AAS 24-192

RESOURCE-CONSTRAINED VISION-BASED RELATIVE NAVIGATION ABOUT SMALL BODIES

F. Piccolo*, C. Balossi†, P. Panicucci‡, M. Pugliatti§, F. Topputo¶, F. Capolupo‖

Vision-based navigation is a common technique for spacecraft operating in close proximity of small bodies. Past missions have consistently used it to navigate spacecraft in challenging and partially known dynamical environments. However, they relied extensively on ground-based and human-in-the-loop processing, meaning that communication delays limited spacecraft reactivity and that large amount of images needed to be transmitted from the spacecraft to Earth. In recent years a significant push towards autonomy has emerged in the space sector. This would decrease mission costs, increase scientific return and reduce the operational effort of deep-space communication assets. In this work, a resource-constrained vision-based navigation algorithm for on-board use is presented. By working only with patches or windows extracted from the input images, on-board computational and storage requirements are significantly reduced while keeping the same level of accuracy of a baseline implementation. The performance of the algorithm is verified through a Monte Carlo campaign in which various sources of uncertainty are considered, including camera calibration parameters. Furthermore, a comparison is carried out between the computational time required by a standard version of the algorithm and the window-based one.

INTRODUCTION

Vision-Based Navigation (VBN) relies on the use of optical cameras for spacecraft navigation. This technique has been used for all missions to small bodies, enabling accurate navigation in a challenging dynamical environment and with little a-priori knowledge of the target body. Specifically, past missions used stereophotoclinometry (SPC), an accurate but computationally intensive technique based on the 3D reconstruction of the small body surface.[1] SPC is executed on ground using images downlinked from the spacecraft. While this technique has proven highly reliable for small body missions, it requries a large number of images taken from different points of view and with different illumination conditions, and extensive effort from human operators.

At the same time, there is an increasing desire for autonomy in space missions. Advancing spacecraft autonomy would reduce mission costs, improve scientific return and lower the pressure on deep-space communication assets, which are already operating at maximum capacity.[2] VBN is one of the most promising techniques for spacecraft autonomy. Cameras are low size, weight and

*PhD Student, Department of Aerospace Science and Technology, Politecnico di Milano, felice.piccolo@polimi.it
†PhD Student, Department of Aerospace Science and Technology, Politecnico di Milano, claudia.balossi@polimi.it
‡Assistant Professor, Department of Aerospace Science and Technology, Politecnico di Milano, paolo.panicucci@polimi.it
§PhD Student, Department of Aerospace Science and Technology, Politecnico di Milano, mattia.pugliatti@polimi.it
¶Full professor, Department of Aerospace Science and Technology, Politecnico di Milano, francesco.topputo@polimi.it
‖Guidance, Navigation and Control Systems Engineer, European Space Research and Technology Centre, ESA, francesco.capolupo@esa.int

power (SWAP) sensors that are already commonly used for deep space missions, and they provide a large amount of information on the surrounding environment. To date, space missions have relied on autonomy only when no alternative was possible. Examples are the touch-and-go (TAG) operations of the Hayabusa, Hayabusa2 and OSIRIS-REx missions.[3–5] In this phase, spacecraft need real-time control, which is not feasible from ground because of communication delays. Thus, these missions relied on autonomous VBN. Hayabusa and Hayabusa2 used recognisable artificial markers that were released by the spacecraft and deposited on the surface of their target.[4] The position of the markers was estimated with high accuracy on ground, and then they were used as reference points for autonomous navigation during the spacecraft descent. On the other hand, OSIRIS-REx used a technique called Natural Feature Tracking, in which natural landmarks from the asteroid surface were rendered in real-time on board and then correlated with images taken during the descent.[5] The 3D models used for the rendering of the landmarks were generated on ground and uploaded to the spacecraft. Therefore, both techniques relied on information generated a-priori on ground.

In this work, a VBN algorithm based on feature tracking and visual odometry is investigated. To reduce the storage and computational resources required, the algorithm does not process entire images. Instead, it considers only a set of image patches called windows. Features are extracted within these windows using the well-known Harris detector,[6] and tracked with a pyramidal Kanade-Lucas-Tomasi (KLT) algorithm.[7] The tracked features provide information about the spacecraft motion, which is then fed to an Extended Kalman Filter that estimates the spacecraft state. The use of windows significantly reduces the number of pixels stored and processed by the VBN algorithm, effectively reducing the computational cost. On the other hand, it also reduces the amount of information available for the image processing and navigation functions, and particular care is required in order to decide in which parts of the image windows should be extracted. It is worth highlighting that the use of windows also brings some additional advantages. Indeed, it effectively provides a first guess for the KLT algorithm, easing the convergence of feature tracking. Furthermore, it reduces the number of pyramid levels required, since the largest part of the motion is already accounted for by the windows extraction procedure, as will be detailed later.

The algorithm is tested on a close flyby to asteroid Bennu. The flyby is assumed to be carried out after an initial characterization phase, similarly to what will be done during the Hera mission.[8] Therefore, partial information about the asteroid is assumed to be available, such as the asteroid mean radius and the inertial direction of the spin axis. The performance of the VBN algorithm is investigated through numerical simulations relying on synthetic images. The latter are generated using the open-source Blender software*, which can produce realistic scenes thanks to its ray-tracing capabilities. Relevant camera effects, such as distortion, point-spread function (PSF) and noises are then added. An extensive Monte Carlo campaign has been carried out to properly consider the effect of uncertain variables such as camera calibration parameters and misalignment.

This work has been carried out in the context of the ESA-funded project "Star Tracker Autonomous Relative Navigation (STAR Nav)", led by Politecnico di Milano and Leonardo, which has been proposed to assess to what extent star trackers can replace navigation cameras to autonomously navigate around celestial bodies.[9] Besides the specific image characteristics of star trackers, an essential aspect of the study is the fact that the considered star trackers are not capable of reading full images from the detector because of memory constraints. Therefore, for each frame only a limited set of image patches can be read and used for subsequent processing. The objective of this work is to develop a VBN algorithm suitable for implementation on such limited-capacity hardware.

*https://www.blender.org/

METHODOLOGY

This section illustrates the components of the VBN algorithm and the procedure used to generate the synthetic images. An overview of the algorithm is first given, followed by a description of the methods used to decide where to extract windows and by the implementation details regarding the image processing and EKF. Finally, the image generation procedure is discussed.

VBN Algorithm Overview

The workflow of the VBN algorithm is illustrated in Figure 1. When a new image is available, the first step is to propagate the EKF-estimated state to the current time instant. The propagated state is used to reproject window positions from the previous to the current camera view. Windows are then extracted from the input image, and fed to the KLT to perform feature tracking. The tracked features are used to estimate the direction of motion of the spacecraft using a visual odometry method, coupled with MSAC to reject incorrect correspondences.[10] This measurement is then used for the EKF update step, after which a new set of windows is computed and extracted. At this point, new features are identified using the Harris detector, and then the loop is repeated. At the first time step, the estimated spacecraft position is retrieved from EKF initialization data. It is assumed that the EKF is initialized at the time of the first image using data provided by ground or by other algorithms running on board. The relevant steps of the algorithm are explained in detail in the following sections.

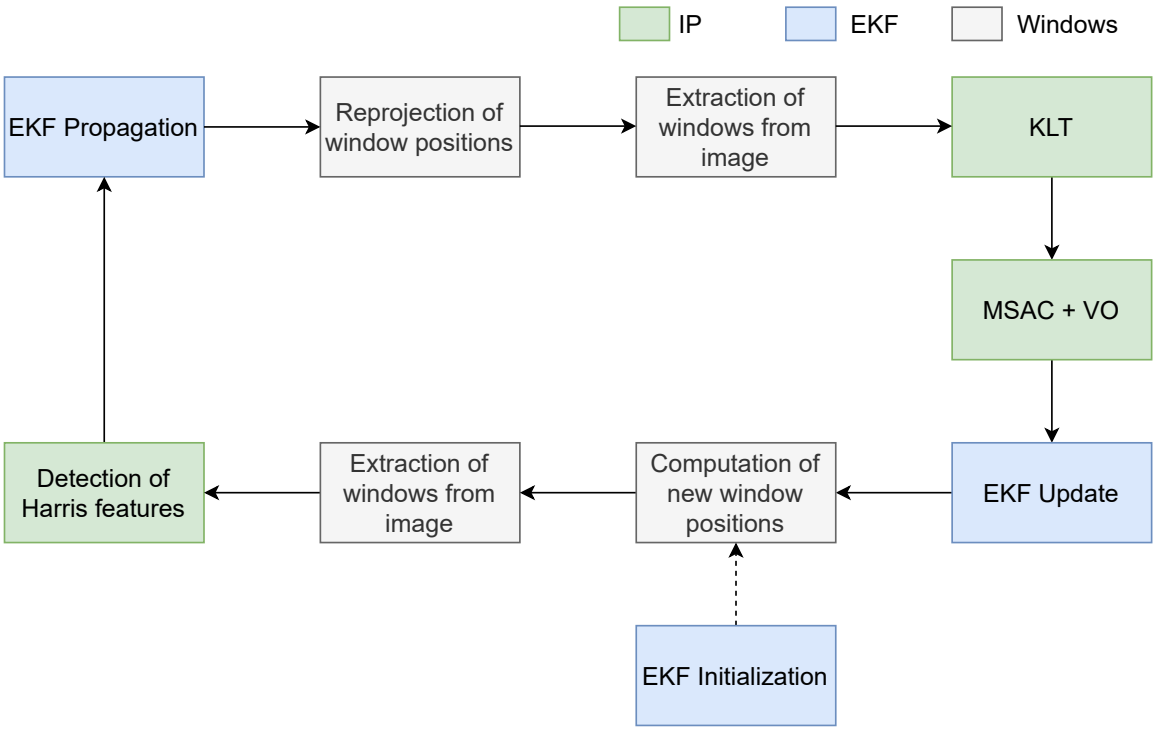


Figure 1: Vision-based navigation algorithm overview.

Window Positions

The determination of window positions is an essential step for the VBN algorithm. In order to obtain an accurate motion estimate, it is important not only to identify recognizable feature points, but also that they are spread as much as possible over the camera field of view (FOV). To guarantee this, windows should be spread over the asteroid, while still remaining within the illuminated portion of the surface.

In order to reduce storage and computational requirements, window positions are computed a priori, without relying on information extracted from the image itself. Thus, this step is based on the position estimate provided by the navigation algorithm, together with a rough estimate of the asteroid shape. For the case of Bennu, considered in this work, a spherical approximation of the asteroid is sufficient. Additionally, an estimate of the Sun direction is used.

Two different windows extraction mechanisms are needed to perform feature tracking, as specified in Figure 1. The first is the computation of a new set of windows, which is performed prior to feature detection. The other is the reprojection of previously extracted windows in the current camera view. The latter is necessary to track features, as the same areas of the asteroid surface need to be observed in two consecutive images. It is worth noting that the algorithm is based on image-to-image tracking, after which features are discarded. In other words, there is no attempt to track features over multiple images. This simplifies the windows extraction procedure, as windows do not need to follow the movement of features over multiple images.

Computation of new window positions New windows are extracted as follows. First, given the camera calibration matrix and the estimated spacecraft position, the center of the asteroid is projected in the image (indicated as c_{ast}^{img}). Then, the expected area covered by the asteroid in the FOV is computed. For a spherical approximation, this area is represented by a circle with radius:

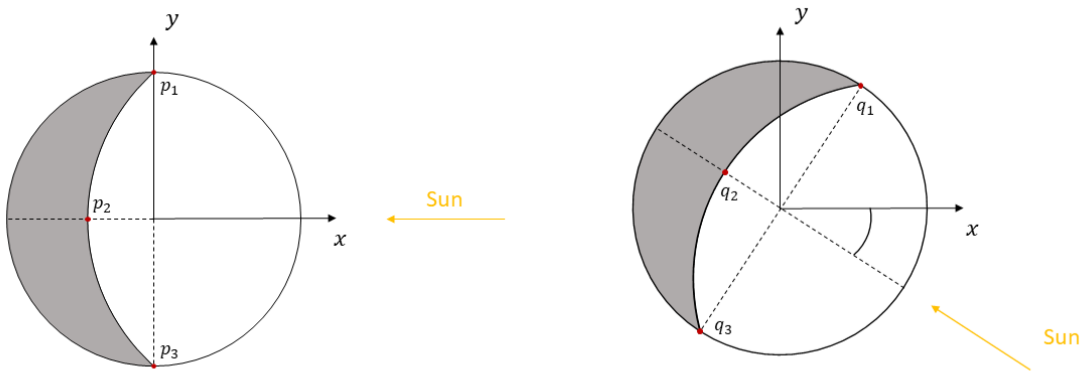
$$r_{ast}^{img} = \frac{\tan^{-1}\left(\frac{r_{ast}}{d}\right)}{\Omega} \tag{1}$$

where r_{ast} is the mean radius of the asteroid, d is the distance of the spacecraft from the asteroid, and Ω is the instantaneous field of view (IFOV) of the camera. To approximate the illuminated area of the asteroid surface, it is assumed that the terminator can be modeled as the arc of a circle. Considering a 2D reference frame centered in c_{ast}^{img}, if the Sun is aligned with the positive x direction the terminator arc passes through the points (considering again a spherical approximation):[11]

$$\begin{aligned} p_1 &= (0, r_{ast}^{img}) \\ p_2 &= (-r_{ast}^{img}\cos(\Psi), 0) \\ p_3 &= (0, -r_{ast}^{img}) \end{aligned}$$

where Ψ is the phase angle (Sun-asteroid-spacecraft angle). Then, the Sun direction is projected in the image and points p_1, p_2, and p_3 are rotated according to the angle between the Sun direction and the horizontal image direction. The procedure is illustrated in Figure 2. Finally, a circle is fit to the rotated points and its center is shifted according to the position of the asteroid center in the image.

At this point, the bounding box (BB) surrounding the asteroid is computed and subdivided in a set of non-overlapping windows of the desired size. Among these, only the windows in the illuminated region are retained. Specifically, if $\Psi < 90\,\mathrm{deg}$, windows are retained if they fall inside both the



(a) Terminator with Sun along positive x direction.

(b) Rotated terminator points.

Figure 2: Terminator approximation using a circle arc.

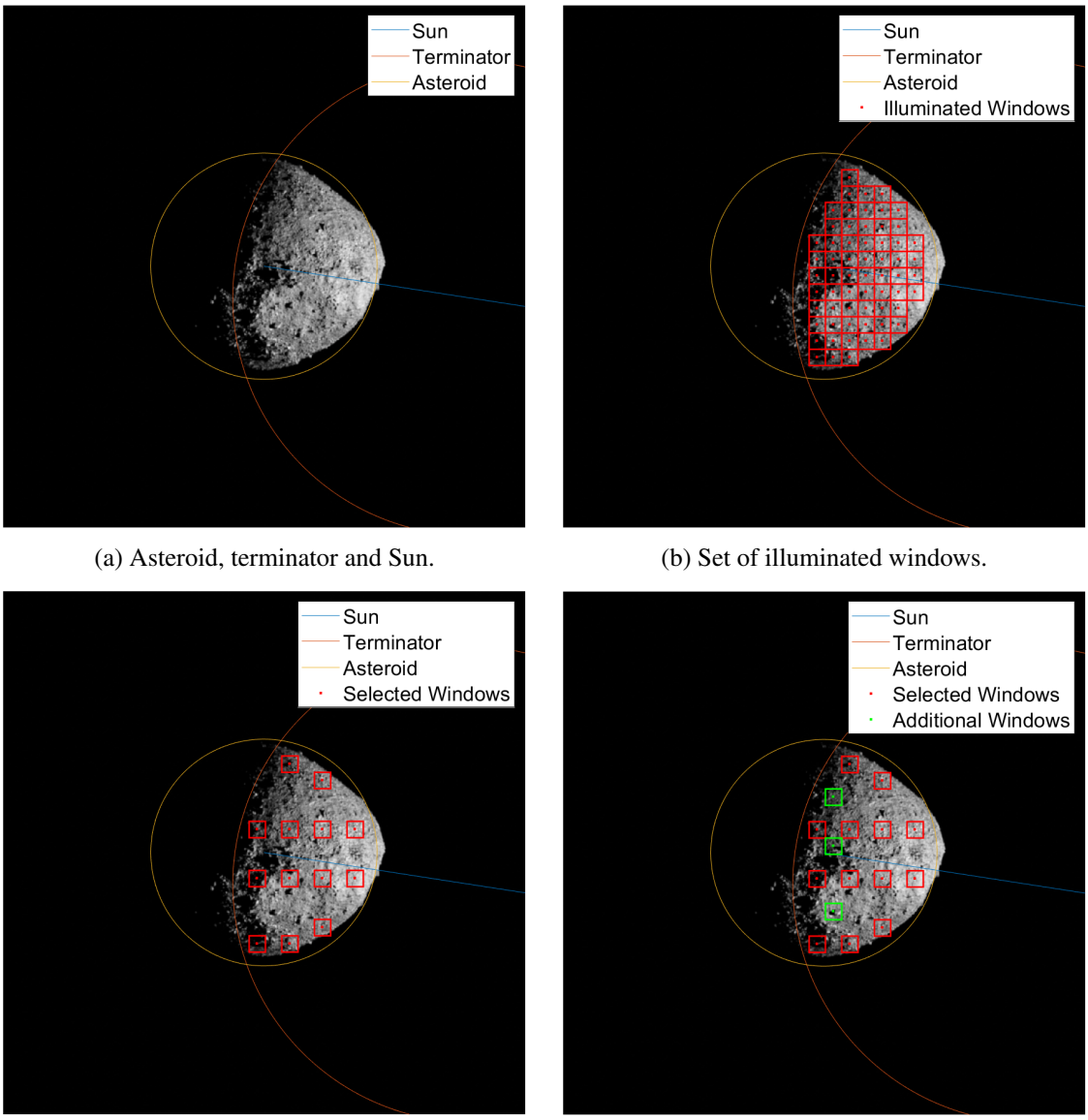
asteroid and the terminator circle. Otherwise, windows are retained if they fall inside the asteroid circle, but outside of the terminator one. To account for uncertainty and for the irregular shape of the asteroid, a margin is considered on the circle radii. Finally, a further check is performed to keep only windows that are farther than a minimum distance from the image borders. Among the remaining windows, a spread set needs to be selected. To do this, a BB containing all the illuminated windows is computed and divided into an equally spaced set of points. The number of points in equal to the number of windows to be extracted. For each point, the closest window is selected. If the asteroid occupies a small portion of the FOV, the same window may be selected multiple times. In this case, a further step of the algorithm selects additional windows until the desired number is reached. These additional windows are selected within the illuminated area by maximizing the minimum distance from the windows already selected.

The steps of the algorithm are illustrated in Figure 3. It is highlighted that this procedure is not optimal, in the sense that windows are not always as spread as possible, but it is fast and thus is a suitable compromise given the objective to limit computational cost. In accordance with the characteristics of the star trackers considered in the STARNav study, the algorithm is currently set to extract 16 windows containing 32x32 pixels each.

Reprojection of window positions The reprojection of window positions from the previous image to the current one is a more straightforward procedure. For each window, given the position of its center in an image and the spherical approximation of the asteroid, a 3D point can be computed by intersecting the camera ray passing through the window center and the sphere. This 3D point can then be projected in a successive view of the scene.

The delta pose between the preceding and the current camera views is computed as:

$$\mathrm{T}_{C_k \backslash C_{k-1}} = \begin{bmatrix} \mathrm{R}_{C_k \backslash C_{k-1}} & -\mathbf{r}^{C_k}_{C_k \backslash C_{k-1}} \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{2}$$



(a) Asteroid, terminator and Sun.

(b) Set of illuminated windows.

(c) Selection of spread windows.

(d) Additional windows.

Figure 3: Algorithm for window positions determination.

$$\mathrm{R}_{C_k \backslash C_{k-1}} = \mathrm{R}_{C_k \backslash A} \mathrm{R}^{\mathrm{T}}_{C_{k-1} \backslash A} \tag{3}$$

$$\mathbf{r}^{C_{k-1}}_{C_k \backslash C_{k-1}} = \mathrm{R}_{C_{k-1} \backslash A} \left(\mathbf{r}^{A}_{C_k} - \mathbf{r}^{A}_{C_{k-1}} \right) \tag{4}$$

where $\mathrm{R}_{Y \backslash X}$ indicates the direction cosine matrix (DCM) from a reference frame X to Y, C_{k-1} and C_k represent the camera reference frames in correspondence of the previous and current views, respectively, A is the asteroid-fixed reference frame, $\mathbf{r}^{C_k}_{C_k \backslash C_{k-1}}$ is the position of the camera at the second view with respect to the first one, expressed in the C_k reference frame, $\mathbf{r}^{A}_{C_k}$ is the position of the camera at the current view expressed in the asteroid-fixed frame, and $\mathrm{T}_{C_k \backslash C_{k-1}}$ is the matrix expressing the pose change between the two camera views. The camera frame is defined with

the positive z axis pointing towards the camera boresight and the x and y axes aligned with the horizontal and vertical image directions, respectively.

The position of window centers in the C_k reference frame can then be obtained:

$$ {}^{H}\mathbf{r}_{W}^{C_k} = \mathrm{T}_{C_k \backslash C_{k-1}} \begin{bmatrix} \mathbf{r}_{W}^{C_k - 1} \\ 1 \end{bmatrix} \tag{5} $$

$$ \mathbf{r}_{W}^{C_k} = \frac{{}^{H}\mathbf{r}_{W}^{C_k}(1{:}3)}{{}^{H}\mathbf{r}_{W}^{C_k}(4)} \tag{6} $$

where ${}^{H}\mathbf{r}_{W}^{C_k}$ is the homogeneous vector from which the position of the window centers in the current view, $\mathbf{r}_{W}^{C_k}$, is computed, and $\mathbf{r}_{W}^{C_k - 1}$ is the position of the window centers in the previous camera view. Finally, the coordinates of the window in the current image are obtained by projection:

$$ {}^{H}\mathbf{r}_{W}^{img} = \mathrm{K}\mathbf{r}_{W}^{C_k} \tag{7} $$

$$ \mathbf{r}_{W}^{img} = \frac{{}^{H}\mathbf{r}_{W}^{img}(1{:}2)}{{}^{H}\mathbf{r}_{W}^{img}(3)} \tag{8} $$

where K is the camera intrinsic matrix.

Image Processing

The image processing section of the algorithm comprises feature detection, feature tracking and motion estimation. As mentioned before, feature detection is based on the Harris-Stephens algorithm,[6] which finds corner points in the image. It has been chosen over other feature detectors because it provides a good compromise between computational cost and accuracy. The algorithm is currently set to extract at most 5 features per window. It is worth noting that in some cases, especially near the terminator regions, some of the extracted windows may contain mostly dark pixels because of the irregularity of the asteroid surface. To increase robustness, features are extracted only if at least a certain number of pixels is above a predefined threshold. Currently, 40% of the window pixels must be above 10% of the maximum possible image value to proceed with feature extraction.

Feature tracking relies on the KLT algorithm. In particular, the pyramidal implementation of the algorithm is used.[7] For tracking, a 9x9 patch is considered around the selected Harris features and is input to the KLT algorithm. In the latter, only 2 pyramid levels are used given the limited size of image windows (32x32 pixels). Furthermore, the windows reprojection procedure provides, in practice, a first guess for the KLT tracker and already accounts for the larger part of feature motion between two consecutive images. Tracking is carried out window by window. Then, the successfully tracked features are aggregated and undistorted before the subsequent outlier rejection step. The latter is based on the MSAC algorithm, a variant of RANSAC.[10] The direction of motion is iteratively computed by considering random subsets of tracked features, and the one with the largest consensus is considered as the best estimate. The computed direction of motion is considered valid only if at least 30% of the input tracked points are inliers. The number of MSAC iterations is not fixed, but varies depending on the number of tracked features and the desired confidence value, that is currently set to 99%. However, a maximum number of iterations is specified (currently 100).

It is assumed that an estimate of the inertial attitude of the spacecraft is available (e.g., from star tracker measurements). From that, the rotation between the two camera views can be retrieved,

while the direction of motion is computed using the algorithm presented in Reference 12. This algorithm finds a maximum likelihood estimate by minimizing a cost function based on the weighted Sampson distance. The output is an estimate of the direction of motion together with an associated covariance matrix, which in theory could be directly fed to the following EKF as the measurement covariance. However, numerical simulations have shown that the EKF performs better using a manually tuned measurement covariance matrix.

Extended Kalman Filter

The navigation filter used is an Extended Kalman Filter (EKF). Since the measurement processed by the EKF is a relative measurement (direction of motion between two camera views), the stochastic cloning technique is used.[13] In order to account for cross-correlations between position estimates at the different time instants involved in the relative measurement, a copy of the position estimate at the time of the previous image is appended to the state vector. Then, when a relative measurement is processed, the copy is discarded and it is replaced with a new one taken in correspondence of the new image.

The EKF propagation step integrates the equations of motion of the spacecraft in the inertial reference frame using a Runge-Kutta 4 integrator. The EKF dynamical model accounts for the central gravity of the asteroid and solar radiation pressure (SRP). The truth trajectory also includes the effects of the Sun's gravitational attraction and the J2 perturbation produced by the asteroid. The EKF propagation equations for the state and covariance matrix are:

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v} \\ -\mu \frac{\mathbf{r}}{||\mathbf{r}||^3} + C\, d_{AU}^2\, C_R\, \frac{A}{m}\, \frac{\mathbf{r_S}}{||\mathbf{r_S}||^3} \\ 0_{3\times 1} \end{bmatrix} \tag{9}$$

$$\overline{\mathrm{P}}_k = \Phi(t_k, t_{k-1}) \mathrm{P}_{k-1} \Phi(t_k, t_{k-1})^{\mathrm{T}} + \mathrm{Q} \tag{10}$$

where $\mathbf{r}$ and $\mathbf{v}$ are the spacecraft position and velocity, μ is the asteroid's gravitational parameter, C is the pressure of solar radiation at 1 AU, d_{AU} is the distance corresponding to 1 AU, C_R and A are the average reflectivity coefficient and exposed area of the spacecraft, m is the spacecraft mass, $\mathbf{r_S}$ is the position of the spacecraft with respect to the Sun, $0_{3\times 1}$ is a 3×1 vector of zeros, $\overline{\mathrm{P}}_k$ is the a priori covariance matrix at t_k, P_{k-1} is the a posteriori covariance matrix at t_{k-1}, $\Phi(t_k, t_{k-1})$ is the state transition matrix (STM) from t_{k-1} to t_k and Q is the process noise covariance matrix. To reduce the computational effort, the STM is computed through a second order approximation:[14]

$$\Phi(t_k, t_{k-1}) = \mathrm{I}_{9\times 9} + \mathrm{J}_{k-\frac{1}{2}} \Delta t + \mathrm{J}_{k-\frac{1}{2}}^2 \frac{\Delta t^2}{2} \tag{11}$$

where $\mathrm{I}_{n\times n}$ indicates an $n \times n$ identity matrix, $\Delta t = t_k - t_{k-1}$ is the EKF time step and $\mathrm{J}_{k-\frac{1}{2}}$ is the Jacobian of the equations of motion evaluated at $t = t_{k-1} + \frac{\Delta t}{2}$, and $\mathrm{J}_{k-\frac{1}{2}}^2 = \mathrm{J}_{k-\frac{1}{2}} \mathrm{J}_{k-\frac{1}{2}}$. The process noise matrix is computed according to the state noise compensation model described in Reference 15:

$$\mathrm{Q} = \begin{bmatrix} q\frac{\Delta t^3}{3}\mathrm{I}_{3\times 3} & q\frac{\Delta t^2}{2}\mathrm{I}_{3\times 3} \\ q\frac{\Delta t^2}{2}\mathrm{I}_{3\times 3} & q\Delta t\, \mathrm{I}_{3\times 3} \end{bmatrix} \tag{12}$$

in which q is a fixed tuning parameter that in this work is set to $q = 10^{-12}\,\mathrm{m}^2\,\mathrm{s}^{-3}$.

The EKF measurement model is given by:

$$\mathbf{h}(\mathbf{x}) = \mathrm{R}_{C_k\backslash A_k} \frac{\mathrm{R}_{A_k\backslash N}\bar{\mathbf{r}}_k - \mathrm{R}_{A_{k-1}\backslash N}\mathbf{r}_{k-1}}{||\mathrm{R}_{A_k\backslash N}\bar{\mathbf{r}}_k - \mathrm{R}_{A_{k-1}\backslash N}\mathbf{r}_{k-1}||} \tag{13}$$

where N is the inertial reference frame, $\bar{\mathbf{r}}_k$ is the a priori position estimate of the EKF at t_k and $\mathbf{r}_{k-1}$ is the stochastic copy of the position estimate at t_{k-1}. It is remarked that the asteroid-fixed reference frame is known, since the asteroid is assumed to be characterized. Furthermore, as mentioned before it is assumed that an estimate of the inertial attitude of the spacecraft is available, from which the attitude with respect to the asteroid-fixed frame is then computed as:

$$\mathrm{R}_{C_k\backslash A_k} = \mathrm{R}_{C_k\backslash N}\mathrm{R}_{N\backslash A_k} \tag{14}$$

The inertial attitude matrix used by the EKF is perturbed by random noise with a standard deviation of 10 arcsec for each body axis. This value is on the order of magnitude of typical star tracker measurement errors.

Finally, a measurement editing procedure based on the Mahalanobis distance of the innovation is implemented to reject grossly incorrect measurements that may be input to the EKF.

Synthetic Images Generation

The image generation procedures relies on the Celestial Objects Rendering Tool (CORTO), which is a Blender-based rendering tool developed at Politecnico di Milano's DART Lab.[16] Blender uses a a physically-based rendering engine named Cycles*. CORTO takes as input a set of spacecraft poses (i.e. position and attitude) and the position of the Sun, and produces a realistic image of the scene observed by the camera. The output is an error-free image respecting the observation geometry and the light-celestial body interaction as defined by a scattering law and the associated celestial object characteristics. The relevant camera effects are applied a posteriori. First, the image is distorted according to the characteristics of the camera optics. Then, it is convoluted with a PSF to reproduce focusing errors. Finally, relevant detector noises (such as dark current, readout noise, and response non-uniformity) are added. The procedure is illustrated in Figure 4.

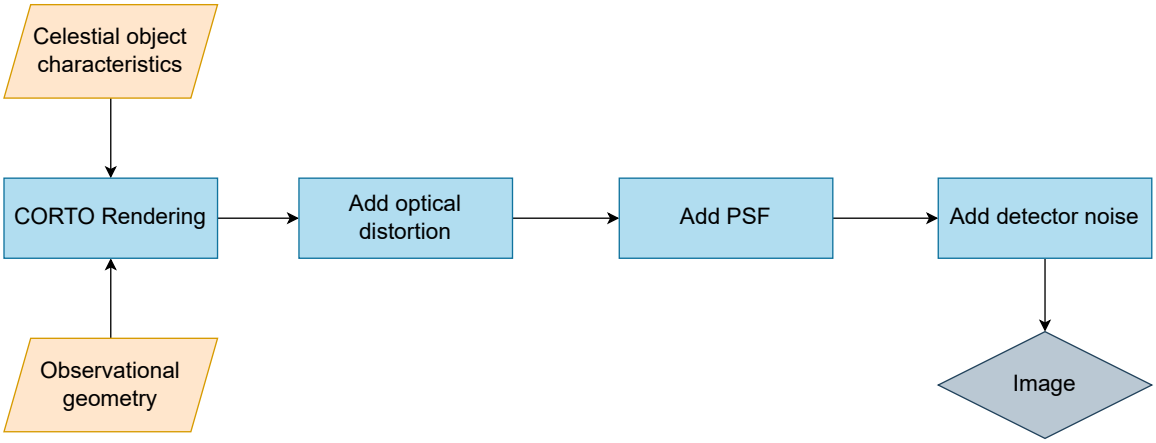


Figure 4: Procedure for the generation of synthetic images.

*https://www.cycles-renderer.org/

In this work, a Lambertian scattering law is used as it was found to give the best compromise between rendering time and image accuracy. Images of Bennu are rendered using a high-fidelity shape model produced by the OSIRIS-REx mission, on top of which a texture map is applied. The considered camera has a FOV of 20.6 deg and a focal length of 50.7 mm. An example of the resulting images at the farthest and closest distance considered is shown in Figure 5.

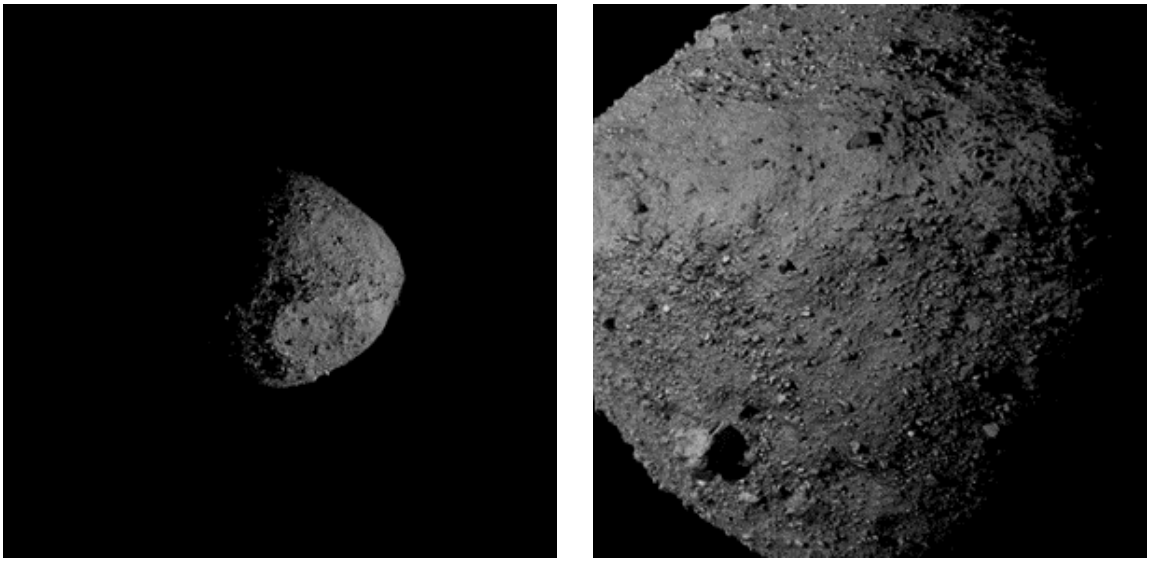


Figure 5: Examples of synthetic images used for numerical simulations.

RESULTS

In this section the results of the analyses carried out in the design and validation phase of the algorithm are presented. First, a dedicated analysis has been executed to select a suitable value for the EKF time step, which corresponds to the Δt between two consecutive images. Different values have been tested and the most appropriate one was used to generate the image sequence in the test scenario around Bennu. The performance of the algorithm has been validated with a Monte Carlo campaign, which also tested its robustness with respect to various uncertain parameters. Finally, the computational time required by the algorithm has been compared with a baseline implementation that processes the full input images.

EKF Time Step Selection

The EKF time step analysis has the objective of identifying a suitable Δt value. Indeed, the latter represents the amount of time between two consecutive images, which directly affects the performance of the image processing algorithm. Large Δt values may reduce the feature tracking accuracy because of the larger motion between the images and the larger appearance change of features. On the other hand, small Δt values may cause only very small changes between the images, so that not enough information could be extracted to perform motion estimation reliably.

The analysis is carried out considering two test cases in opposite conditions along the considered flyby trajectory. The first is the farthest point from the asteroid, while the other is the closest one. For each of them, a reference image is generated at an initial time t, from which features are extracted.

Then, features are tracked over the subsequent image, that is rendered at time $t + \Delta t$. Finally, the motion estimation step is performed. The following set of Δt values has been considered:

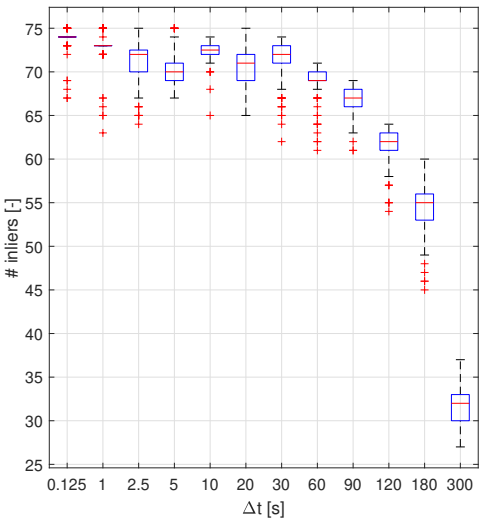
$$S_{\Delta t} = \{0.125, 1, 2.5, 5, 10, 20, 30, 60, 90, 120, 180, 300\} \text{ s}$$

The set spans from very small to relatively large values of Δt. The lower values should cover cases with very small appearance changes in the images, as expected given the relatively slow dynamics of spacecraft around small bodies. The upper ones, instead, are large enough to cause significant feature motion.

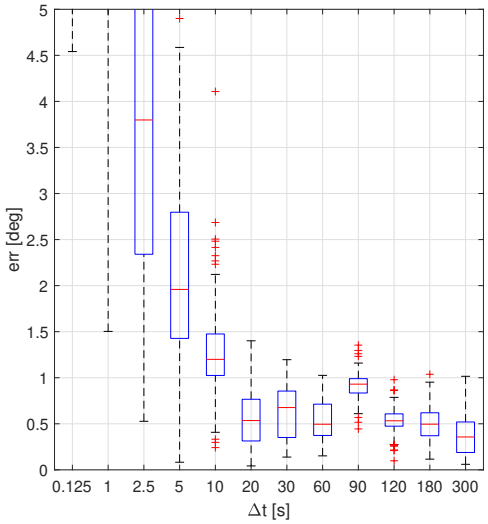
The feature tracking and motion estimation steps are repeated 100 times for each value in the considered set, so to have some indication about their stochastic variability. Two metrics are analyzed: the number of inliers after MSAC and the accuracy of the motion estimation. The former is associated with the accuracy and reliability of the image processing. A larger number of inliers is typically indicative of good image processing performance, and also guarantees more robustness in the subsequent steps as motion estimation is well constrained. The latter is the measurement provided to the EKF, so its level of accuracy is the main factor in the analysis. The motion estimation error is computed as the angle between the estimated direction of motion and the true one.

Results are illustrated in Figure 6 and Figure 7 for the far and close case, respectively. They are reported in the form of box plots obtained from the 100 runs performed for each pair of images. The motion estimation error plots have been cropped between 0 and 5 degrees for clarity. In general, the two test cases show similar trends. The number of inliers initially remains relatively constant and then starts decreasing as Δt increases, which is expected since this implies a larger feature displacement between the two images. However, even for the largest Δt considered a sufficient number of inliers is retained. For a given Δt, the close case is characterized by a lower number of inliers with respect to the far one. This is probably due to two factors that make feature tracking more difficult. First, when the spacecraft is closer to the asteroid it has larger velocity, implying a larger displacement of features for a fixed time step. Second, changes in feature appearance are more significant at closer distance. The motion estimation error also shows interesting trends. For very small values of Δt, motion estimation is dominated by noise, giving very large errors despite the high number of inliers. As mentioned before, this is expected since images are taken so close that there is very limited change in the observed scene, and no information can be extracted to estimate the spacecraft motion. Then, as Δt increases, the performance improves. Here some difference is observed between the far and the close case. For the former, the motion estimation error keeps getting smaller up to $\Delta t = 20$ s, after which a plateau is reached. For the latter, instead, there is a minimum between 90 s and 120 s, after which the error starts growing. This is again explained by the fact that, for large Δt values, feature tracking has a larger performance drop at close distance. On the other hand, when the spacecraft is closer the asteroid occupies a larger portion of the camera FOV, leading to better motion estimation with respect to the far case. However, in both cases the motion estimation error is quite small, and remains below 1 deg even for the largest Δt.

The above analysis shows that values of Δt larger than 20 s give reasonable motion estimation performance for these test cases. A slight, further improvement is seen for values between 60 and 180 s. All considered, the value selected for subsequent analyses is $\Delta t = 60$ s. Indeed, this gives both good motion estimation accuracy and a high number of inliers. Furthermore, as long as the motion estimation performance remains reasonable, it is desirable to have a smaller Δt, as this ensures more robustness in cases in which, for example, feature tracking may fail between two images or a measurement could be rejected by the Mahalanobis test implemented in the EKF.

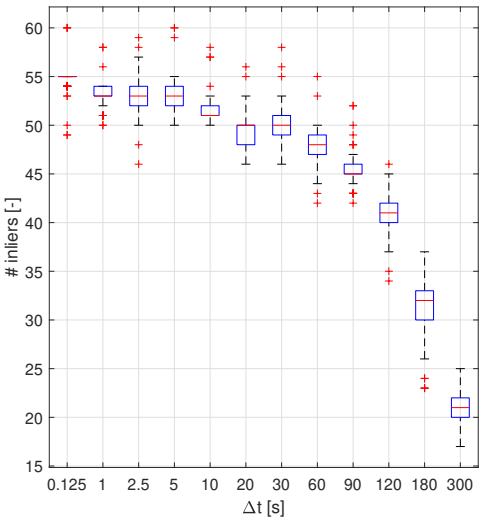


(a) Number of inliers.

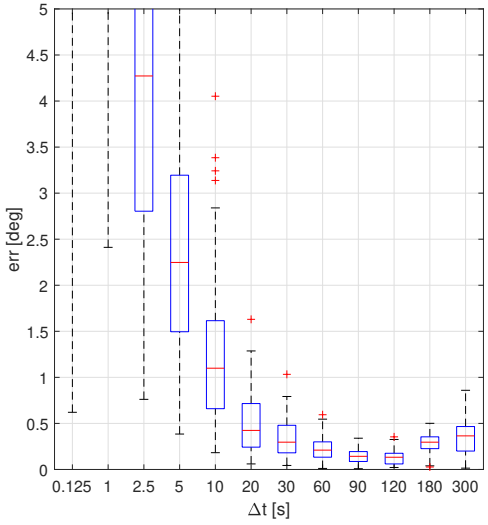


(b) Motion estimation error.

Figure 6: EKF time step analysis: far case results.



(a) Number of inliers.



(b) Motion estimation error.

Figure 7: EKF time step analysis: close case results.

Monte Carlo Campaign

The scenario in which the algorithm is tested is a slow flyby of asteroid Bennu. The reference trajectory starts at a distance of 3.32 km from the target, with a phase angle of 75 deg. Given the camera characteristics, at this distance the asteroid occupies a relatively small portion of the FOV, which is a challenging condition for motion estimation. The closest approach, at a distance of 1.25 km, is reached after 1 day, with a phase angle of 60 deg. After closest approach the phase angle quickly increases, making feature tracking difficult since most of the asteroid surface is shadowed. The simulated trajectory lasts 1.2 days in total. It is shown in Figure 8 in an inertial reference frame centered in the asteroid center of mass.

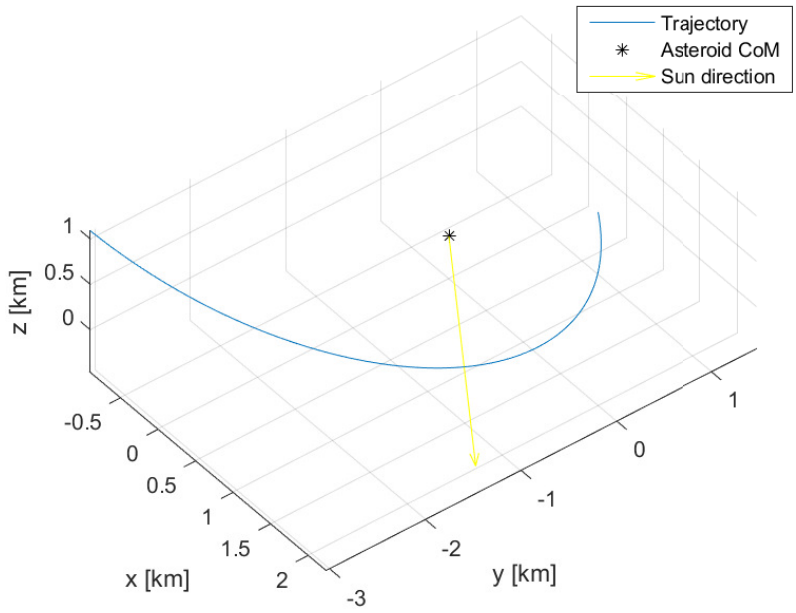


Figure 8: Reference flyby trajectory in the inertial frame.

As mentioned before, it is assumed that the flyby is carried out after an initial characterization phase. Therefore, the asteroid mean radius and the inertial direction of the spin axis are assumed known. The latter is used to define the asteroid-fixed reference frame, which has its z axis aligned with the spin direction. The other two axes can be fixed arbitrarily on the equatorial plane.

A Monte Carlo campaign is carried out to assess the performance of the algorithm considering the relevant sources of uncertainty. The latter include the initial position and velocity of the spacecraft, the camera calibration parameters and the orientation of the camera within the spacecraft. For each of them, a Gaussian error distribution with zero mean is assumed, while the standard deviation values are listed in Table 1. It is noted that only a single set of synthetic images is generated for the MC campaign and used for all test cases. Then, for each run the camera parameters used in the VBN algorithm are perturbed according to the associated uncertainty. Finally, as mentioned before, the spacecraft attitude is also affected by a Gaussian uncertainty of 10 arcsec on each axis.

100 simulations have been carried out as part of the MC campaign. Figure 9 shows the performance of the EKF over all the MC runs in the inertial frame. The results show good agreement

Table 1: Sources of uncertainty considered in the MC campaign.

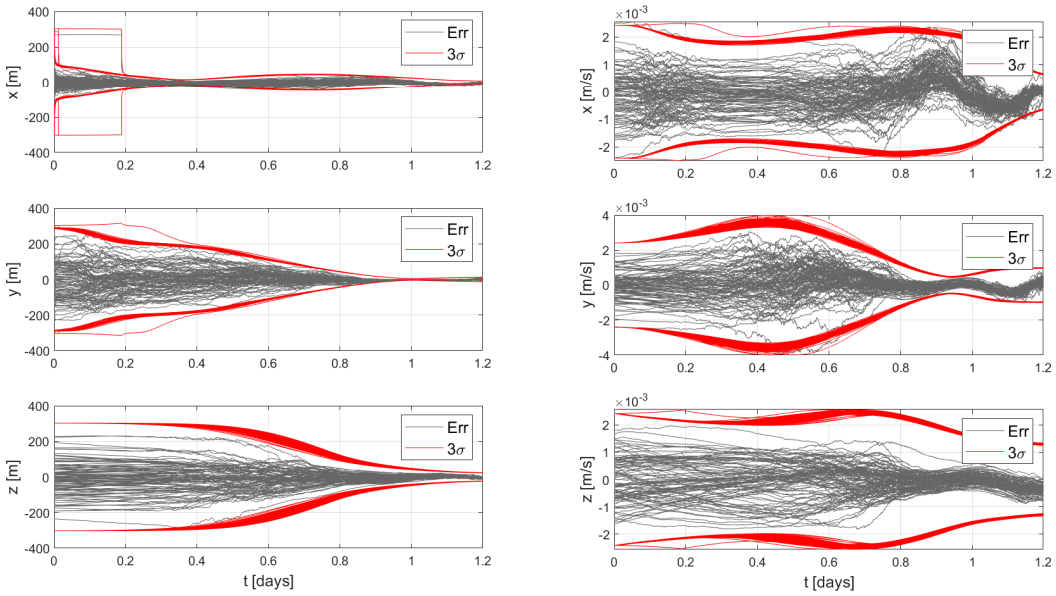
Uncertainty	σ
Initial position	3% of true range (for each component)
Initial velocity	2% of the norm of the inertial velocity vector (for each component)
Focal length calibration residual	1 px
Optical centre calibration residual	0.5 px
Camera orientation in body frame	100 arcsec on each axis
Radial distortion coefficients	1%

between the estimation error and the expected covariance. The y and z directions are characterized by a slow convergence of the EKF, while there is a quick drop of the uncertainty in the x direction. This is due to the fact that the direction of motion of the spacecraft is almost aligned with the x axis direction, that is then significantly more constrained than the others. While in most cases there is an immediate drop in the uncertainty, there are a couple of instances in which it happens later. In these cases the initialization error on the spacecraft position causes most of the windows to be initially extracted outside of the asteroid, so that the image processing is not able to produce a valid measurement. As soon as a valid measurement is produced and processed by the filter, the covariance is reduced. At closest approach, after 1 day, all the position components are well estimated, with errors in the order of 10-20 m. The velocity estimation error shows a systematic behaviour towards the final part of the simulation, probably due to a combination of factors. First, the fact that the same set of images is used for all MC runs could limit the variability in the image processing output. Second, the final part of the trajectory is characterized by challenging illuminations conditions due to the high phase angle, as mentioned before, which could produce some bias in the motion estimation.

To give further insight, Figure 10 shows the position estimation error in the camera reference frame, together with a zoom of the plot. Similar trends to the ones highlighted above are observed. Also in the camera frame the x direction is more constrained than the others, meaning that the spacecraft motion is mostly aligned with the horizontal direction of the images. The zoomed plots show that the systematic behaviour of the error also affects the position estimation, especially in the x direction, which also exceeds slightly the expected 3σ bounds. This is consistent with the fact that the challenging illumination conditions degrade the motion estimation accuracy. Figure 11 shows the last image of the simulated sequence. Only a small portion of the asteroid surface is visible, and with significant shadowing from boulders. Not only the high phase angle makes feature tracking more difficult, but since only a small portion of the asteroid is illuminated, motion estimation is also worse because features are concentrated in a small region of the image plane. As a result, the estimation error is not fully captured by the expected covariance.

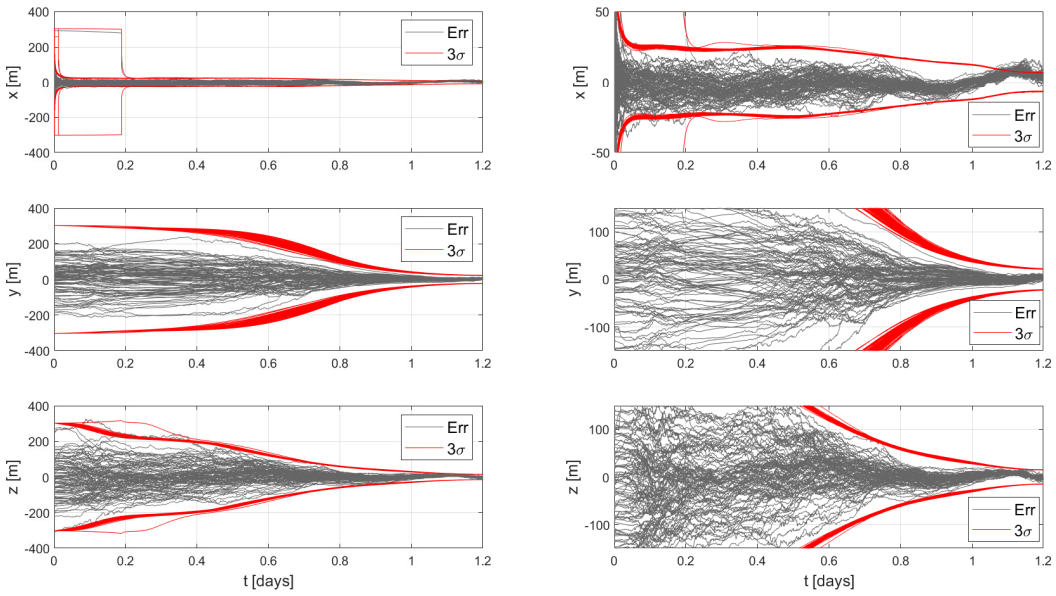
Timing Comparison

The window-based algorithm is compared with a baseline version that performs feature tracking and motion estimation using the entire image. Timing information are retrieved from a MATLAB



(a) Position estimation error. (b) Velocity estimation error.

Figure 9: EKF performance for the MC campaign in inertial frame.



(a) Position estimation error. (b) Zoom of position estimation error.

Figure 10: EKF performance for the MC campaign in camera frame.

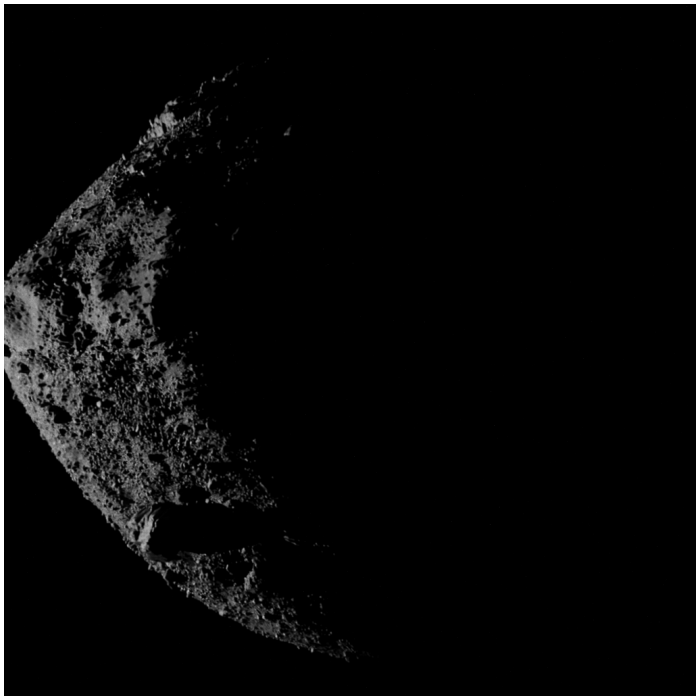
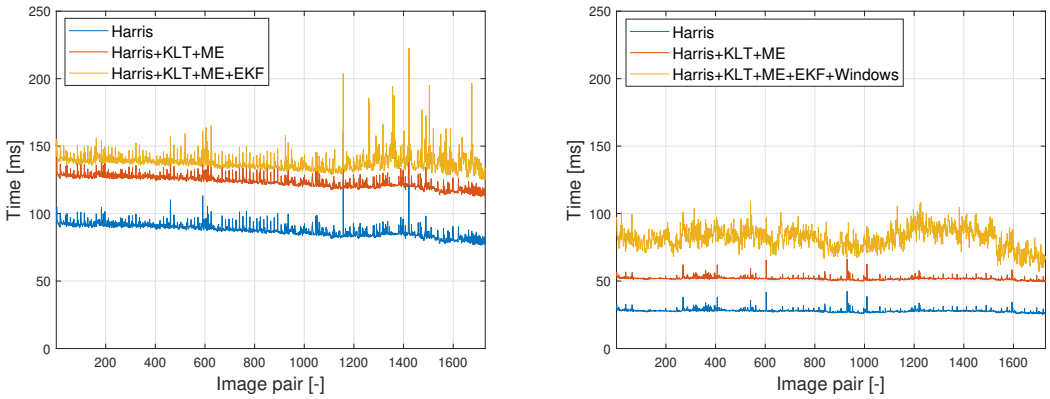


Figure 11: Last image of the sequence.

implementation on a desktop computer, so the analysis only serves as a relative comparison between the two versions of the algorithm, and it is not intended to assess the absolute amount of time required to run the algorithm. The settings of the two algorithm versions are analogous, so that for example the total number of features tracked is the same. The only difference is that the baseline version the KLT algorithm is set to use 4 image pyramid levels, which is necessary in order to successfully track features when there are significant changes among the images.

Figure 12 reports the timing results for the standard and window-based VBN algorithms over a full image sequence on the flyby trajectory. For the baseline version, the feature extraction step using the Harris detector is the most expensive part. On average, a full cycle of the baseline algorithm takes 138.1 ms, of which 78.6 are dedicated to feature extraction. KLT and motion estimation (ME) take on average 39.2 ms, and the rest is dedicated to the EKF. Some peaks can be observed, which typically correspond to cases in which a larger number of iterations were required for the KLT or MSAC convergence. On the other hand, for the window-based algorithm the computational cost is more equally spread. There is a significant reduction in the time required for feature extraction, since a smaller number of image pixels need to be processed. The average time needed for feature extraction is 25.8 ms, less than one third of the baseline algorithm. The KLT and ME steps have a similar cost, with an average time of 37.1 ms. The cost in this section of the algorithm is dominated by the number of features that need to be tracked and by the size of feature patches, which are the same in the two versions. The total amount of time required for a cycle of the window-based algorithm is on average 81.6 ms, including the EKF and the computation of window positions. Thus, the window-based algorithm is characterized by an average reduction of the computational time of 40.9%, while keeping the same level of performance of the standard version and also potentially reducing storage requirements, since only windows need to be stored rather than the entire image.



(a) Standard algorithm.

(b) Window-based algorithm.

Figure 12: Timing analysis for the standard and window-based algorithms.

CONCLUSION

This work presents a resource-constrained VBN algorithm based on the feature tracking and visual odometry techniques. By extracting patches or windows from the input images, the required computational effort is significantly reduced, especially in the feature extraction step, which represents the most expensive part of the standard algorithm. At the same time, the algorithm retains a high level of accuracy, as demonstrated by a MC campaign in which various sources of uncertainty have been considered, including camera calibration parameters.

Future work will be directed towards increasing the robustness of the algorithm. As shown by the MC campaign, if the navigation error is particularly large the window extraction procedure is affected, and windows can be extracted in regions of the image not containing the asteroid. This, in turn, can prevent the image processing from producing valid measurements. This problem could be tackled in various ways. For example, a backup mechanism could be implemented to recompute the window positions when they are extracted outside of the asteroid. Another possibility would be to first process the entire image to find the asteroid bounding box, within which windows would then be extracted. Additionally, the EKF implementation could be improved by including the measurement bias and residual accelerations as solve for states or considered parameters. Furthermore, a variable model for the measurement covariance could be introduced to account for the reduction in motion estimation performance towards the final part of the trajectory. To conclude, the algorithm will be tested in other scenarios, including small bodies with a more irregular shape, such as Itokawa.

ACKNOWLEDGMENTS

This work has been conducted under ESA Contract No. 4000139932/22/NL/CRS within the General Support Technology Programme (GSTP) through the support of the national delegation of Italy (ASI). The authors would like to acknowledge the support received by Simone Becucci, Damiano Macchi, Ranieri Marchi, Andrea Sica and Marcella Belcari in the context of the STAR Nav study.

REFERENCES

[1] R. Gaskell, O. Barnouin, M. Daly, E. Palmer, J. Weirich, C. Ernst, R. Daly, and D. Lauretta, "Stereophotoclinometry on the OSIRIS-REx Mission: mathematics and Methods," *The Planetary Science Journal*, Vol. 4, No. 4, 2023, p. 63.

[2] E. Goh, H. S. Venkataram, M. Hoffmann, M. D. Johnston, and B. Wilson, "Scheduling the NASA deep space network with deep reinforcement learning," *2021 IEEE Aerospace Conference (50100)*, IEEE, 2021, pp. 1–10.

[3] T. Kubota, M. Otsuki, and T. Hashimoto, "Touchdown dynamics for sample collection in Hayabusa mission," *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 158–163.

[4] G. Ono, F. Terui, N. Ogawa, S. Kikuchi, Y. Mimasu, K. Yoshikawa, H. Ikeda, Y. Takei, S. Yasuda, K. Matsushima, *et al.*, "GNC strategies and flight results of Hayabusa2 first touchdown operation," *Acta Astronautica*, Vol. 174, 2020, pp. 131–147.

[5] C. Norman, C. Miller, R. Olds, C. Mario, E. Palmer, O. Barnouin, M. Daly, J. Weirich, J. Seabrook, C. Bennett, *et al.*, "Autonomous navigation performance using natural feature tracking during the OSIRIS-REx Touch-And-Go sample collection event," *The Planetary Science Journal*, Vol. 3, No. 5, 2022, p. 101.

[6] C. Harris and M. Stephens, "A combined corner and edge detector," *Alvey vision conference*, Vol. 15, Citeseer, 1988.

[7] J.-Y. Bouguet *et al.*, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel corporation*, Vol. 5, No. 1-10, 2001, p. 4.

[8] P. Michel *et al.*, "The ESA Hera mission: detailed characterization of the DART impact outcome and of the binary asteroid (65803) Didymos," *The planetary science journal*, Vol. 3, No. 7, 2022, p. 160.

[9] C. Balossi *et al.*, "Moon Limb-Based Autonomous Optical Navigation using Star Trackers," *46th Annual AAS Guidance, Navigation and Control Conference*, 2024.

[10] P. H. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer vision and image understanding*, Vol. 78, No. 1, 2000, pp. 138–156.

[11] Q. Changeat and A. Al-Refaie, "TauREx3 PhaseCurve: a 1.5 D model for phase-curve description," *The Astrophysical Journal*, Vol. 898, No. 2, 2020, p. 155.

[12] J. A. Christian, L. Hong, P. McKee, R. Christensen, and T. P. Crain, "Image-based lunar terrain relative navigation without a map: Measurements," *Journal of Spacecraft and Rockets*, Vol. 58, No. 1, 2021, pp. 164–181.

[13] S. I. Roumeliotis and J. W. Burdick, "Stochastic cloning: A generalized framework for processing relative state measurements," *Proceedings 2002 IEEE International Conference on Robotics and Automation*, Vol. 2, IEEE, 2002, pp. 1788–1795.

[14] J. R. Carpenter and C. N. D'Souza, "Navigation filter best practices," tech. rep., 2018.

[15] B. Schutz, B. Tapley, and G. H. Born, *Statistical orbit determination*. Elsevier, 2004.

[16] M. Pugliatti, C. Buonagura, and F. Topputo, "CORTO: The Celestial Object Rendering TOol at DART Lab," *Sensors*, Vol. 23, No. 23, 2023, p. 9595.