is plotted in Figure 8. The reset events as described in Section III are also represented: in blue for accepted resets, and in red for rejected resets that did not produce the minimum required number of RANSAC inliers. From these plots it can be seen that the three aforementioned events correspond to successful pose resets. These represent trade-offs where estimation accuracy is necessarily sacrificed in order to prevent the Tukey M-estimator from converging to a local minimum. Note how the uncertainty of the solution is brought down after each successful reset.

By analysing Figures 9(a) and 9(b), it can be observed that the number of inliers and the minimisation function scores (i.e. the normalised residuals), respectively, tally with the progression of the error in time. Indeed, an decrease in the number of inliers and an increase in the residuals correlate with a degradation of the solution. The effect is more noticeable for the point features, where the decrease in the number of

Table 2. Pose estimation computation times.

Module	Time [ms]	Relative [%]
Brute-force initialisation	82.9024	-
Hierarchical clustering initialisation	46.7066	56.3392[a]
Point detection	17.0264	21.6849
Edge detection	3.5229	4.4868
Point description	5.5647	7.0872
Point matching	3.5382	4.5063
Edge matching	0.7316	0.9318
M-estimation	48.1335	61.3030
Nominal total	78.5174	100.0000

[a] With respect to the brute-force search counterpart.

inliers is sharper and the normalised residuals suffer an increase of approximately 15 %, when compared to edges, for which the increase in residuals is barely noticeable. This degeneration of the point features may be explained due to errors in the matching process coming from the topography of ENVISAT. The solar panel array consists of a repetitive grid pattern, whereas the MLI in the main body produces noise in the images when subject to changing illumination. For the former case, the computed descriptors might lack sufficient distinctiveness, and for the latter case it is possible that the features are too disparate from the ones in the database. Both cases will contribute to a decay in the matching process. The edge features are therefore shown to be more robust towards divergence between the camera images and the database.

Furthermore, Figure 9(c) shows the evolution in time of the self-tuning weights used in the IRLS minimisation process. For the beginning, middle, and end parts of the sequence the weights favour both types of features nearly evenly, showing a distribution of 40%-60% leaning towards the edges, which can be explained by their lower normalised residual magnitude being generally 67% lower than its feature point counterpart, even though the latter demonstrates a larger number of inliers. As the estimation error starts to increase, the weights begin to shift their influence further towards the edges; in particular, in the vicinity of frame 180, a weight of at least 90% is attributed to the edge features, as the algorithm reacts to the growth of the point features' residuals and to the decrease in the number of inliers. Thus, the edge features and the adaptive weighing mechanism prove to be key in preventing the pose estimation from diverging.

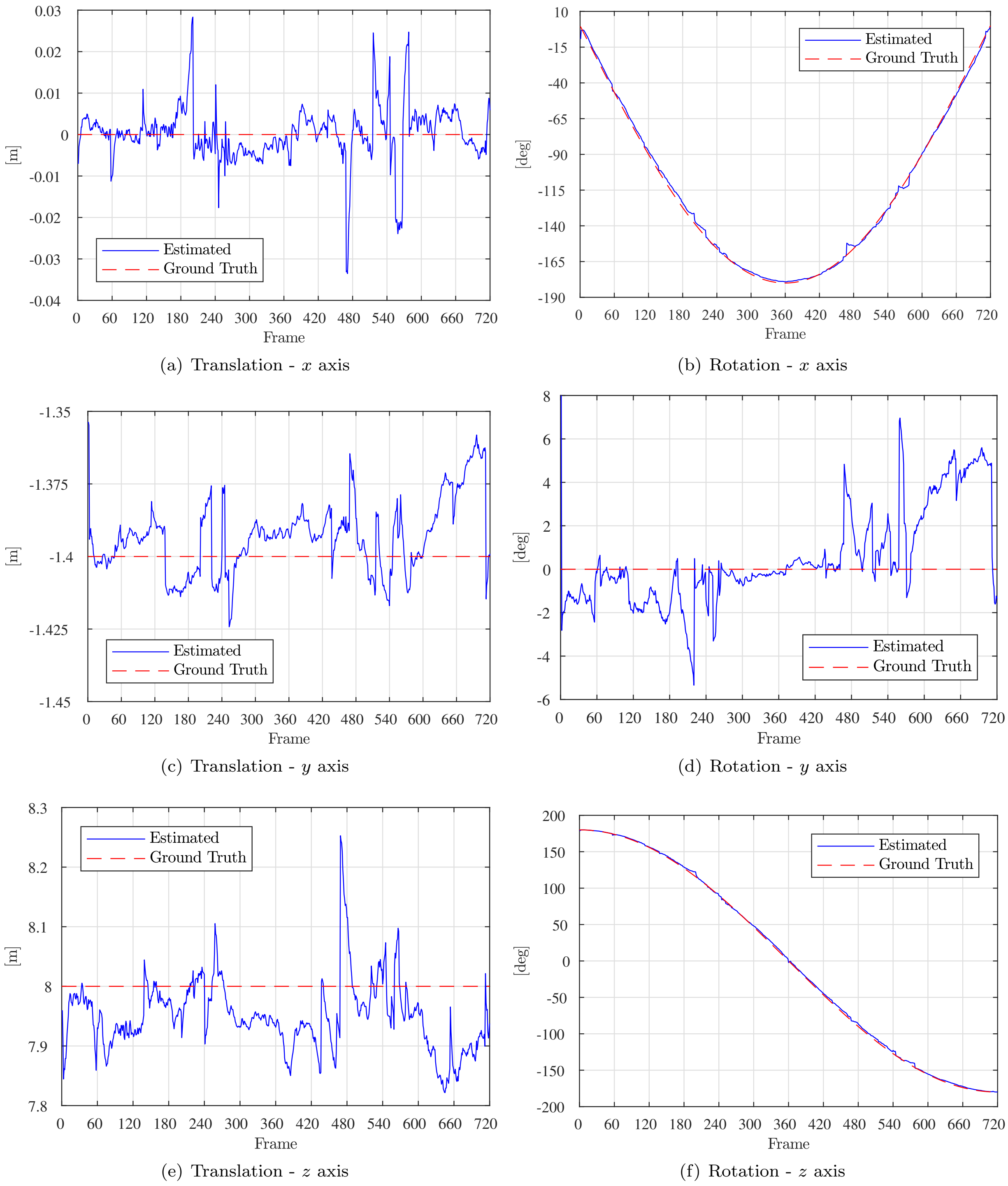
The same effect is observed in the neighbourhood of frame 540, albeit slightly more conservatively. The point features are seldom weighed more than the edges, this occurring only for frames 398, 400, and 403, where the number of inliers of the former peak at 350.

Lastly, Table 2 depicts the average computation times for the pose estimation framework. The initialisation times were obtained by averaging the results of 1000 trials where a random frame from the sequence was considered at each time. This accounts for the descriptor matching, NNDR test, and pose extraction with RANSAC + EPnP. It can be seen that our initialisation with the hierarchical clustering search cuts the running time in almost half when compared to brute-force searching, providing a solution with approximately 56% of the cost with acceptable accuracy (see Figures 5 (a) and (b)). The nominal pose estimation times were attained by averaging the results for each frame of the sequence. The mean nominal pose estimation time per frame is approximately 78.5 ms, equivalent to a mean frame rate of around 12 frames per second (FPS), where we again emphasize that only the CPU is being utilised and the potential for real-time capability. This is an improvement of 4 FPS relative to the work of Ref. 21 which makes use of GPU processing power. The M-estimation module is clearly the costliest one, taking up approximately 61% of the total execution time. However, this can be limited by tuning the algorithm's parameters, such as the maximum number of input point and edge matches, and the maximum number of LM iterations, possibly with a trade-off on accuracy, but ensuring the frame rate is kept above a desired minimum.

VI. Conclusions

In this work, we have developed a robust model-based solution for relative navigation by using a three-dimensional model of the target and hybrid features. The importance of the developed framework stands on the fact that it does not depend on complicated real-time model rendering techniques; instead, a select set of keyframes are rendered *a priori* for which 3D points are registered on the surface, allowing the retrieval of the pose based only on the matching of 2D point and edge features. The incorporated adaptive weighing algorithm autonomously shifts the influence of both types of features based on the quality of their matching.

We have tested our method on synthetic images of a complex, realistic spacecraft debris, showing promising results for visual-based NCRV, as the obtained solution shows an attitude error limited to 8 deg and sub-metre translation accuracy for a full target revolution at a high spin rate, relying only on the CPU. In the future, an inter-frame tracking module can be added to the present algorithm to further reduce the error and limit jitter.



(a) Translation - x axis

(b) Rotation - x axis

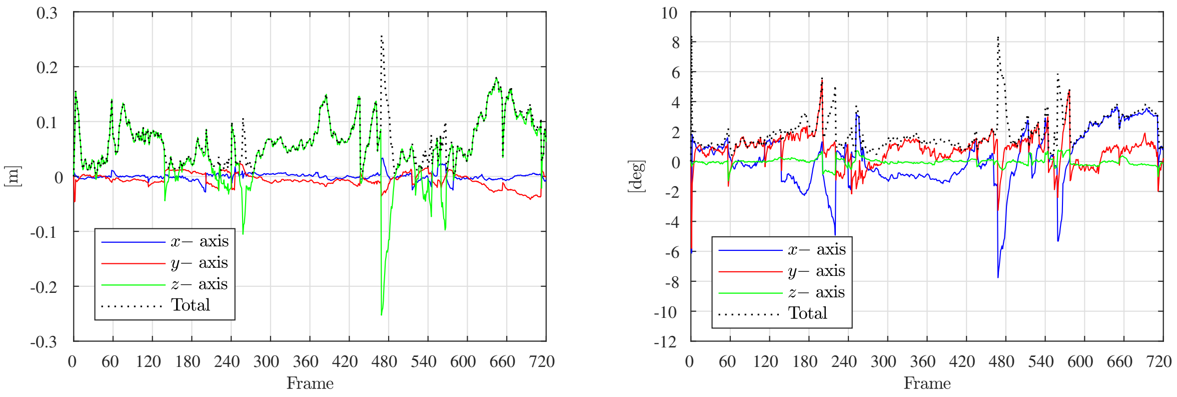
(c) Translation - y axis

(d) Rotation - y axis

(e) Translation - z axis

(f) Rotation - z axis

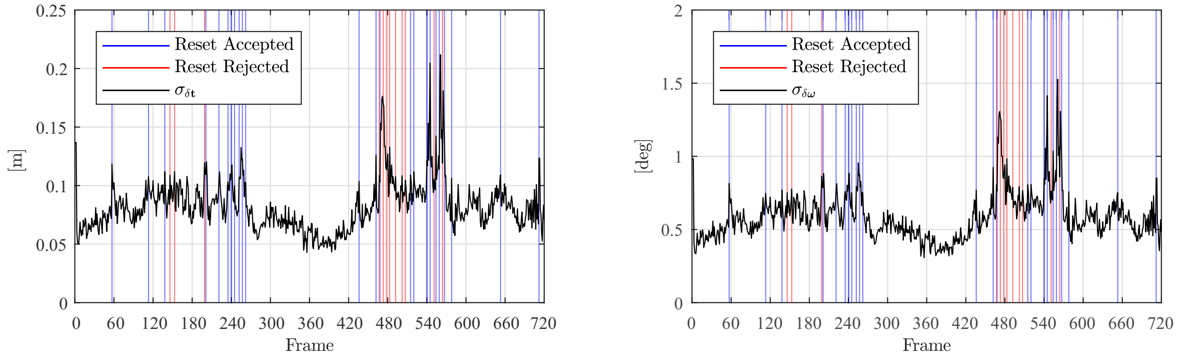
Figure 6. Estimated and true values for the target relative translation vector and rotation vector.



(a) Translation

(b) Rotation

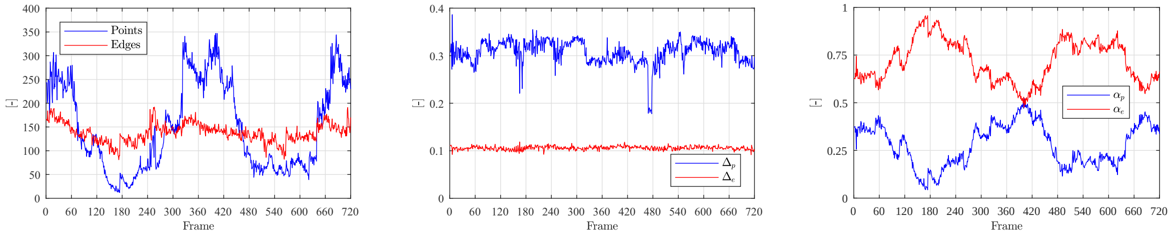
Figure 7. Estimation error for the target relative translation and rotation.



(a) Translation

(b) Rotation

Figure 8. Standard deviation of the target estimated relative translation and rotation.



(a) Number of inliers

(b) Minimisation function

(c) Weights

Figure 9. Figures of merit for point (blue) and edge (red) features.

References

[1] Luo, Y., Zhang, J., and Tang, G., "Survey of Orbital Dynamics and Control of Space Rendezvous," *Chinese Journal of Aeronautics*, Vol. 27, No. 1, 2014, pp. 1–11.

[2] Yawen, L., Yuming, B., and Gaopeng, Z., "Survey of Measurement of Position and Pose for Space Non-Cooperative Target," *34th Chinese Control Conference*, IEEE, 2015, pp. 5101–5106.

[3] Moravec, H. P., "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Tech. rep., DTIC Document, 1980.

[4] Harris, C. and Stephens, M., "A Combined Corner and Edge Detector," *Alvey Vision Conference*, Vol. 15, Citeseer, 1988, p. 50.

[5] Tomasi, C. and Kanade, T., "Detection and Tracking of Point Features," Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, 1991.

[6] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, 2004, pp. 91–110.

[7] Bay, H., Tuytelaars, T., and Van Gool, L., "Surf: Speeded Up Robust Features," *European Conference on Computer Vision*, Springer, 2006, pp. 404–417.

[8] Rosten, E. and Drummond, T., "Machine Learning for High-Speed Corner Detection," *European Conference on Computer Vision*, Springer, 2006, pp. 430–443.

[9] Agrawal, M., Konolige, K., and Blas, M. R., "Censure: Center Surround Extremas for Realtime Feature Detection and Matching," *European Conference on Computer Vision*, Springer, 2008, pp. 102–115.

[10] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., "ORB: An Efficient Alternative to SIFT or SURF," *International Conference on Computer Vision*, IEEE, 2011, pp. 2564–2571.

[11] Leutenegger, S., Chli, M., and Siegwart, R. Y., "BRISK: Binary Robust Invariant Scalable Keypoints," *International Conference on Computer Vision*, IEEE, 2011, pp. 2548–2555.

[12] Alahi, A., Ortiz, R., and Vandergheynst, P., "FREAK: Fast Retina Keypoint," *Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 510–517.

[13] Canny, J., "A Computational Approach to Edge Detection," *Transactions on Pattern Analysis and Machine Intelligence*, , No. 6, 1986, pp. 679–698.

[14] Duda, R. O. and Hart, P. E., "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Communications of the ACM*, Vol. 15, No. 1, 1972, pp. 11–15.

[15] von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G., "LSD: A Fast Line Segment Detector with a False Detection Control," *Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 4, 2010, pp. 722–732.

[16] Akinlar, C. and Topal, C., "EDLines: A Real-Time Line Segment Detector with a False Detection Control," *Pattern Recognition Letters*, Vol. 32, No. 13, 2011, pp. 1633–1642.

[17] Lepetit, V. and Fua, P., "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundations and Trends in Computer Graphics and Vision*, Vol. 1, No. 1, 2005, pp. 1–89.

[18] Comport, A. I., Marchand, E., Pressigout, M., and Chaumette, F., "Real-Time Markerless Tracking for Augmented Reality: the Virtual Visual Servoing Framework," *Transactions on Visualization and Computer Graphics*, Vol. 12, No. 4, 2006, pp. 615–628.

[19] Kelsey, J. M., Byrne, J., Cosgrove, M., Seereeram, S., and Mehra, R. K., "Vision-Based Relative Pose Estimation for Autonomous Rendezvous and Docking," *Aerospace Conference*, IEEE, 2006, pp. 20–pp.

[20] Petit, A., Marchand, E., and Kanani, K., "A Robust Model-Based Tracker Combining Geometrical and Color Edge Information," *International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 3719–3724.

[21] Petit, A., Marchand, E., and Kanani, K., "Combining Complementary Edge, Keypoint and Color Features in Model-Based Tracking for Highly Dynamic Scenes," *International Conference on Robotics and Automation*, IEEE, 2014, pp. 4115–4120.

[22] Vacchetti, L., Lepetit, V., and Fua, P., "Fusing Online and Offline Information for Stable 3D Tracking in Real-Time," *Computer Society Conference on Computer Vision and Pattern Recognition Proceedings*, Vol. 2, IEEE, 2003, pp. II–241.

[23] Shang, Y., Zhang, Y., and Liu, H., "3D Model-Based Detection and Tracking for Space Autonomous and Uncooperative Rendezvous," *Applied Optics and Photonics China*, International Society for Optics and Photonics, 2015, pp. 96761A–96761A.

[24] Post, M. A., Yan, X. T., Li, J., and Clark, C., "Visual Pose Estimation System for Autonomous Rendezvous of Spacecraft," *13th Symposium on Advanced Space Technologies in Robotics and Automation*, 2015, pp. 1–9.

[25] Lepetit, V., Moreno-Noguer, F., and Fua, P., "EPnP: An Accurate O(n) Solution to the PnP Problem," *International Journal of Computer Vision*, Vol. 81, No. 2, 2009, pp. 155–166.

[26] Kneip, L., Li, H., and Seo, Y., "UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability," *European Conference on Computer Vision*, Springer, 2014, pp. 127–142.

[27] Ferraz, L., Binefa, X., and Moreno-Noguer, F., "Very Fast Solution to the PnP Problem with Algebraic Outlier Rejection," *Conference on Computer Vision and Pattern Recognition*, IEEE, 2014, pp. 501–508.

[28] Szeliski, R., *Computer Vision: Algorithms and Applications*, Springer Science & Business Media, 2010.

[29] Stewart, C. V., "Robust Parameter Estimation in Computer Vision," *SIAM review*, Vol. 41, No. 3, 1999, pp. 513–537.

[30] Fernández-Madrigal, J.-A. and Blanco Claraco, J. L., *Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods: Introduction and Methods*, IGI Global, 2012.

[31] Zou, Y., Wang, X., Zhang, T., and Song, J., "Combining Point and Edge for Satellite Pose Tracking Under Illumination Varying," *12th World Congress on Intelligent Control and Automation*, IEEE, 2016, pp. 2556–2560.

[32] Thomas, A., Ferrar, V., Leibe, B., Tuytelaars, T., Schiel, B., and Van Gool, L., "Towards Multi-View Object Class Detection," *Conference on Computer Vision and Pattern Recognition*, Vol. 2, IEEE, 2006, pp. 1589–1596.

[33] Liebelt, J., Schmid, C., and Schertler, K., “Independent Object Class Detection using 3D Feature Maps,” *Conference on Computer Vision and Pattern Recognition*, IEEE, 2008, pp. 1–8.

[34] Möller, T. and Trumbore, B., “Fast, Minimum Storage Ray/Triangle Intersection,” *Journal of Graphics Tools*, 1997.

[35] Fischler, M. A. and Bolles, R. C., “Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Communications of the ACM*, Vol. 24, No. 6, 1981, pp. 381–395.

[36] Gifford, H., “Hierarchical k-Means for Unsupervised Learning,” Tech. rep., Carnegie Mellon University, 2014.

[37] Muja, M. and Lowe, D. G., “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration,” *VISAPP*, Vol. 2, No. 331-340, 2009, pp. 2.