A Robust Monocular Line-based Relative Navigation Approach for Spacecraft Rendezvous and Proximity Operations

Iason Georgios Velentzas*, Mehregan Dor†, and Panagiotis Tsiotras‡
Georgia Institute of Technology, Atlanta, GA, 30313, USA

Simultaneous Localization and Mapping (SLAM) algorithms have demonstrated considerable efficacy in ground robotics applications. However, their application to space-based scenarios is constrained by many challenges, such as harsh lighting, large distances between the observer and the target, as well as on-board computational limitations. With the goal of enhancing resilience against conditions encountered in orbit, we incorporate geometric primitives into a previously developed visual SLAM pipeline for the space environment and create a sophisticated keyframe selection metric, that is able to identify the keyframes online in a smart way. These modifications render our algorithm resilient to lighting variations and to the sudden or slow motion of the spacecraft. The algorithm is tested on synthetic datasets generated specifically to capture these characteristics, as well as real-life imagery from a NASA proximity operation to the Hubble Space Telescope.

I. Introduction

As the number of artificial satellites in close proximity to the Earth continues to rise, the ability to successfully conduct spacecraft rendezvous and proximity operations has become increasingly crucial. These proximity operations are essential for maintaining the safety and reliability of space assets, as well as for optimizing the use of space resources. A few applications of critical importance are active debris removal [1, 2], in-space assembly [3, 4], and satellite servicing [5]. In this work, we consider the problem of relative navigation between two spacecrafts in orbit. In this non-cooperative scenario, the spacecraft are located in different orbits and they do not share any information with each other. A key aspect of the problem at hand is the chaser spacecraft's ability to accurately estimate its relative pose, given that the challenging lighting conditions in space often result in the spacecraft losing track of the target.

Monocular cameras are a popular sensor of choice for space robotics applications. Despite being lightweight and inexpensive, they provide information-rich measurements and are robust to highly dynamic environments, where the environment conditions can change drastically and the targets are moving. As a result, our main focus is the Visual SLAM (VSLAM) problem where the agent maintains a map of the environment and localizes itself in it using only visual input on-the-fly. Typically, as denoted in [6], there are two different methods of solving the VSLAM problem: filter-based, and keyframe-based algorithms. In [7] the authors developed an Extended Kalman Filter solution to estimate the 3D structure of the target spacecraft and navigate around it, while in [8] the use of a Multiplicative Extended Kalman Filter is studied as a solution to the visual-based navigation problem. In both cases, the precision and the accuracy of the estimation are limited. On the other hand, keyframe methods solve a batch optimization (Bundle Adjustment) problem using a selected subset of the input images, called keyframes. An extensive comparison between the two methods has been conducted in [6], where it is concluded that, with enough processing power, the Bundle Adjustment (BA) approach is superior to filtering in terms of accuracy, despite being heavier in terms of computation. This outcome has shifted the focus of the literature in the past years toward Bundle Adjustment.

BA approaches can be separated into dense and sparse. Dense methods use the intensities of all pixels in the image, while sparse methods detect features in the image. They only use these pixels to perform BA, disregarding the rest of the information in the image. Sparse approaches are more preferable in space applications since dense approaches are computationally heavy and require a lot of processing power. A popular state-of-the-art sparse VSLAM algorithm is ORB-SLAM2 [9].

*Ph.D. Student, Department of Aerospace Engineering, Atlanta, USA, iason.velentzas@gatech.edu
†Ph.D. Student, Department of Aerospace Engineering, Atlanta, USA, mehregan.dor@gatech.edu
‡Professor, Department of Aerospace Engineering, Atlanta, USA, AIAA Fellow, tsiotras@gatech.edu

II. Prior Related Works

Keyframe Selection: The images that are captured by the camera and are given as an input to the system are called frames. An interesting aspect of VSLAM, which will be one of our focus areas, is the selection of keyframes, i.e. a subset of the frames that will be used to calculate the SLAM solution. A survey of keyframe-based monocular SLAM solutions [10] concludes that the two main parameters for a frame to be selected as a keyframe are either a significant pose change or a significant scene appearance change between the frames (or both). Other approaches use the number of features that are being matched between the frames [9] or the change in the relative brightness factor between the frames [11]. The majority of the research on this topic thus far is based on constant arbitrary thresholds that are hand-tuned, application specific, and do not account for challenges that a spacecraft will face in orbit, such as specular reflections [12]. To address this issue, we propose a novel keyframe selection scheme that incorporates multiple factors and utilizes problem-specific characteristics to intelligently decide in real-time, whether a frame should be considered as a keyframe.

Line-augmented pipelines: Another addition to the traditional VSLAM problem that is considered in our work, is the incorporation of geometric primitives in the 3D map, that the agent maintains as part of the SLAM solution. This map is a set of 3D landmarks, typically points, but in our case, it also contains 3D line segments. It is known that point correspondences suffer from poor matching performance in reflective environments [8]. Recent works in VSLAM leverage line segments to provide more stringent geometric constraints and hence, improve the robustness and efficiency of the solution. The use of line segments, however, increases the computational complexity both in the detection and the matching phases. Reference [13] describes why the Line Segment Detector (LSD) [14] wastes resources in detecting short line segments that do not provide useful information and will not yield matches in the subsequent frames. This stage of the pipeline can become a bottleneck for real-time implementation and hence, it is necessary to carefully design the detection, description, and matching processes. In [13], the authors use a hidden parameter tuning of the detector to enforce constraints on the line segments. This tuning exploits the fact that the line segments are not used for scene reconstruction, but only for pose estimation. To efficiently match line segments, the Line Band Descriptor (LBD) is developed in [15]. LBD is robust against positional changes along the line segment and is computationally efficient. Another important feature of the LBD that can be proved to be significant for space applications is the fact that by restraining each dimension of the descriptor to lower values, the descriptor is more resilient against non-linear illumination changes. In order to maintain real-time performance, the authors of [15] employ a Relational Graph Strategy, which enforces pairwise geometric constraints during the matching process.

In applications with easily recognizable geometric structures in the scene, layout information can increase the robustness of the estimation, especially in low-textured environments [16]. This approach has already been used in some terrestrial robotics applications in order to tackle the problem of Structure-from-Motion (SfM) [17]. We utilize advancements in this field to exploit the geometric structure of human-made spacecraft.

One of the earliest works in monocular VSLAM using lines is PL-SLAM [16]. The authors use point and line correspondences in the ORB-SLAM2 pipeline and integrate the line segments via their endpoints to formulate the 2D-2D and 2D-3D correspondences. PL-SLAM achieves better tracking performance than ORB-SLAM2, but it is restricted by the assumption of the so-called "Manhattan worlds", where all surfaces are aligned along three dominant directions. In order to maintain real-time performance, the authors of [16] employ strategies from [15], where geometric constraints are used to create a relational graph and accelerate the line matching process. A recent extension of this work for indoor environments is PLJ-SLAM [18], in which junctions of co-planar lines are also considered. PLJ-SLAM seems to have improved performance and lower execution time than PL-SLAM, while demonstrating increased robustness against blur and brightness variations.

Visual Perception in Orbit: Fewer works have actually applied SLAM-based algorithms to the spacecraft rendezvous scenario. The authors of [19] tackle the rendezvous and proximity operation problem by employing a sparse feature-based VSLAM method using the GTSAM library [20] and demonstrated functionality on the SPHERES test bed onboard the International Space Station. We note that, there is no notion of the keyframe concept in that paper, which assumes that the selection process of the image sequence is solved a priori. The results of that paper also show significant tracking error on specific parts of the trajectory. ORB-SLAM2 was applied to a similar spacecraft rendezvous scenario [21] on imagery from a real spacecraft rendezvous scenario. The results confirmed that, even with a state-of-the-art algorithm, the space environment's specific characteristics can cause loss of tracking of the target object and difficult initialization predicated on the need for both sufficient parallax angle and percentage of overlap of feature matches.

The difficulties in initialization and the loss of tracking described in [21] stem from competing effects borne from the geometry and the kinematic evolution encountered in the spacecraft circumnavigation problem. Typically, navigation cameras used for rendezvous have a small field of view and the distances between the chaser and target are relatively large compared to ground-based applications. In addition, the evolution of the chaser-target relative geometry is constrained by the typically slow free orbital motion of the chaser around the target. These two effects combined typically produce situations in which the parallax between successive camera images is insufficient to meet the minimum parallax requirement. Due to this small parallax between frames, obtaining successful keyframe insertion is difficult since the frame separation with respect to the last keyframe becomes too large to satisfy the minimum number of features matched. We therefore expect any algorithm predicated on matching features directly between successive keyframes to suffer from the same challenges in the spacecraft rendezvous scenarios. This includes ORB-SLAM2, and all of the derived algorithms that use the same tracking module, such as PL-SLAM [16], Stella-SLAM (formerly OpenVSLAM) [22], PLP-SLAM [23], etc.

To tackle these challenges, and building upon the previous work of Dor et. al. [24], which successfully solves the problem of relative navigation around an asteroid, we augment this pipeline with points and line segments for spacecraft rendezvous and proximity operations, which is targeted to maintain tracking of the observed satellite despite the adverse space environment's conditions.

Specifically, a) instead of directly matching feature points between successive keyframes, we employ a pairwise feature matching between successive frames, and then query matches between keyframes, which improves the number of surviving tracked matches. This improves the map initialization and mapping performance; b) we introduce a robust keyframe selection approach which uses problem-specific characteristics and inserts keyframes based on multiple heuristics with adaptive thresholds, diminishing the likelihood of ill-determined triangulation and optimization; and c) motivated by the success of PL-SLAM [16] in textureless environments, we enrich the map with line segments, thus improving the ability to maintain tracking of the target even in challenging situations. We test this framework using simulated and real datasets from the challenging scenario of relative navigation against a non-cooperative target spacecraft.

The paper is structured in the following way: Section III presents the mathematical formulation of the problem statement, while Section IV details the specific components of the solution algorithm. The datasets that are used to test our algorithm are being discussed in section V. Sections VI and VII present the results of the two main modules (FrontEnd and BackEnd) and the conclusions of this work, respectively.

III. Problem Statement

Consider two coordinate frames $\boldsymbol{A}$ and $\boldsymbol{B}$ defined by their origin points $O_{\boldsymbol{A}}, O_{\boldsymbol{B}} \in \mathbb{R}^3$ and their unit vector triads of axes $(\hat{x}_{\boldsymbol{A}}, \hat{y}_{\boldsymbol{A}}, \hat{z}_{\boldsymbol{A}}), (\hat{x}_{\boldsymbol{B}}, \hat{y}_{\boldsymbol{B}}, \hat{z}_{\boldsymbol{B}}) \in \mathbb{S}^2$, respectively. Then, the relative pose between these frames can be represented in terms of a rigid-body homogeneous transformation. The pose of frame $\boldsymbol{B}$ relative to frame $\boldsymbol{A}$, denoted by $T_{\boldsymbol{AB}}$ is a member of the SE(3) group and is defined such that

$$T_{\boldsymbol{AB}} = \begin{bmatrix} R_{\boldsymbol{AB}} & t^{\boldsymbol{A}}_{O_{\boldsymbol{B}} O_{\boldsymbol{A}}} \\ 0_{1\times 3} & 1 \end{bmatrix}, \tag{1}$$

where $R_{\boldsymbol{AB}}$ represents the respective rotation matrix belonging to the special orthogonal group SO(3) and $t^{\boldsymbol{A}}_{O_{\boldsymbol{B}} O_{\boldsymbol{A}}}$ represents the respective translation vector belonging to $\mathbb{R}^3$ expressed in frame $\boldsymbol{A}$.

For the relative navigation problem of two non-cooperative spacecraft in space, we will use the following notation:

- $\boldsymbol{G}$: the target spacecraft's frame (chief), having an arbitrary origin on the body of the target spacecraft and an arbitrary, but body-fixed, set of orthonormal unit vectors. For the non-cooperative problem, both the origin and the unit vectors of $\boldsymbol{G}$ are unknown to the designer.
- $\boldsymbol{S}$: the chaser spacecraft's frame (deputy), having an arbitrary origin on the body of the chaser spacecraft and an arbitrary, but body-fixed, set of orthonormal unit vectors. The designer is free to place frame $\boldsymbol{S}$ at their convenience.
- $\boldsymbol{C}$: the camera frame that is mounted on the chaser spacecraft. The convention for the $\hat{x}_{\boldsymbol{C}}, \hat{y}_{\boldsymbol{C}}, \hat{z}_{\boldsymbol{C}}$ unit vectors comprising the frame's orientation is: $\hat{z}_{\boldsymbol{C}}$ point out of the sensor's optical center,

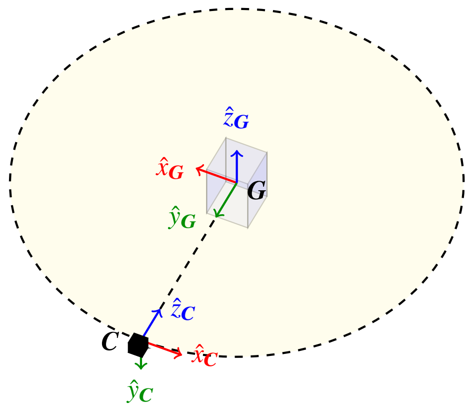


Fig. 1 Frames C and G in a circular orbit.

perpendicular to the image plane, $\hat{y}_C$ points downwards from the image plane, and $\hat{x}_C$ points to the right of the image plane. The origin of the frame is the optical center of the mounted camera. Since the camera is considered to be located at a known and fixed position and orientation in the chaser's body, the transformation between the camera frame $\boldsymbol{C}$ and the chaser frame $\boldsymbol{S}$ is constant and known.

We wish to design a visual SLAM (VSLAM) algorithm that receives images of the target and estimates the rigid-body transformation T_{GS} between the target-fixed frame $\boldsymbol{G}$ and the chaser-fixed frame $\boldsymbol{S}$, expressed in the chaser frame. Since the transformation between the $\boldsymbol{C}$ and $\boldsymbol{S}$ frames is known and constant, we can retrieve the relative pose by estimating T_{GC}. The Frames $\boldsymbol{G}$ and $\boldsymbol{C}$ are depicted in Figure 1

IV. Proposed Approach

SLAM consists of two simultaneously solved tasks: relative pose estimation and mapping of the environment. The map of the environment is the set of the observed 3D landmarks and is updated every time a new frame is entered in the optimization graph. As a simplifying assumption and in order to resolve the scale ambiguity, we assume that we have access to the ground truth poses corresponding to the two images selected for map initialization. These two image frames must have sufficient parallax between them. Along with the camera poses, the landmarks are the unknown variables in the VSLAM formulation. The images captured by the camera sensor are the input to the algorithm. As a result, the measurements are the detected features in the images and the measurement function is the projection function of the 3D landmarks to image points. Following the procedure of [24], Factor Graphs [20] are used to model VSLAM as a probabilistic inference problem. The keyframe-based formulation triangulates the features into the map of only a subset of the total input frames, which are called keyframes. A good keyframe selection is crucial since it aims to include sufficient and meaningful information [25] for landmark triangulation and position estimation with subsequent measurements, as well as to keep the graph sparse to ensure the tractability to solve the problem on-the-fly.

The main components of the algorithm structure are presented in the diagram of Figure 2 and will be further detailed in the following subsections. These include the Frontend module, the Keyframe Selection module, and the BackEnd module. The FrontEnd module processes each image to perform feature detection, matching, and tracking, and provides a set of measurements to be handled by the Keyframe Selection module. Then, the Keyframe Selection module provides a subset of these measurements for insertion into the graph, performed by the BackEnd module. Finally, the BackEnd module executes incremental graph optimization to provide the solution consisting of a 3D map of landmarks and the camera's relative pose trajectory.

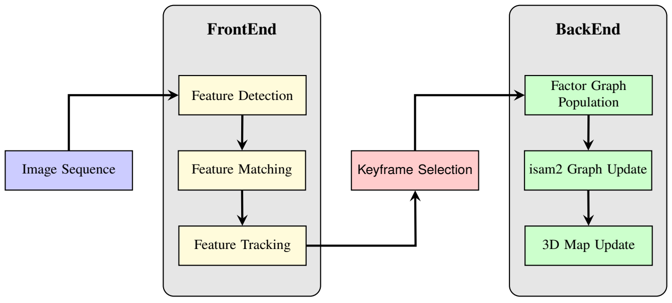


Fig. 2 Main components of the algorithm

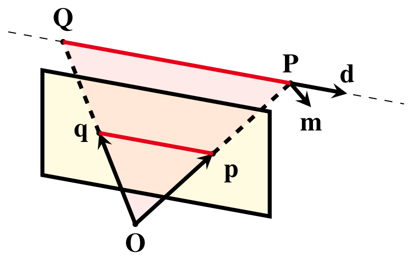


Fig. 3 Representation of 2D line segment, 3D line segment and Plücker coordinates of a 3D line

A. FrontEnd: Feature Extraction and Matching

In order to increase the robustness of the FrontEnd module, we combine point features with line segment features. Indeed, there are scenarios that might be encountered in orbit in which most point features may be lost, while line features can still be detected and matched. This argument is predicated on the fact that most points detected on a prismoid object's surface typically lie on small number of planes, while many of its lines lie at the intersections of these planes. Hence, when the satellite rotates, a sizeable portion of points may either vanish or be erroneously matched, while lines may still be accurately matched and thus, provide useful information for tracking. We designed the following features to be considered in our solution:

Point Features: The binary descriptor ORB is rotation-invariant and resistant to noise [26]. ORB is an efficient and quick-to-compute descriptor allowing us to extract a high number of feature points and filter out the low-quality matched features. The point features are matched through a nearest-neighbor search based on their descriptors. In order to improve the reliability of the matches, we employ a culling technique, where we apply a geometric constraint through the calculation of the two-view essential matrix and select only the matches that respect the epipolar constraint within some threshold via Random Sample Consensus (RANSAC).

Line segment representation: A line segment in the 2D image l_{pq} can be represented by its two 2D endpoints: p and $q \in \mathbb{R}^2$. Similarly, a 3D line segment L_{PQ} can be represented by its two 3D endpoints P and $Q \in \mathbb{R}^3$. We use and maintain the 3D endpoints representation of a line segment only for visualization and filtering purposes. The reason for this is that the constraint being enforced while tracking line segments is mainly imposed by the line and not its endpoints. By using a 3D line representation in contrast to a 3D line segment representation, we exploit this constraint and minimize the effect of endpoint noise in the projections and backprojections between 2D and 3D space. A 3D line L associated with the 3D line segment L_{PQ} can be represented by its Plücker coordinates as: $L = \left[m, d\right]^\top$, where $d = P - Q \in \mathbb{R}^3$ is the direction vector, and $m = P \times d \in \mathbb{R}^3$ is the moment vector. This representation involves two constraints: the direction and moment vectors are perpendicular to each other ($m \cdot d = 0$) and the ratio of the norms of these vectors ($\|m\|/\|d\|$) is constant. A graphical presentation of the different representations is shown in Figure 3.

We resort to the orthonormal representation of a 3D line for optimization purposes. As introduced in [27], any 3D line in 3-space has 4 degrees of freedom and may thus be represented by $(U, W) \in \mathrm{SO}(3) \times \mathrm{SO}(2)$, where $\mathrm{SO}(2)$ and $\mathrm{SO}(3)$ are the Lie Groups of (2×2) and (3×3) rotation matrices, respectively. * Given that $W \in \mathrm{SO}(2)$ may be represented by a single parameter $\theta_W \in \mathbb{R}$ and $U \in \mathrm{SO}(3)$ may be represented using 3 parameters $\theta_U \in \mathbb{R}^3$, we can use the vector $\theta = \left[\theta_U^\top,\ \theta_W\right]^\top \in \mathbb{R}^4$ for the representation of the 3D line, assuming that the θ_U representation of $SO(3)$ is not locally singular.

Line segment features: Line segment detection is achieved using the Line Segment Detector [14]. The algorithm uses only the detected line segments, whose length is greater than 30 pixels. The matching process follows the procedure of [13] and employs the advantages of Line Band Descriptors (LBD) [15]. Given that lines enforce a stronger geometric constraint than points, it is necessary to reject poor-quality line segment matches. If the distance ratio between the best match and the second-best match of the descriptor is greater than 0.85, the match is considered ambiguous and as a result, it is rejected.

B. FrontEnd: Feature tracking

Most hand-crafted feature descriptor extraction methods are predicated on the analysis of the image gradient in some neighborhood of an interest point (keypoint). Meanwhile, in rendezvous and proximity operations, factors such as lack of texture, high-contrast, sharp shadowing due to collimated lighting, and specular reflections may heavily impact the local image gradient. Consequently, matching features between frames and distinguishing good from bad matches is more challenging in space due to the effects described above. Ideally, the pair of images used for feature descriptor computation are similar in appearance — that is, in terms of viewing angle and illumination conditions — so as to obtain similar feature descriptors at given interest points in the image.

Typically, as is done in ORB-SLAM and its derived algorithms, matching of features is performed between the current frame's features (points or lines) to those detected in the last keyframe. According to this paradigm, the population of matched features may deteriorate if the viewing angle between the current frame and the last keyframe is significant (e.g., above 2°).

As mentioned in Section II, our feature tracking algorithm is a variation on the typical procedure used in the literature and is detailed in Algorithm 1. Specifically, instead of matching directly each frame's feature to the ones from the last keyframe, we match each frame's features to those in its immediately preceding frame. Due to the kinematics (dynamics) of the satellite circumnavigation problem, successive frames captured at a constant rate will have little parallax between them, so long as this rate is high enough. Subsequently, this procedure will produce a large population of reliable matches since the appearance of features in successive frames is similar. We encode these index matches in frame-to-frame mappings, and recursively query the resulting mappings of feature indices to ascertain the subset of features that have survived since the last keyframe. We expect this procedure to produce a higher number of features matched between keyframes, at the expense of maintaining and querying said mappings. Specifically, in order to keep

*Intuitively, defining a 3D line as $\{Q \in \mathbb{R}^3 | Q = P + sv,\ s \in \mathrm{R}\}$ implies 7 initial parameters with 3 constraint equations, yielding 4 degrees of

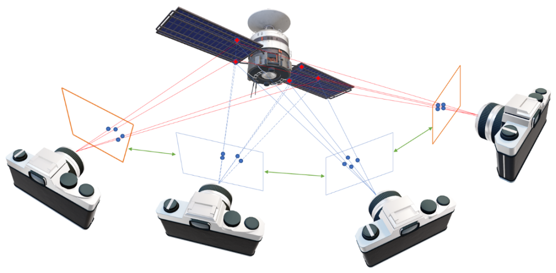


Fig. 4 FrontEnd Pairwise Matching Tracking Scheme and Keyframe Identification

track of the associations between features and landmarks, we maintain the following data structures. The *FrameMap* associates every 2D feature match with its corresponding match in the previous frame. The *TrackMap* accumulates all pairwise successive matches of an image feature into an ordered list. These correspond to all of the sightings of the said feature point. There is a TrackMap object for each tracked feature. The *KeyframeMap* is responsible for maintaining the associations between 2D features across keyframes in preparation for triangulation. We query the TrackMap to find all index mappings between the current frame and the previous keyframes. The *LandmarkMap* associates every successfully triangulated landmark to a TrackMap object.

Although a surface feature may be consistently matched across several frames, if at some point it is no longer matched with a candidate feature in the previous frame, it would no longer be tracked. To mitigate this effect, we adopt a sliding window approach, whereby features from the current frame are matched not only with those of the immediately preceding frame but also with features from n preceding frames, where for our application n is typically smaller than five frames. This approach helps revive some of the lost matches, positively influencing the matching procedure, while at the same time keeping the sliding window size small relative to the number of frames between keyframes to limit the impact on the growing computational complexity.

Algorithm 1: Feature Tracking

```
Data: current frame index $k$, current frame $F_k$, previous keyframes $KF_i, i = 1 : m$, 3D Map $\Phi_k$, sliding
      window size $n$
Detect features in $F_k$;
Match features with $F_{k-i}, i = 1, ..., n$;
Update the FrameMap;
Update all TrackMaps;
Query TrackMaps to generate candidate KeyframeMap;
if $F_k$ is a new keyframe then
    $KF_{m+1} \leftarrow F_k$;
    for each feature in KeyframeMap do
        Query LandmarkMap for the associated TrackMap;
        if corresponds to initialized landmark then
            if landmark passes reprojection test then
                Mark sighting for insertion into the graph.
            end
        else if corresponds to uninitialized landmark & number of sightings $\geq 3$ then
            Triangulate new landmark with KeyframeMap and $\{KF_i\}_{i \in \mathcal{J} \subset \{1,...,m+1\}}$ ;
            Mark sightings in KeyframeMap for insertion into the graph.
        else
            Update KeyframeMap with new sighting.
        end
    end
end
```

freedom.

C. Keyframe Selection

After the FrontEnd module has processed the input frames, the keyframe selection module needs to extract the keyframes that are being used by the optimizer. The newly inserted keyframe into the factor graph should be sufficiently distinct from the previous one to ensure ample visual coverage, while also maintaining enough similarity to continue tracking previously observed landmarks. The process of selecting keyframes involves striking a balance between exploration and exploitation —- exploring new parts of the target satellite while exploiting the current map to enhance the pose estimate. As detailed in a recent survey [28], the vast majority of keyframe-based VSLAM approaches employ heuristics to decide whether a frame is a good keyframe candidate or not. These heuristics typically fall into one of the following three categories: 1) *Time-based* approaches rely on the assumption of a strong correlation between elapsed time and changes in the scene's appearance. However, this assumption may not hold true for rendezvous and proximity operations. During inspection, the chaser spacecraft may perform a quasi-hovering maneuver, wherein minimal relative motion between the chaser and the target spacecraft may induce little appearance change over an extended observation horizon. As a result, we do not incorporate this type of heuristic in our solution; 2) *Distance-based* approaches rely on the assumption of a strong correlation between spatial change and appearance change. Spatial change is commonly determined through the estimation of camera's relative motion. This heuristic is amenable to our application, however, care must be exercised to address the special case of the pure rotation case. During pure rotation — that is, when the camera is static in position, yet rotated in attitude with respect to the target — appearance change does not correlate with spatial change; 3) *Appearance-based* approaches estimate directly the appearance change between two images. Sparse approaches use data association information, while dense ones quantify the photometric change between the two images. In [29], it is shown experimentally that the best representation of similarity between two images is retrieved from feature matching and the global histogram of each image.

Proposed Scheme: Many VSLAM applications exploit a single heuristic with a constant threshold for selecting new keyframes. In contrast, we propose a keyframe selection method based on multiple criteria: 1) *Estimated Parallax (Distance-based)*: This criterion ensures that the viewpoints are sufficiently separated due to relative motion, in order to achieve successful triangulation and sparse measurement insertion into graph. Calculated using the estimated poses of two frames, the parallax angle is obtained through the dot product of position vectors between the origins of frames $\boldsymbol{G}$ and $\boldsymbol{S}$, as can be seen in Figure 5. Assuming that the camera remains center-pointed toward the target spacecraft, then the parallax angle may be estimated by computing the dot product of the line-of-sight unit vectors at the two instances. We use appearance-based heuristics to detect whether the center-pointing assumption has been violated, in which case we discard this heuristic (e.g., in the pure translation case); 2) *Matched Features (Appearance-based)*: This metric estimates the dissimilarity of the current frame to the last keyframe based on the percentage of matched features tracked, when compared to the total number of matched features. If this percentage falls below a predetermined threshold, it signals that the current image is dissimilar enough to warrant being marked as a keyframe. To ensure continued tracking of the observed satellite, we dynamically adjust the matching sliding window size according to the number of tracked matches to recover more matches (i.e., the lower the match count, the wider the sliding window); 3) *Change in Optical Flow (Appearance-based)*: Optical flow serves as a reliable metric for scene change, as argued in [30]. For feature-based applications, the Lucas-Kanade filter is commonly used to calculate optical flow. In our work, we first compute the optical flow of each matched feature. Subsequently, using the computed optical flows, we determine the image sum optical flow and the centroid of the optical flows.

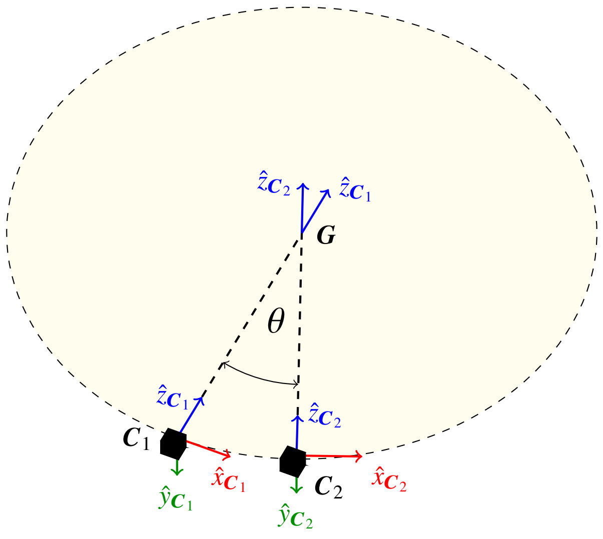


Fig. 5 Geometry of Parallax Angle (θ) from two separate camera line-of-sight axes $\hat{z}_{C_1}$ and $\hat{z}_{C_2}$

In our application, we assume that the observed surface features pertain to a single object and that the camera position remains always outside the convex hull of the observed landmarks. This assumption is crucial to ensure that the observed points are not points at infinity and, consequently, that a finite motion of the camera may result in sufficient parallax. If the center of mass (centroid) of the matched features significantly moves away from the image center, then we may detect that the camera is not target-center-pointing. In such a case, the estimated parallax is likely inaccurate. We still exploit this metric to gain some insightful information, by distinguishing between the two following extreme scenarios. The first case is the pure rotation situation, in which we want to avoid selecting a keyframe. For the pure rotation of the camera, the distribution of the optical flow among tracked features will exhibit low variance, as most points will experience nearly identical optical flow due to apparent image translation. The second scenario is the pure translation of the camera, in which case keyframe insertion is desirable. The distribution of the optical flow here will display a larger variance among the tracked features, as those closest to the camera optical center will exhibit higher optical flow. This metric allows us to discern between the pure rotation and pure translation cases, compensating for the unreliability of the parallax estimation.

Algorithm 2: Keyframe Selection

```
Data: optical flow $of$, center of gravity of point
      features $cf$, estimated parallax $\theta$, percentage of
      matched features $pf$, number of matched features
      $nf$, sliding window size $n$
if $nf < nf_{th}$ then
    Increase $n$;
end
if $of \geq of_{th}$ then
    if $cf > cf_{th}$ then
        Set $\theta_{th} \leftarrow 0$;
        Detect if pure translation exists;
        if Trnaslation is detected then
            Reset all thresholds ;
            Insert a new KeyFrame;
        end
    end
    if $\theta f > \theta f_{th}$ then
        Reset all thresholds ;
        Insert a new KeyFrame;
    else
        if $pf < pf_{th}$ then
            Decrease $\theta f_{th}$;
        end
    end
end
```

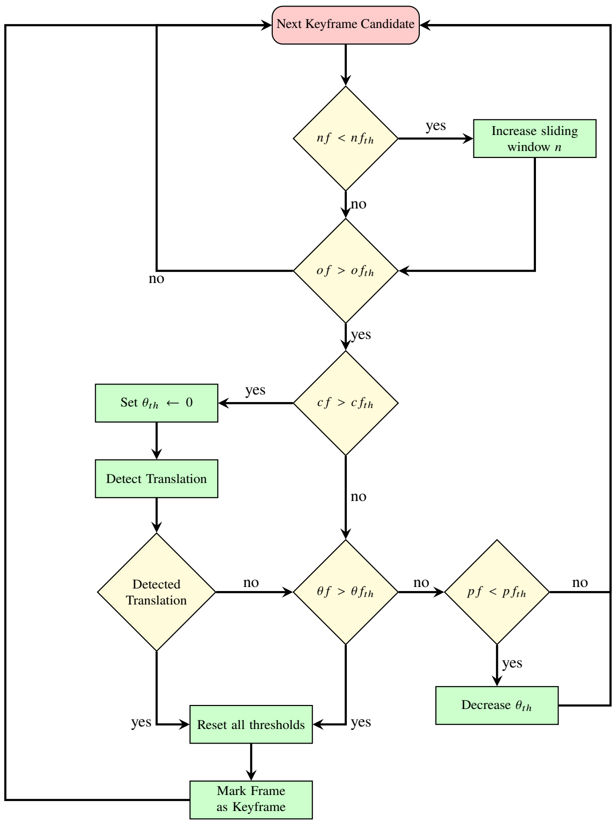


Fig. 6 Algorithmic representation (left) and flowchart diagram (right) of the keyframe selection scheme

The keyframe selection procedure is briefly laid out in Algorithm 2. We maintain five distinct heuristics along with their associated thresholds: 1) of and of_{th}: sum of the optical flows of all features matched between the current frame and the last keyframe. The optical flow threshold remains constant; 2) cf and cf_{th}: centroid position of feature points in the current frame. The threshold is the maximum radial distance of the centroid from the center of the image in order to satisfy the center pointing assumption; 3) θ and θ_{th}: estimated parallax between the current frame and the last keyframe. If the estimated parallax is deemed reliable, then we compare it against a minimum threshold. The parallax threshold is slightly relaxed when the appearance-based metrics indicate enough dissimilarity, but the estimated parallax still indicates insufficient viewpoint change; 4) pf and pf_{th}: percentage of matched features being tracked since the last keyframe with respect to the number of total matched features. If the percentage threshold is not satisfied, while the parallax and the optical flow thresholds are met, then we slightly relax the percentage threshold; 5) nf and nf_{th}: number of matched features being tracked since the last keyframe. When the number of matched features falls below its associated threshold, the matching sliding window size is increased in order to recover feature matches.

D. BackEnd: Factor Graph Optimization

The incremental smoothing optimization problem, cast as a factor graph, is solved using the iSAM2 algorithm [31] within the GTSAM library framework [20]. iSAM2 efficiently solves the incrementally growing navigation problem by directly encoding the factor graph as a Bayes tree, and updating the tree on-the-fly as new factors are inserted into the graph. The line segment measurements are modeled as nonlinear factors in the factor graph. The line factor encodes the error, at a given camera pose, between the the 2D reprojection of an initial 3D line estimate and a noisy 2D line segment measurement. The line factor requires the Jacobian of the error with respect to the relative pose, $\boldsymbol{T_{GS}} \in \mathrm{SE}(3)$ and with respect to the orthonormal (minimal) representation of the 3D line, denoted by $\theta \in \mathbb{R}^4$.

Initial 3D line Estimate: The process of triangulating a 3D line given the observation of line segments from two separate views is well-known [32]. Line triangulation uses two 2D line segments $L_{p_i q_i}, i = 1, 2$, where $p_i, q_i, i = 1, 2$ are the 2D endpoints of each line segment, in order to calculate the reconstructed 3D line in its Plücker coordinates representation m_r, d_r. After reconstructing the 3D line we can calculate the 3D endpoints of the line using the Endpoints Trimming algorithm [33].

When the observed 3D line lies close to the epipolar plane of the two views, the 3D line reconstruction problem from two views may become degenerate (numerically ill-conditioned). Hence, in our work, we reconstruct a line from three separate views using three 2D line segments $l_{p_i q_i}, i = 1, 2, 3$, that are matched with each other in three consecutive keyframes, and the camera projection matrix $P \in \mathbb{R}^{3x4}$:

$$P = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2}$$

where $f_x, f_y \in \mathbb{R}$ are the camera's focal length and c_x and $c_y \in \mathbb{R}^2$ are the camera's optical center coordinates. We use the homogeneous coordinates of the 2D endpoints $\bar{p}_i = [p_i, 1]^\top,\ \bar{q}_i = [q_i, 1]^\top, i = 1, 2, 3 \in \mathbb{R}^3$ and define for each line segment the plane $\pi_i = P^\top(\bar{p}_i \times \bar{q}_i) \in \mathbb{R}^{4x1}, i = 1, 2, 3$, that contains the two endpoints and the camera's optical center, as seen in Figure 7. The 3D line is the intersection of the two most dominant planes among the three planes. We recover the two dominant planes using singular value decomposition of the matrix $W = UDV^\top \in \mathbb{R}^{3x4}$, with W given such that its rows are the calculated plane vectors $\pi_i^\top, i = 1, 2, 3$. The first two columns of V correspond to the two largest eigenvalues of W, denoted as $v_1, v_2 \in \mathbb{R}^{4x1}$, and constitute the best approximation of W with rank 2. This can be used to extract the line's Plücker coordinates m_r, d_r as follows

$$\begin{bmatrix} [d_r]_\times & m_r \\ -m_r^\top & 0 \end{bmatrix} = v_1 v_2^\top - v_2 v_1^\top, \tag{3}$$

where $[\cdot]_\times$ indicates the skew-symmetric matrix obtained from a vector. We filter the triangulated 3D line segments based on the Klein quadric constraint [23], which dictates that a 3D reconstructed line must satisfy $m_r^\top d_r < 0.1$.

Line reprojection error: The error function used to guide the optimization process for the line landmarks is the line reprojection error. After reprojecting the 3D line in the image plane, the error is defined as the distance between each measured 2D line segment endpoint $p, q \in \mathbb{R}^2$ and the reprojected 2D line $l_r = [l_1, l_2, l_3]^\top \in \mathbb{R}^3$, as can be seen in Figure 8. Given the homogeneous coordinates $\bar{p} = [p, 1]^\top,\ \bar{q} = [q, 1]^\top \in \mathbb{R}^3$ of the measured line segment endpoints and the expected reprojected line l_r, the error e is defined as

$$e(\bar{p}, \bar{q}, l_r) = \left[\frac{\bar{p} \cdot l_r}{\sqrt{l_1^2 + l_2^2}}, \frac{\bar{q} \cdot l_r}{\sqrt{l_1^2 + l_2^2}} \right]^\top. \tag{4}$$

The line factors are finally inserted into the factor graph if they satisfy the $\mathcal{X}^2$ distribution test on the reprojection error [23], which uses the reprojection process to identify outliers.

Line Jacobians: The Jacobians of the error are calculated using the chain rule, for which the analytical derivations may be found in [33]. Assume a 3D line has an orthonormal representation of θ_l, then $\boldsymbol{L^G},\ \boldsymbol{L^S} \in \mathbb{R}^6$ are the Plücker coordinates representation of the line in the target and spacecraft frame respectively and $l_r \in \mathbb{R}^3$ is the reprojected line coordinates in the image. Then, the Jacobians of the error e with respect to θ_l and the parameters of the pose, $\xi_{GS} \in \mathbb{R}^6$, are given as

$$\frac{\partial e}{\partial \theta_l} = \frac{\partial e}{\partial l_r} \frac{\partial l_r}{\partial L^S} \frac{\partial L^S}{\partial L^G} \frac{\partial L^G}{\partial \theta_l}, \quad \frac{\partial e}{\partial \xi_{GS}} = \frac{\partial e}{\partial l_r} \frac{\partial l_r}{\partial L^S} \frac{\partial L^S}{\partial \xi_{GS}} \tag{5}$$

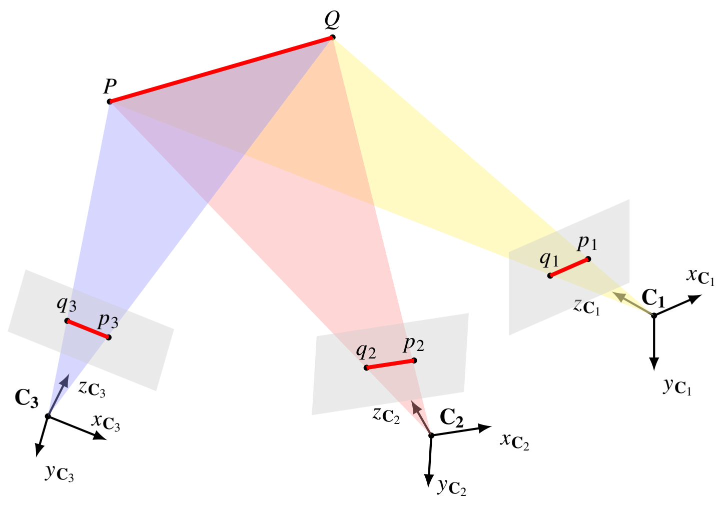


Fig. 7 Three view triangulation of a line

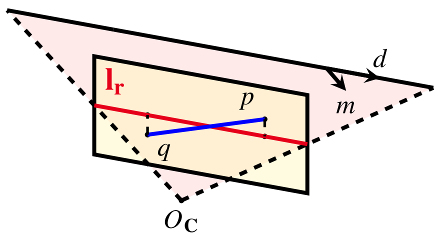
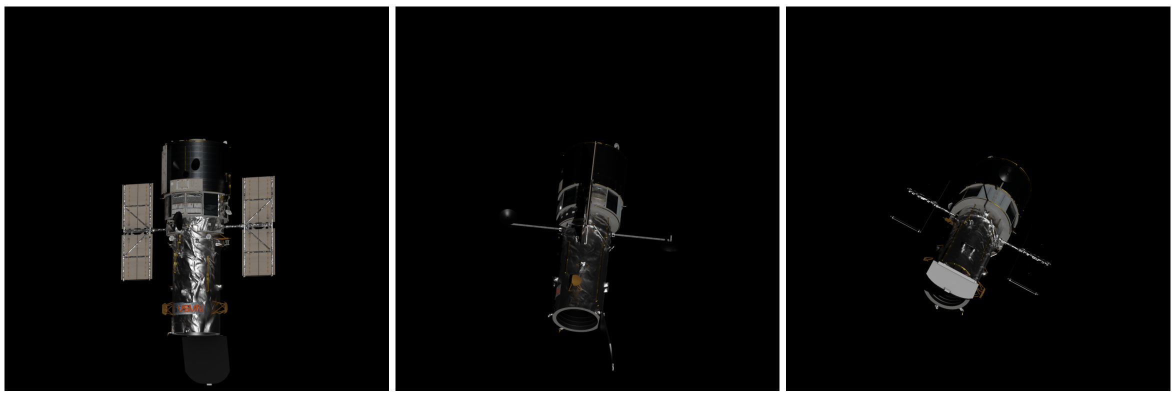


Fig. 8 Visualization of line reprojection error. l_r: reprojected line, p, q: measured endpoints.

V. Image Sequence Datasets

In order to test the proposed VSLAM solution, we use simulated sequences of images of the Hubble Space Telescope. The simulated sequences are generated using the Blender software [34], a rendering tool known for its ability to create high-quality and photorealistic synthetic images. The camera parameters used to capture the simulated images are f_x, f_y: 3824.46 and c_x, c_y: 500 pixels. The trajectories simulate the Proximity Operations B phase of a space rendezvous scenario with the Hubble Space Telescope. The relative distance between frames $\boldsymbol{G}$ and $\boldsymbol{S}$ is approximately 100 meters and the motion of the $\boldsymbol{S}$ frame is propagated using the Clohessy-Wiltshire (CW) equations [35]. Simulation scenarios are obtained by varying the angular velocity of the $\boldsymbol{G}$ frame with respect to the Hill frame $\boldsymbol{H}$ and the initial conditions of the chaser spacecraft's Hill-relative motion: position, velocity, and orientation. Specific Sun vector directions may be assigned, with collimated illumination conditions. Each dataset consists of 1500 images captured every 0.1 seconds, which results in trajectories of approximately 2.5 minutes.

We use two poses with sufficient parallax as the first two keyframes and triangulate landmarks into the initial map with these ground truth poses. The first dataset (DAT1) begins with nominal lighting conditions and relative pose. Initially, all of the visible surface area of the target is lit and the panels of the target spacecraft do not occlude the body of the spacecraft or cast shadows on it. The second dataset (DAT2) begins with more challenging conditions — one of the solar panels partially occludes and intense self-shadowing affects parts of the spacecraft. We expect in both situations that occlusions and shadowing will negatively impact the number of successfully tracked landmarks.



(a) Frame 0 **(b) Frame 600** **(c) Frame 1200**

Fig. 9 Sample of frames from DAT1 demonstrating challenging conditions

We tested our algorithm using the real-life dataset captured from the NASA STS-125 Servicing Mission 4 (SM4) in

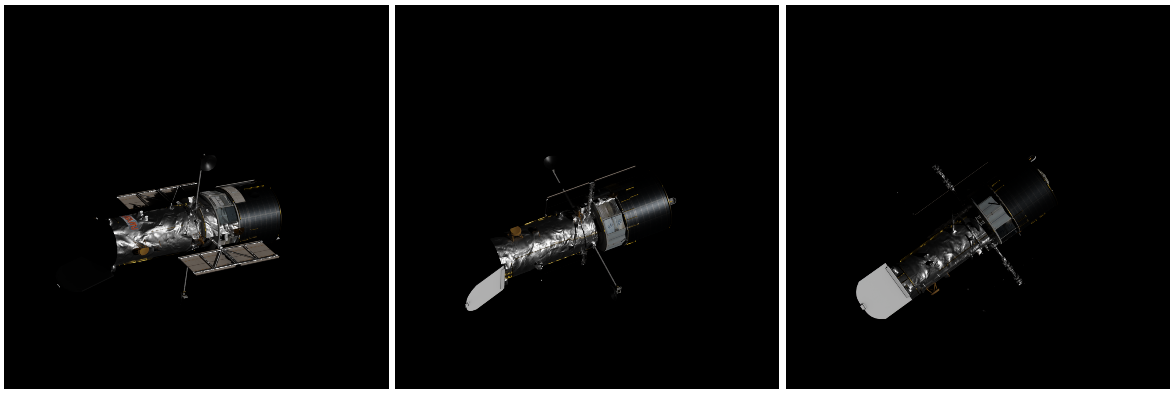

(a) Frame 0 (b) Frame 600 (c) Frame 1200

Fig. 10 Sample of frames from DAT2 demonstrating challenging conditions

2009. The images were captured during the Hubble Servicing Mission 4. Testing the FrontEnd module, as depicted in Figure 2 on this realistic dataset demonstrates the robustness and applicability of the line augmented algorithm to real-life navigation scenarios. Since ground truth poses do not exist for this dataset, we exploit the Hubble Spacecraft's 3D Shape model to perform model-based pose estimation and to thus establish ground truth values.

VI. Experimental Results

A. Feature tracking

We conduct a comparative analysis of the matched features obtained by our algorithm, which allows us to draw meaningful conclusions about its performance. Specifically, we evaluate the performance of the algorithm's FrontEnd module functionality, which includes feature detection, matching, and tracking. Our investigation focuses on examining the disparities between point features and line features in specific parts of the trajectory. By comparing these feature types, we gain insights into their respective strengths, weaknesses, and suitability for visual localization and mapping in spacecraft rendezvous and proximity operations.

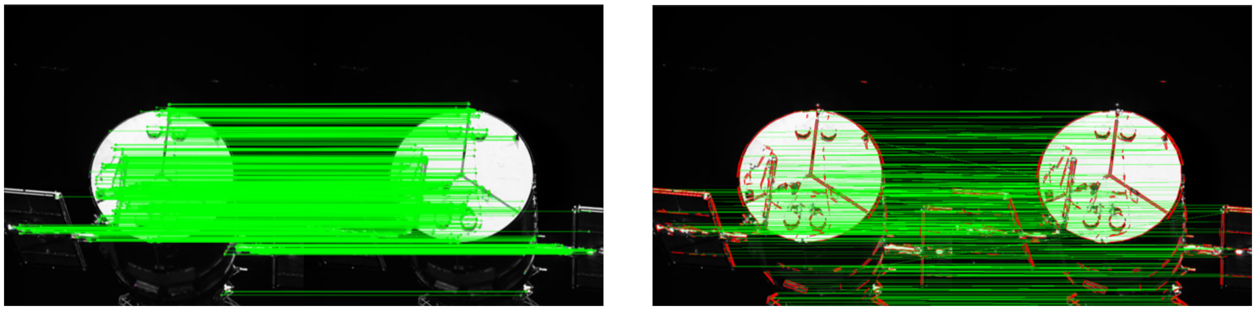


Fig. 11 Tracked point features (left) and line features (right) between two consecutive keyframes under intense shadowing conditions

Intense Shadowing: In this scenario, our focus is on examining the potential of line segments to offer more information compared to point features in areas of the image that experience significant shadowing. For this purpose, we have selected two consecutive keyframes, wherein the spacecraft's body and panels are fully shadowed. We then compare the performance of our algorithm with and without the utilization of line segments. In order to visualize and compare the tracked features, we present the tracked point and line features between the two keyframes in Figure 11.

In this particular segment of the trajectory, our goal is to identify features within the spacecraft's body and panels, whose parts are subjected to extensive shadowing. By examining Figures 11 and 12, we observe a notable distinction in the distribution of point features and line segments in these areas. Detected and tracked point features are predominantly concentrated in the brighter regions of the image, while they are limited in the heavily shadowed areas. In contrast, the line segments exhibit matches both across the panels and along the body of the spacecraft. This observation suggests that despite the challenging lighting conditions, the line matches are capable of recovering layout information, offering a broader distribution of features that is crucial for maintaining tracking of the target in challenging situations.

Informative Measurements: In this analysis, we investigate the efficacy of point and line features in constructing a high-quality map for the target spacecraft. We specifically focus on a segment of the trajectory where the entire satellite remains within the field of view. We compare tracked features only pertaining to the panels of the spacecraft, which possess a distinctive structure and offer rich geometric information. Figure 12 shows the tracked point features extracted from the panels of the satellite alongside the corresponding tracked line features within the same region.

Upon examining these figures, it is evident that the point matches are relatively scarce and are concentrated in high-contrast regions of the image. On the other hand, the line segments exhibit a significantly higher number of matches, more spread out across the surface of the panels. We expect that line segments should contribute substantially to the mapping process, yielding more impactful measurements for both mapping and localization.

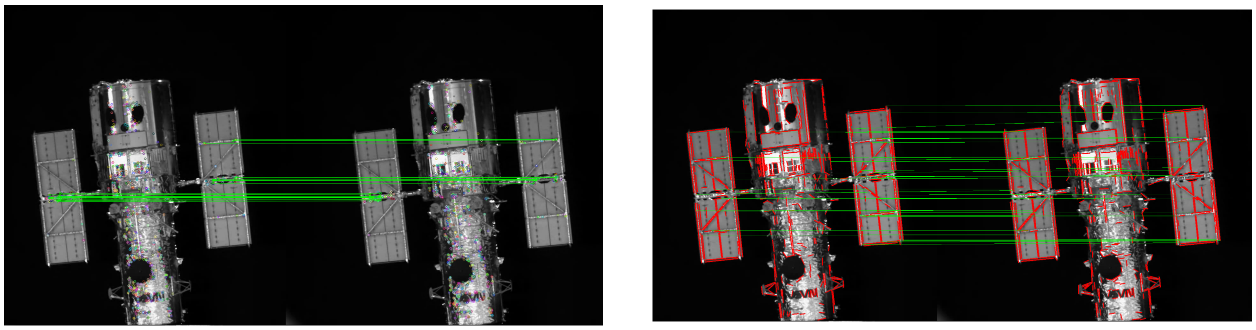


Fig. 12 Matched point features (left) and line features (right) between two consecutive frames on solar panels

B. Factor Graph Optimization

The main evaluation metric for our VSLAM solution is navigation error. We compare the performances of two different algorithms: one using only point features (P) and another using point and line features simultaneously (PL). We provide for each synthetic dataset, DAT1 and DAT2, the trajectory of the estimated position expressed in the $\boldsymbol{G}$ frame and the trajectory of the estimated attitude represented in quaternions in Figures 13, and 16. Furthermore, we provide the position error (m) expressed in the Local Vertical Local Horizontal (LVLH) frame, denoted as $\boldsymbol{L}$, given as $\delta t = R_{\boldsymbol{LG}} \left(\hat{t}^{\boldsymbol{G}}_{O_S O_G} - t^{\boldsymbol{G}}_{O_S O_G} \right)$, where the hat notation represents an estimate of the signal in Figure 14. The LVLH frame is constructed using the target-centric position vector and the target-relative velocity vector, providing meaningful information in the following axes: (AT) along track, (XT) cross track, and (RAD) radial directions. Moreover, the attitude error is defined as $\delta\kappa = \log \left(R^{\top}_{\boldsymbol{GS}} \hat{R}_{\boldsymbol{GS}} \right)$ and shown in Figure 17. This error vector represents the attitude error around the ground truth $\boldsymbol{S}$ frame axes. Finally, for each dataset, we provide a diagram with the population of the landmarks that constitute the map in Figures 15, and 18. For the PL solution, we show the number of known and tracked landmarks and the number of new landmarks inserted into the map at every keyframe. The navigation errors are given in Table 1, as Root Mean Squared Error (RMSE) values.

Table 1 RMSE of position (m) and attitude (deg) error DAT1

Position	P	PL	% ↓
δt_{AT}	4.3655	**0.9808**	77.53
δt_{XT}	**0.3376**	1.5350	78.01
δt_{RAD}	**1.1477**	1.2636	9.17
$\|\|\delta t\|\|_2$	4.5265	**2.2170**	51.02

Attitude	P	PL	% ↓
$\delta\kappa_1$	**0.5704**	0.9793	41.75
$\delta\kappa_2$	2.5173	**0.4300**	82.92
$\delta\kappa_3$	1.5335	**0.4068**	73.47
$\|\|\delta\kappa\|\|_2$	3.0023	**1.1443**	61.89

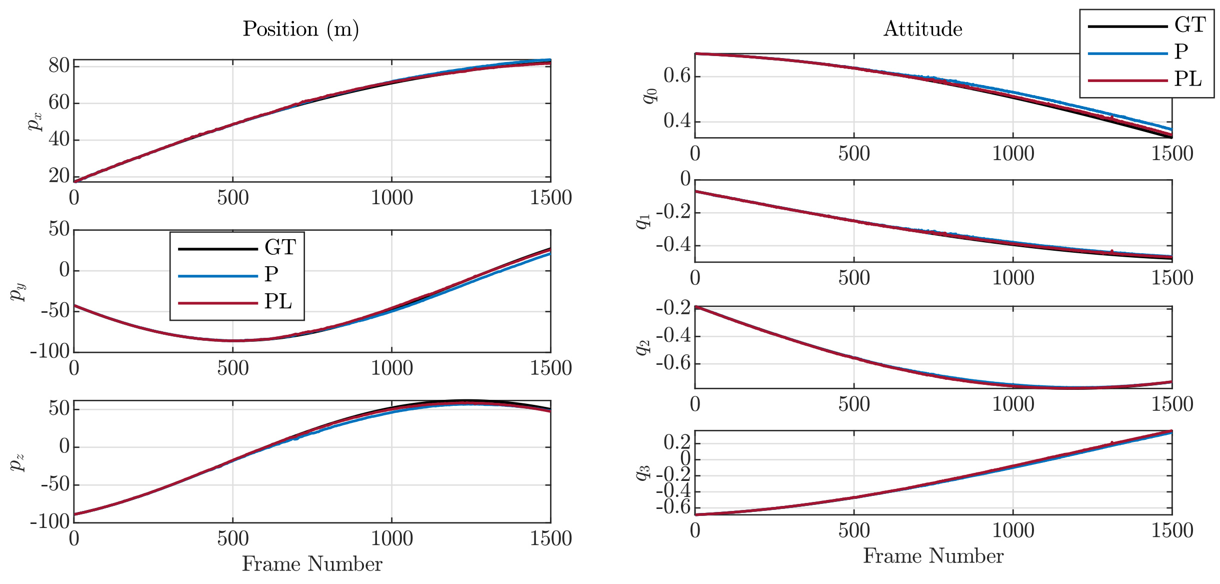


Fig. 13 Position in meters (left) and Attitude quaternions (right) in the *G* frame for DAT1

Table 2 RMSE of the position (m) and attitude (deg) error DAT2

Position	P	PL	% ↓
δt_{AT}	3.8484	**1.3686**	64.44
δt_{XT}	**1.0930**	1.1257	2.90
δt_{RAD}	1.2305	**1.1683**	5.06
$\Vert\delta t\Vert_2$	4.1856	**2.1226**	49.29

Attitude	P	PL	% ↓
$e\delta\kappa_1$	1.3155	**0.7293**	44.56
$\delta\kappa_2$	1.8171	**0.6420**	64.67
$\delta\kappa_3$	**1.2479**	1.4644	14.79
$\Vert\delta\kappa\Vert_2$	2.5670	**1.7575**	31.54

Position error: Table 1 indicates a significant improvement of 77.53% in the along-track direction, but no improvement in the cross-track and radial directions. We note that the radial direction coincides with the camera line-of-sight direction and therefore error in the radial direction corresponds to error in the depth estimate. The lack of improvement in the radial direction with respect to the P algorithm is explained by the fact that the line reprojection error function, which is used in the line factor presented in Section IV.D, suffers from the same inherent sensitivity to depth as the point reprojection function since these are both predicated on the projective geometry in the image plane. We note that the corrective nature of 2D line measurements is inherently perpendicular to the direction of the measured line. The preponderance of reliable measurement lines tracked along the cross-track direction explains the corrective power of the PL algorithm in the along-track direction for DAT1, as may be viewed in the upper left subplot of Figure 14. In contrast, the lack of strong rectilinear features on the surface of the target satellite in the along-track direction results in unreliable lines detected along that direction and, consequently, in poor cross-track error when compared to the P algorithm. This seems to be a possible explanation for the increase in the position error of the PL algorithm in the cross-track direction specifically at and after frame 600. In addition, the stark decrease in the number of both tracked point and line landmarks as of Frame 500 may explain the increasing overall error in both P and PL algorithms. This may be due to the challenging detection and matching conditions, such as self-occlusion and self-shadowing as depicted in Figure 9. In sum, the norm of the error is much smaller — 51% decrease with respect to the P algorithm. We observe the same trends in the DAT2 Figure 17 plots. The values in Table 2 also reflect the conclusion that the corrective nature of the PL algorithm in position error lies mainly in the along-track direction with little to no improvement in the cross-track and radial directions. We note that the maximum magnitude of the position error in the P algorithm is larger compared to that of the DAT1, i.e., up to 12 m deviation in DAT2 compared to 6 m deviation in DAT1. Notably, until frame 1000 the P and PL algorithms seem to perform similarly in the DAT2 scenario. Given the more challenging initial condition, the number of reliable line landmarks initialized in the early phases of DAT2 is significantly lower than that of DAT1. Naturally, the weight of corrections of the pose owing to the line landmarks is lessened and the PL algorithm

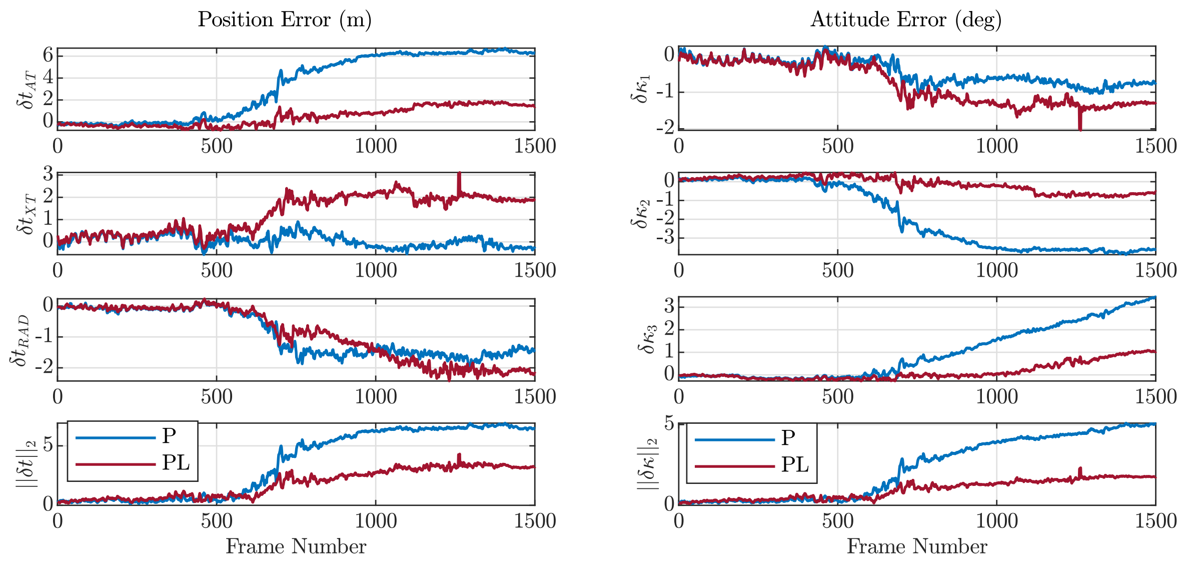


Fig. 14 Position error (left) and attitude error (right) in the LVLH frame for DAT1

demonstrates a similar performance to the P algorithm. Nevertheless, as of frame 1000 in the DAT2 sequence, the PL algorithm outperforms the P algorithm in the along-track direction owing to the drastic reduction of tracked point and line landmarks.

Attitude error: The attitude error around the camera line-of-sight direction is markedly improved (by 73%) with respect to the P algorithm. This is expected since the reprojection error function of the line factors is sensitive to the camera roll angle and therefore, we expect impactful correction around this direction. We note that the plots of the position error in the cross-track direction and the along-track direction resemble the plots of the attitude errors around the spacecraft x and y axes, respectively. This is also expected behavior since the deviation in attitude around the two axes perpendicular to the line of sight axis (the spacecraft x and y axes) will induce a position error in the image plane axes — that is, the cross-track and along-track directions, respectively. Therefore, it is natural that the signals are highly correlated, and, in turn, the corrective nature of the line features in attitude can be explained in the same way, as in the discussion regarding the position errors. Tables 1 and 2 clearly demonstrate that the PL algorithm outperforms the P algorithm in norm of attitude error in both datasets.

DAT1 vs. DAT2 Landmark tracking: In DAT1 a higher number of landmarks is successfully tracked within the first 500 frames. This is due to the fact that the initial map of landmarks constructed during initialization stays within view for a longer period. A significant tracked landmark drop is observed after frame 500 in Figure 15, which is associated with the increase of the navigation error in the system. Despite the fact that both point and line landmarks drop, the PL algorithm does not suffer from deteriorated performance. In DAT2 the same pattern is observed after frame 1000 in Figure 18 with a stark decrease in tracked line landmarks, with the worst-case scenario at frame 1400, as can be seen in Figure 10. This frame number also coincides with the worst navigation error, as seen in Figure 17.

VII. Conclusion

In this paper, we investigate the problem of purely visual-based navigation for non-cooperative spacecraft rendezvous and proximity operations in orbit. We build upon an existing visual SLAM (VSLAM) pipeline for the space environment and tailor it to tackle the challenges of reliable tracking in spacecraft rendezvous applications. We apply a pair-wise matching method combined with a holistic keyframe selection scheme that takes into account the problem's geometry and adapts the thresholds accordingly. In addition, we integrate line segments as a stronger geometric primitive than points, in order to increase the robustness of the FrontEnd module of the algorithm against the adverse environment conditions in orbit, specifically self-occlusions and self-shadowing, which are known to affect point-tracking algorithms. The FrontEnd module is tested against real-life imagery showcasing significantly better performance in representing the entirety of the observed satellite and overcoming intense shadowing conditions. The PL version of the algorithm,

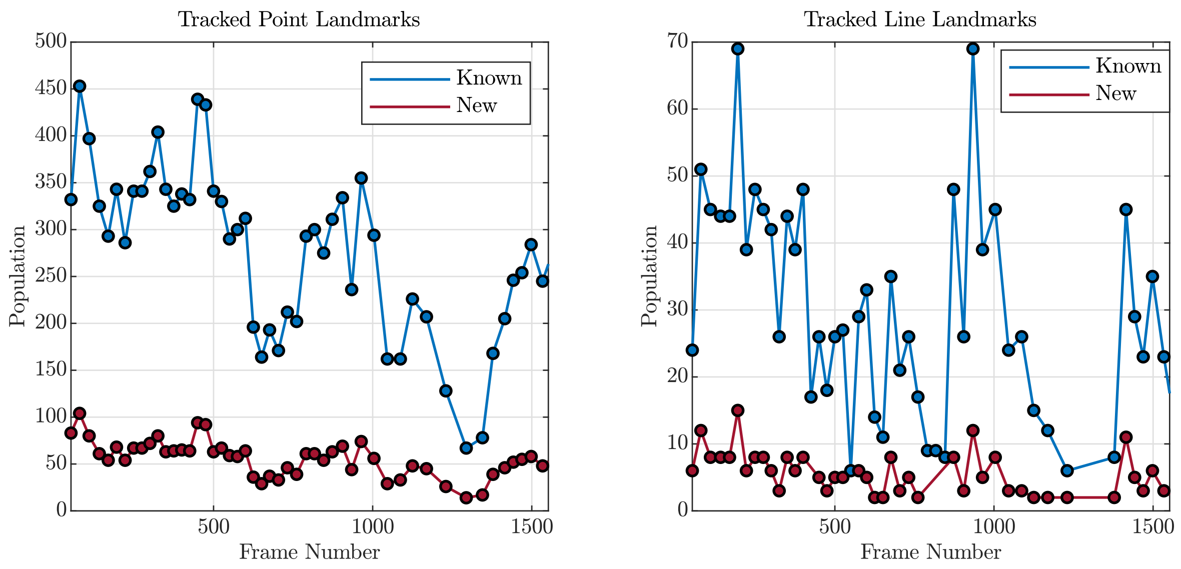


Fig. 15 Population of known tracked and newly triangulated landmarks, for points (left) and lines (right) separately, in every keyframe for DAT1.

which uses both point and line features, is tested against two synthetic datasets and exhibits markedly better position and attitude estimates compared to the P version, which uses only point features. This decrease in the estimation error is crucial to the mission's success and is necessary for the critical task of keeping track of the target satellite over long time horizons.

Acknowledgments

This work was supported by the National Science Foundation grant award 2101250 and the Air Force Office of Space Research Office's Space University Research Initiative grant award FA9550-23-1-0723. The authors would like to thank the following individuals: Juan-Diego Florez-Castillo for his contributions to glueing the modules together and Lorenzo Ticozzi for his contributions to generating the synthetic datasets.

References

[1] Liou, J. C., and Johnson, N. L., "Planetary science. Risks in space from orbiting debris," *Science*, Vol. 311, No. 5759, 2006, pp. 340–341.

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

[3] Rembala, R., and Ower, C., "Robotic assembly and maintenance of future space stations based on the ISS mission operations experience," *Acta Astronautica*, Vol. 65, No. 7-8, 2009, pp. 912–920.

[4] Mohan, S., "Reconfiguration methods for on-orbit servicing, assembly, and operations with application to space telescopes," Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[5] Center, G. S., "On-orbit satellite servicing study project report," Tech. rep., NASA Goddard Space Flight Center, 2010.

[6] Strasdat, H., Montiel, J. M. M., and Davison, A. J., "Real-time monocular SLAM: Why filter?" *IEEE International Conference on Robotics and Automation, Anchorage, AL, May 2010*, 2010, pp. 2657–2664.

[7] Sonnenburg, A., Tkocz, M., and Janschek, K., "EKF-SLAM based Approach for Spacecraft Rendezvous Navigation with Unknown Target Spacecraft," *IFAC Proceedings Volumes*, Vol. 43, No. 15, 2010, pp. 339–344.

[8] Tweddle, B. E., "Computer Vision Based Navigation for Spacecraft Proximity Operations," Ph.D. thesis, Massachusetts Institute of Technology, 2013.

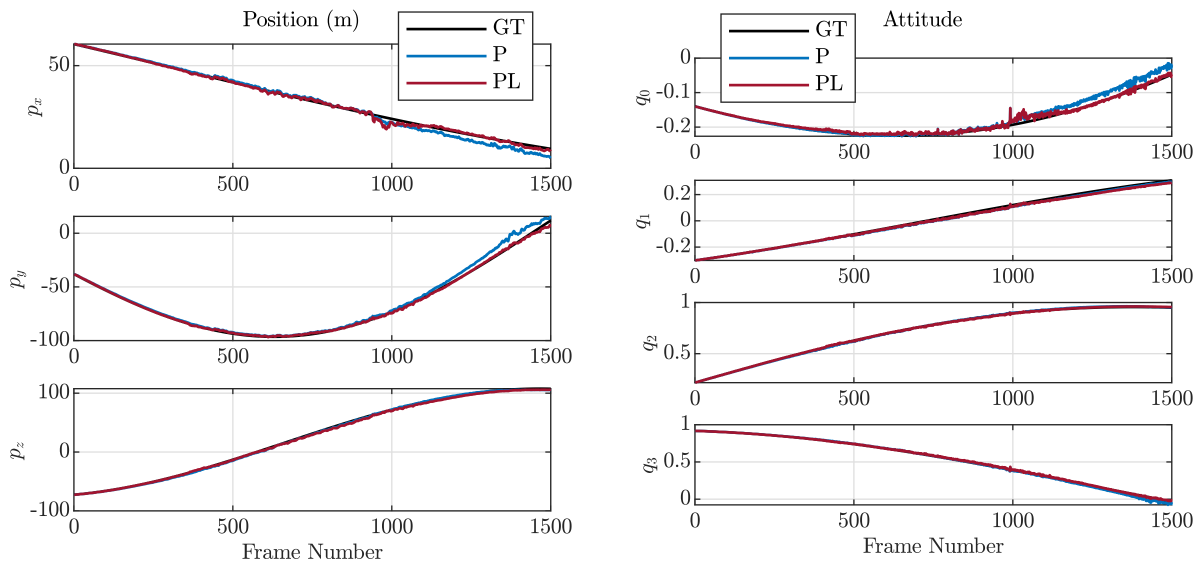


Fig. 16 Position in meters (left) and attitude quaternions (right) in the *G* frame for DAT2

[9] Mur-Artal, R., and Tardós, J. D., "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, Vol. 33, No. 5, 2017, pp. 1255–1262.

[10] Younes, G., Asmar, D., Shammas, E., and Zelek, J., "Keyframe-based monocular SLAM: design, survey, and future directions," *Robotics and Autonomous Systems*, Vol. 98, 2017, pp. 67–88.

[11] Engel, J., Schöps, T., and Cremers, D., "LSD-SLAM: Large-Scale Direct Monocular SLAM," *Computer Vision – ECCV 2014*, edited by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Springer International Publishing, Cham, 2014, pp. 834–849.

[12] Oumer, N. W., and Panin, G., "Camera-Based Tracking for Rendezvous and Proximity Operation of a Satellite," *Advances in Aerospace Guidance, Navigation and Control*, edited by J. Bordeneuve-Guibé, A. Drouin, and C. Roos, Springer International Publishing, Cham, 2015, pp. 625–638.

[13] Fu, Q., Wang, J., Yu, H., Ali, I., Guo, F., He, Y., and Zhang, H., "PL-VINS: Real-Time Monocular Visual-Inertial SLAM with Point and Line Features," , 2022.

[14] Grompone von Gioi, R., Jakubowicz, J., Morel, J.-M., and Randall, G., "LSD: a Line Segment Detector," *Image Processing On Line*, Vol. 2, 2012, pp. 35–55.

[15] Zhang, L., and Koch, R., "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, Vol. 24, No. 7, 2013, pp. 794–805.

[16] Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., and Moreno-Noguer, F., "PL-SLAM: Real-time monocular visual SLAM with points and lines," *IEEE International Conference on Robotics and Automation, Marina Bay Sands Singapore, May 2017*, 2017, pp. 4503–4508.

[17] Micusik, B., and Wildenauer, H., "Structure from Motion with Line Segments under Relaxed Endpoint Constraints," *2014 2nd International Conference on 3D Vision, Tokyo, Japan, December 2014*, Vol. 1, 2014, pp. 13–19.

[18] Ren, G., Cao, Z., Liu, X., Tan, M., and Yu, J., "PLJ-SLAM: Monocular Visual SLAM With Points, Lines, and Junctions of Coplanar Lines," *IEEE Sensors Journal*, Vol. 22, No. 15, 2022, pp. 15465–15476.

[19] Thomas, D., Kelly, S., and Black, J., "A monocular SLAM method for satellite proximity operations," *American Control Conference, Boston, MA, July 2014*, 2016, pp. 4035–4040.

[20] Dellaert, F., and Kaess, M., "Factor Graphs for Robot Perception," *Foundations and Trends in Robotics*, Vol. 6, No. 1-2, 2017, pp. 1–139.

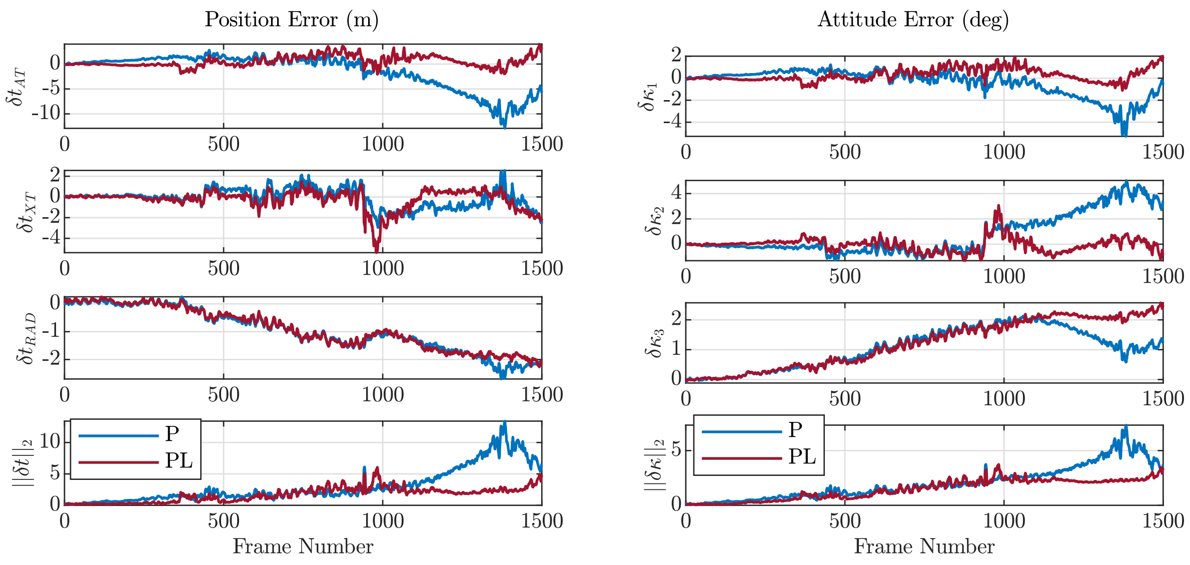


Fig. 17 Position error (left) and attitude error (right) in the LVLH frame for DAT2

[21] Dor, M., and Tsiotras, P., "ORB-SLAM Applied to Spacecraft Non-Cooperative Rendezvous," *Space Flight Mechanics Meeting, Kissimme, FL, January 2018*, AIAA, 2018.

[22] Sumikura, S., Shibuya, M., and Sakurada, K., "OpenVSLAM: A Versatile Visual SLAM Framework," *Proceedings of the 27th ACM International Conference on Multimedia*, Association for Computing Machinery, 2019, p. 2292–2295.

[23] Shu, F., Wang, J., Pagani, A., and Stricker, D., "Structure PLP-SLAM: Efficient Sparse Mapping and Localization using Point, Line and Plane for Monocular, RGB-D and Stereo Cameras," *IEEE International Conference on Robotics and Automation, London, England, May*, 2023.

[24] Dor, M., Driver, T., Getzandanner, K., and Tsiotras, P., "AstroSLAM: Autonomous Monocular Navigation in the Vicinity of a Celestial Small Body – Theory and Experiments," *arXiv preprint arXiv:2212.00350*, 2022.

[25] Ma, Y., Soatto, S., J., K., and Sastry, S., *An Invitation to 3-D Vision: From Images to Geometric Models*, 2nd ed., Interdisciplinary Applied Mathematics, Vol. 26, Springer, 2004.

[26] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., "ORB: An efficient alternative to SIFT or SURF," *International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 2564–2571.

[27] Bartoli, A., and Sturm, P., "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer Vision and Image Understanding*, Vol. 100, No. 3, 2005, pp. 416–441.

[28] Dias, N. J. B., Laureano, G. T., and Da Costa, R. M., "Keyframe Selection for Visual Localization and Mapping Tasks: A Systematic Literature Review," *Robotics*, Vol. 12, No. 3, 2023.

[29] Zhang, H., Li, B., and Yang, D., "Keyframe detection for appearance-based visual SLAM," *IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, October 2010*, 2010, pp. 2071–2076.

[30] Nourani-Vatani, N., and Pradalier, C., "Scene change detection for vision-based topological mapping and localization," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, October 2010*, 2010, pp. 3792–3797.

[31] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F., "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, Vol. 31, No. 2, 2012, pp. 216–235.

[32] Hartley, R., and Zisserman, A., *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge University Press, 2004.

[33] Lee, S. J., and Hwang, S. S., "Elaborate Monocular Point and Line SLAM With Robust Initialization," *IEEE/CVF International Conference on Computer Vision, Seoul, Korea, October 2019*, 2019, pp. 1121–1129.

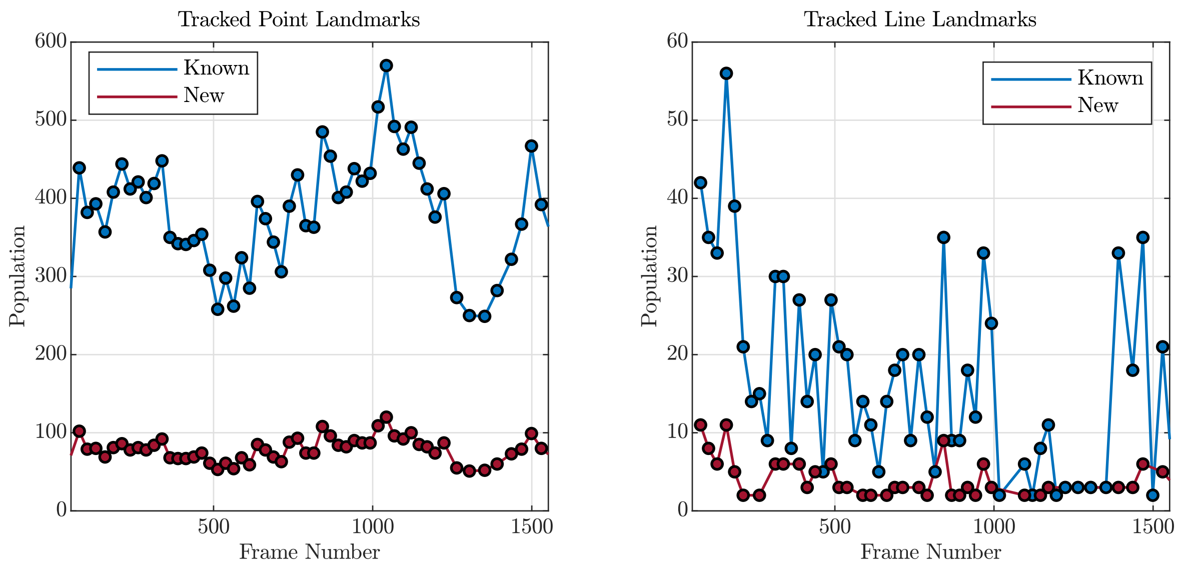


Fig. 18 Population of known tracked and newly triangulated landmarks, for points (left) and lines (right) separately, in every keyframe for DAT2.

[34] Community, B. O., *Blender - a 3D Modelling and Rendering Package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[35] Prussing, J. E., and Conway, B. A., *Orbital Mechanics*, Oxford University Press, USA, 1993.