International Journal of Aeronautical and Space Sciences
https://doi.org/10.1007/s42405-024-00817-2

IJASS International Journal of Aeronautical and Space Sciences

ORIGINAL PAPER

Monocular SLAM with Point and Line Features Applied to Spacecraft Relative Navigation

Ruitao Pan[1,2] · Chenxi Wang[2,3] · Zhi Zhai[2,3] · Jinxin Liu[2,3] · Tianhang Pan[4] · Xuefeng Chen[2,3]

Received: 21 February 2024 / Revised: 8 May 2024 / Accepted: 26 August 2024

Abstract
Real-time estimation of the target's pose is crucial for spacecraft relative navigation. If the target is uncooperative and unknown, i.e., with no prior information, the simultaneous localization and mapping (SLAM) technique is utilized to estimate both the target's pose and 3D shape. Several point-feature-based methods, such as ORB-SLAM, have recently been tested for spacecraft rendezvous. However, point features perform poorly in weak-textured targets and illumination changes, commonly appearing in space environments. This paper presents a monocular SLAM system using point and line features for spacecraft relative navigation. The strengths of different features are fully explored. Specifically, the line feature extraction and matching algorithms are improved, and Plücker coordinates for line representation are used to solve the endpoint inconsistency problem. Moreover, the smoothing approach is utilized for better state estimation while the real-time performance is guaranteed. Compared to the ORB-SLAM, our method is more robust and accurate in complex space environments. Experiments on a challenging dataset demonstrate that adding line features can improve the system's robustness and pose estimation accuracy by 63.6% with a better 3D shape reconstruction. The algorithm runs at 17.83 Hz, satisfying the real-time requirement.

Keywords Relative navigation · Uncooperative and unknown targets · Visual SLAM · Pose estimation · Line features

1 Introduction

On-orbit service (OOS) missions have been conducted more than 130 times since the 1960s [1] and have attracted rapidly increasing attention in recent years. These missions involve spacecraft maintenance and debris removal [2], aiming to extend the spacecraft's lifetime and protect limited orbital resources. The service platforms of OOS can be divided into space shuttles, space stations, and satellites [3]. Satellites have the advantages of autonomy and flexibility. They are the favored option for future development. During the OOS process, navigation to the target is an essential ability of the service satellite (called the chaser), which enables the estimation of the relative pose and feedback to the closed-loop control system in real time.

According to whether targets are equipped with navigation aids, e.g., markers and radio frequency antennas, they can be classified as cooperative targets or uncooperative targets. Currently, most OOS missions are executed on cooperative targets. For uncooperative targets, the relative navigation problem is more difficult, which can be solved by model-known methods and model-unknown methods [4]. If the 3D model of the target is known in advance, the pose can be estimated by the Perspective-n-Points (PnP) algorithm and appearance-based methods [5]. However, no prior information about the target is common, such as for damaged spacecraft and unknown space debris. Research has shown that more than 100 unknown space objects float within the geostationary orbit belt and need to be inspected, repaired, and removed [6]. For uncooperative and unknown targets, the target's pose and 3D shape should be simultaneously estimated. This technique, called Simultaneous Localization and Mapping (SLAM), has been developed for decades in the robotics community. Compared to pose estimation, mapping

Communicated by Donghyun Cho.

✉ Chenxi Wang
wangchenxi@xjtu.edu.cn

1 School of Future Technology, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China

2 National Key Lab of Aerospace Power System and Plasma Technology, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China

3 School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China

4 Shaanxi Fast Gear Co., Ltd., Xi'an, Shaanxi 710119, China

Published online: 11 September 2024

or 3D shape reconstruction is indispensable if the target is uncooperative and unknown. In addition, the reconstructed target's 3D shape is helpful for downstream tasks, such as close rendezvous and maneuvers.

Electro-optical (EO) sensors are popularly used for relative navigation of spacecraft. Compared with Light Detection and Ranging (LIDAR), cameras have the unique advantages of a wide field of view (FOV) and rich color information, making them an essential part of the chaser's awareness system. The method utilizing cameras as input for SLAM is called visual SLAM, one of the hottest topics in computer vision (CV) research [7]. Cameras can be categorized as monocular, stereo, and Time-of-Flight (ToF). Their characteristics give rise to various visual SLAM methods suitable for different rendezvous distances. The stereo and ToF cameras can recover the absolute scale, but they are limited by a small measurement range of less than 100 m. Compared with them, the monocular cameras are applicable for far rendezvous distances greater than 1 km. Although the monocular cameras lose the absolute scale, there is no risk of collision during far rendezvous. In addition, lightweight, low-consumption, and flexible monocular cameras are acceptable for small spacecraft. Therefore, this paper focuses on monocular SLAM applied to the relative navigation of uncooperative and unknown targets. It is worth mentioning that there are deep-learning-based methods [8] for monocular pose estimation. However, the performance of the neural network strongly depends on the quality of the training data, i.e., the difference between the source domain and the target domain, which is still an issue for space image datasets. Monocular SLAM can work regardless of prior information about the target; thus, it is a more general method.

1.1 Related Work

There are two estimation approaches for visual SLAM, i.e., filtering and smoothing (or optimization). Augenstein and Rock [9] first proposed a monocular SLAM method utilizing the Rao–Blackwellized particle filter to estimate an unknown target's pose and 3D shape. The kinematic state is propagated by the linear motion model while the target is tracked and reconstructed by SIFT features [10]. However, the angular rate and velocity of the target in the motion model are regarded as constants. Later, they improved upon previous work by utilizing a hybrid algorithm to solve significant noise covariance caused by the target's unknown mass center when considering dynamics in the motion model, where the Bayesian filter was used to predict rotation, and the measurement inversion was used to predict translation [11]. For detailed motion modeling, Sonnenburg et al. [12] proposed an extended Kalman filter SLAM (EKF-SLAM) integrated with rendezvous kinematics and dynamics. Although this method successfully adapts SLAM to the spacecraft rendezvous problem, the design of the filter is complex, and the target's movement must be restricted to a circular orbit. To simplify dynamic modeling, Barbier and Gao [13] used the Gaussian Processes algorithm, a machine learning method, to model the dynamics of a rotating spacecraft without estimating inertia parameters. Then, an Error-State Kalman filter is used to estimate the spacecraft's pose and 3D shape. For 3D shape estimation, Schnitzer et al. [14] used the Random Sample Consensus (RANSAC) algorithm [15] to remove the outliers of the target's 3D shape reconstructed by EKF-SLAM. However, it is only a post-processing step and hence has no contribution to pose estimation. A novel work combined filter-based SLAM and Structure from Motion (SfM) to refine the target's 3D model [16]. SfM can construct a dense model from multi-view images. Afterward, the 3D model can be used to enhance the visual navigation. However, SfM is an offline method and requires significant computing resources. Filter-based SLAM methods are mainstream in related research [17], but they have several drawbacks compared with the smoothing approach. On the one hand, filters estimate the latest state depending only on the last state under the first-order Markov hypothesis so that it might converge to a local minimum. In contrast, the smoothing approach estimates the latest state employing historical data, whose performance is better than filters under the condition of enough computing resources [18]. On the other hand, filters do not have an anomaly detection mechanism, leading to a reconstructed 3D shape with many outliers.

With the development of spaceborne computers, smoothing is a viable approach for the relative navigation of spacecraft. Tweddle et al. [19] proposed a comprehensive smoothing-based SLAM for a spinning space target. Incremental smoothing and mapping (iSAM) [20] is adapted to estimate the target's movement and inertial parameters, where the motion model incorporates rigid body kinematics and dynamics under the probabilistic factor graph, and the target's 3D shape is tracked and reconstructed by SURF features [21]. The method was successfully evaluated in the International Space Station microgravity environment. Nonetheless, utilizing SURF features and the smoothing approach requires extensive calculations, resulting in a low frame rate. Owing to fast and robust ORB features [22], a lightweight smoothing-based SLAM method, ORB-SLAM [23], was applied to rendezvous to a spacecraft by Dor and Tsiotras [24]. Unlike previous works, the relative dynamics between the chaser and the target are not treated. Experiments were conducted on a video captured from the Hubble Space Telescope [25], which demonstrated the potential for space application. If depth information is available from range sensors, ORB features can be improved to ORBFPFH features [26]. Yan et al. [27] employed it for accurate and robust pose estimation for uncooperative and unknown tar-

gets. Most existing filter-based or smoothing-based SLAM methods leverage point features (SIFT, SURF, ORB, ORBF-PFH) for visual tracking. However, they do not carefully consider illumination changes and weak-textured targets, which strongly affects the algorithm's robustness, resulting in tracking loss. Moreover, the target's 3D shape represented by a spare point cloud is difficult to recognize and is not conducive to subsequent visual tasks. Several SLAM methods can directly manipulate pixels without feature extraction for pose estimation and dense reconstruction. For example, Large Scale Direct SLAM (LSD-SLAM) [28] has been explored for spacecraft pose estimation [29]. However, these methods are based on the grayscale invariance hypothesis, which may be invalid in space environments with remarkable illumination changes.

Fortunately, other features, such as lines and circles, can be used for pose estimation in addition to point features. Meng et al. [30] proposed a single perspective circle and line method for satellite pose estimation. The line segment is used to eliminate the ambiguity caused by the duality of the circle and recover the roll angle around the normal of the circle. Hu and Jiang [31] improved the circle-based pose estimation method with two lines, where descriptors are used to evaluate the error levels under noisy conditions. These methods treat line features as constraints for solving the singularity problem, but the demand of an observed circle limits their application. Liu et al. [32] used 2D–3D line correspondences for spacecraft pose estimation. 2D lines are extracted from images and directly match the lines of the spacecraft's 3D model, and then PnP is used to estimate the target's pose. Bechini et al. [33] proposed a lightweight framework utilizing a learning-based line segment detector with effective PnP (EPnP) to calculate the target's initial pose under the known 3D model, and then the junctions of line segments are used to refine the pose by optimizing keypoint reprojection error. Zhang et al. [34] utilized the target's contour for pose estimation, where 2D–3D correspondences are established by contour feature matching. Liu et al. [35] used different geometric primitives of the target for pose estimation by geometric curve fitting, where the pose was initialized by the target's template. These methods use other robust features instead of point features in complex space environments. However, they highly rely on prior information about the target's 3D model or template. The application of robust features for relative navigation to unknown targets has not been explored yet.

Luckett [36] concluded that for spacecraft pose estimation, point-feature-based methods are the most accurate but with the worst robustness; circle-feature-based methods are the most robust but limited by the small number of circles; line-feature-based methods are robust, and line features frequently appear in artificial objects. Line features can be extracted by Canny edge detector [37] followed by Hough transform [38], which is commonly used for spacecraft edge detection [39, 40]. However, they are sensitive to thresholds with many false detections. Compared with them, Line Segment Detector (LSD) [41] and Edge Drawing Lines (EDLines) [42] do not require parameter tuning. Capuano et al. [43] proposed parallel processing streams for robust feature extraction, where Shi-Tomasi corner detection [44], Hough transform, and LSD are employed to extract point and line features of the spacecraft. Then, the feature synthesis process retains the same features in all three sets. However, utilizing several parallel algorithms increases the number of calculations. Directly employing line feature detection algorithms, such as LSD and EDLines, always leads to poor real-time performance. Therefore, the primary consideration of feature extraction is the trade-off between accuracy and speed. Different from shallow handcrafted features, the neural network has a solid ability to extract high-level abstract features. After training with a large amount of data, learning-based methods achieve high accuracy and a fast inference speed. Bechini et al. [33] explored M-LSD [45] for spacecraft edge detection to accelerate line feature extraction by a convolution neural network (CNN). However, before utilizing the CNN model, it should be retrained on the domain dataset, which still requires significant effort. In space environments, illumination changes can cause pronounced variances in the endpoints of lines, and lines are easily cut into many line segments, which makes feature matching and reconstruction more difficult. Until now, these problems have not been carefully considered in the research.

1.2 Contributions of Paper

This work focuses on monocular SLAM for spacecraft relative navigation in challenging space environments. The strengths of point and line features are fully explored. Compared with the existing research, the contributions of this paper are as follows:

1) A monocular SLAM method is applied to spacecraft relative navigation to the uncooperative and unknown target, which is a difficult problem with no prior information or assistance from range sensors. Different from filter-based methods, smoothing-based SLAM is utilized for state estimation with outlier detection. We are not concerned about the dynamics of the target, which is feasible for monocular-based far rendezvous because the target's inertia parameters are only needed if close operation.
2) We combine point and line features to handle illumination changes and weak-textured targets. Point-feature-based methods have high accuracy for pose estimation. Line features are robust to illumination changes and are usually detectable in artificial objects. Specifically, different line representations are used in our method. Plücker coordinates are used for line reconstruction to solve the

endpoint inconsistency problem, and the intuitive endpoint representation is used for the target's 3D shape. In addition, a unified cost function, including point and line features, is utilized for state estimation.

3) Directly using the line feature detection algorithm can lead to poor performance. Therefore, hidden parameter adjustment is utilized to improve the speed of feature extraction. Moreover, the rejection strategy is used to discard the unqualified line segments, and the merging algorithm is subsequently used to connect the discontinuous line segments, which can generate longer line segments for accurate line feature matching.

The rest of the paper is organized as follows: Sect. 2 states the relative navigation problem modeled by SLAM; Sect. 3 details the proposed monocular SLAM method utilizing point and line features; Sect. 4 validates our method on a public rendezvous dataset; and finally, Sect. 5 provides conclusions and future work.

2 Problem Statement

This section illustrates the mathematical model of typical SLAM adapted to the relative navigation problem. Our method does not restrict the target trajectory, but there is an assumption that the target spins slowly without motion blur. High-speed motion of the target can lead to image artifact, which is a disaster for visual SLAM.

2.1 Reference Frames

State variables are calculated and expressed in different reference frames. It is essential to distinguish them to perform accurate coordinate transformations. The definitions of the reference frames refer to [24], shown in Fig. 1. Typically, SLAM assumes the environment is static, with a mobile robot moving around. In our scenario, even though the target floats and spins in space, SLAM is still applicable when considering the relative motion of a rigid body of the target.

Earth Centered Inertial (ECI) frame provides global coordinates of spacecraft positions with respect to the Earth. It is defined as $\mathcal{E} = \{\mathrm{E}; \hat{e}_x, \hat{e}_y, \hat{e}_z\}$, where E is the origin located at the center of the Earth, and $\hat{e}_x, \hat{e}_y, \hat{e}_z \in \mathbb{S}^2$ are three orthogonal unit vectors. $\hat{e}_z$ points along the Earth's rotation axis toward the poles, $\hat{e}_x$ lies in the equatorial plane and points to the equinox, and $\hat{e}_y = \hat{e}_z \times \hat{e}_x$ is determined by the right-hand Cartesian coordinate system. The positions of the chaser and the target expressed in the frame $\mathcal{E}$ are given by $r_{\mathrm{S}}^{\mathcal{E}}, r_{\mathrm{T}}^{\mathcal{E}} \in \mathbb{R}^3$, respectively, where S is the chaser's center of mass, and T is the target's center of mass. Thus their relative position is given by $r^{\mathcal{E}} = r_{\mathrm{T}}^{\mathcal{E}} - r_{\mathrm{S}}^{\mathcal{E}}$.

The chaser's fixed-body frame describes the attitude changes of the spacecraft itself. It is defined as $\mathcal{S} = \{\mathrm{S}; \hat{s}_x, \hat{s}_y, \hat{s}_z\}$, where S is the origin, and the unit vectors $\mathcal{S} = \{\mathrm{S}; \hat{s}_x, \hat{s}_y, \hat{s}_z\}$ are aligned with the chaser's principal axes of inertia. The frame $\mathcal{S}$ is calibrated before launch. Attitude can be represented by rotation matrix, axis angle, Euler angles, and quaternions. A group of rotation

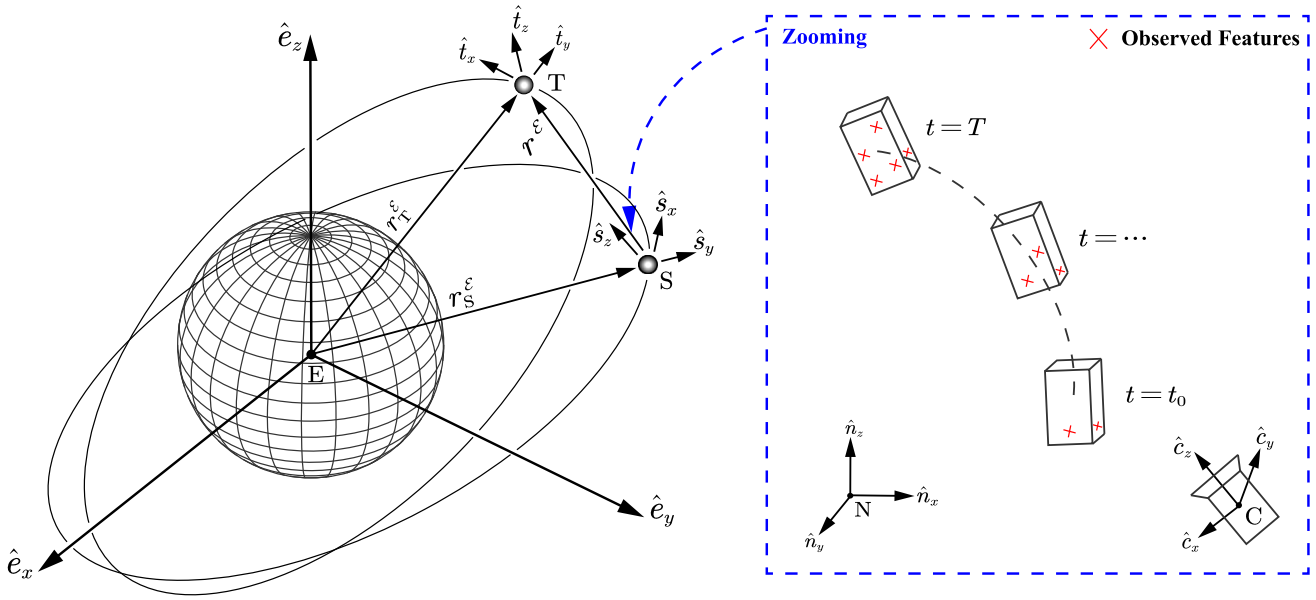


Fig. 1 Reference frames for relative navigation

matrices R forms a special orthogonal group $\mathrm{SO}(3) = \left\{R \in \mathbb{R}^{3\times3} | RR^{\mathrm{T}} = I, \det(R) = 1\right\}$. Euler angles are intuitive for visualization, among which yaw, pitch, and roll angles demonstrate rotation around $\hat{s}_z, \hat{s}_y, \hat{s}_x$, respectively. Quaternions q are non-singular and usually used for calculation.

The camera's fixed-body frame records the local coordinates of the target under the camera's view. It is defined as $\mathcal{C} = \left\{\mathrm{C}; \hat{c}_x, \hat{c}_y, \hat{c}_z\right\}$, where C is the origin located at the optical center of the camera, $\hat{c}_z$ is aligned with the camera's optical direction, $\hat{c}_x$ is parallel to the x-axis direction of the image plane, and $\hat{c}_y = \hat{c}_z \times \hat{c}_x$. The coordinate transformation between the frame $\mathcal{S}$ and the frame $\mathcal{C}$ is known in advance from the designer. For convenience, we usually treat the pose estimation problem as the estimation of the target's pose with respect to the frame $\mathcal{C}$ instead of the frame $\mathcal{S}$ because features are observed in the frame $\mathcal{C}$.

Similarly, the target's fixed-body frame is defined as $\mathcal{T} = \left\{\mathrm{T}; \hat{t}_x, \hat{t}_y, \hat{t}_z\right\}$, where T is the origin, and the unit vectors $\hat{t}_x, \hat{t}_y, \hat{t}_z \in \mathbb{S}^2$ are aligned with the target's principal axes of inertia. In our method, we are not concerned about relative dynamics, i.e., $\mathcal{T}$ is not estimated. Instead, the target's attitude is expressed by an interested frame selected by the algorithm, called the feature frame. To some extent, it is still a pose estimation problem because the only difference between the frame $\mathcal{T}$ and the feature frame is a fixed coordinate transformation. An obvious benefit is that it simplifies the motion model without considering relative dynamics in complex space environments.

Feature frame is used to express the localization of the target's components. It is defined as $\mathcal{N} = \left\{\mathrm{N}; \hat{n}_x, \hat{n}_y, \hat{n}_z\right\}$, where N is the origin, and $\hat{n}_x, \hat{n}_y, \hat{n}_z \in \mathbb{S}^2$ are three orthogonal unit vectors. The target's components are represented by point and line features. They consist of the target's 3D shape, i.e., the 3D map in the typical SLAM concept. The frame $\mathcal{N}$ is determined by the algorithm in the pose initialization process.

There are several coordinate transformations among these reference frames. In a projective space $\mathbb{P}^3$ ($\mathbb{P}^n = \mathbb{R}_{\neq0} \times \mathbb{R}$), for a homogeneous vector $\bar{x}^{\mathcal{A}} \in \mathbb{P}^3$ expressed in the frame $\mathcal{A}$ and a homogeneous vector $\bar{x}^{\mathcal{B}} \in \mathbb{P}^3$ expressed in the frame $\mathcal{B}$, a rotation matrix $R_{\mathcal{A}}^{\mathcal{B}} \in \mathrm{SO}(3)$ and a translation vector $t_{\mathcal{A}}^{\mathcal{B}} \in \mathbb{R}^3$ represents a rigid body transformation:

$$\bar{x}^{\mathcal{B}} = \begin{bmatrix} R_{\mathcal{A}}^{\mathcal{B}} & t_{\mathcal{A}}^{\mathcal{B}} \\ 0 & 1 \end{bmatrix} \bar{x}^{\mathcal{A}} \tag{1}$$

The transformation pair $R_{\mathcal{A}}^{\mathcal{B}}, t_{\mathcal{A}}^{\mathcal{B}}$ is equivalent to a transformation matrix $T_{\mathcal{A}}^{\mathcal{B}}$, given by:

$$T_{\mathcal{A}}^{\mathcal{B}} \triangleq \begin{bmatrix} R_{\mathcal{A}}^{\mathcal{B}} & t_{\mathcal{A}}^{\mathcal{B}} \\ 0 & 1 \end{bmatrix} \tag{2}$$

A group of transformation matrix T forms a special Euclidean group $\mathrm{SE}(3) = \left\{T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} | R \in \mathrm{SO}(3), t \in \mathbb{R}^3\right\}$, whose continuous and smooth properties are used for nonlinear optimization.

Pose estimation aims to calculate the rigid body transformation $T_{\mathcal{T}}^{\mathcal{S}}$. However, there is no information about the frame $\mathcal{T}$, which is not studied in this paper. Note that features expressed in the frame $\mathcal{N}$ are initialized at time t_0, and we consider the frame $\mathcal{N}_{\{t=t_0\}}$ to be equivalent to the frame $\mathcal{T}$ for representation of the target. After that, when the camera tracks the target, $\mathcal{N}_{\{t\}}$ changes over time. Thus, the pose estimation objective is converted to calculate $T_{\mathcal{N}}^{\mathcal{S}}$. For convenience, we usually estimate $T_{\mathcal{N}}^{\mathcal{C}}$, given by:

$$T_{\mathcal{N}}^{\mathcal{C}} = \left(T_{\mathcal{C}}^{\mathcal{S}}\right)^{-1} T_{\mathcal{N}}^{\mathcal{S}} \tag{3}$$

where $T_{\mathcal{C}}^{\mathcal{S}}$ is available for the designer. Here, we assume that it is an identity matrix.

If the global location of the camera $T_{\mathcal{C}}^{\mathcal{E}}$ is available by onboard sensors or Global Navigation Satellite System (GNSS), the global location of the target $T_{\mathcal{N}}^{\mathcal{E}}$ can be calculated by:

$$T_{\mathcal{N}}^{\mathcal{E}} = T_{\mathcal{C}}^{\mathcal{E}} T_{\mathcal{N}}^{\mathcal{C}} \tag{4}$$

It is important to realize that monocular SLAM estimates the pose $T_{\mathcal{N}}^{\mathcal{C}}$ with a scale ambiguity:

$$T_{\mathcal{N}}^{\mathcal{C}} = \begin{bmatrix} R_{\mathcal{N}}^{\mathcal{C}} & \lambda t_{\mathcal{N}}^{\mathcal{C}} \\ 0 & 1 \end{bmatrix} \tag{5}$$

where λ is a scale factor estimated by range sensors, which is not considered in our method.

2.2 Motion Model and Measurement Model

SLAM is a state estimation problem that is formulated as follows:

$$x_k = f(x_{k-1}, u_k) + \omega_k, \quad k = 1, \ldots, N \tag{6}$$

$$z_{k,j} = h\left(x_k, y_j\right) + v_{k,j}, \quad j = 1, \ldots, M \tag{7}$$

For the motion model of Eq. (6), x_k represents the target's pose with respect to the camera at time k, defined as $x_k = T_{\mathcal{N}\{t=k\}}^{\mathcal{C}}$. For convenience, We denote this as $x_k = T_k$. It can be converted to the pose vector $x = \left[r^{\mathrm{T}}, q^{\mathrm{T}}\right]^{\mathrm{T}}$, where the position is $r = \left[r_x, r_y, r_z\right]^{\mathrm{T}}$, and the attitude is $q = [q_1, q_2, q_3, q_4]^{\mathrm{T}}$. u_k is the control input obtained by motion

sensors. ω_k is noise under zero-mean Gaussian distribution with covariance R_k. The target's pose is propagated by $f(\cdot)$ (here x_k is regarded as the pose vector):

$$x_k = x_{k-1} + \dot{x}_{k-1}\Delta t + \omega_k \tag{8}$$

where $\Delta t = t_k - t_{k-1}$ and $\omega_k \sim N(0, R_k)$. There are two reasons for modeling $f(\cdot)$ as a speed motion model. First, because the target is in motion, the maneuver of the chaser is not equal to u_k, i.e., the relative motion depends on both the chaser's movement and the target's movement. One solution is to use the relative speed for pose propagation, which is calculated by the two previous estimated poses. Second, not considering rendezvous dynamics can simplify the motion model in complex space environments.

For the measurement model of Eq. (7), y_j represents the jth feature composing the target's 3D shape, and $z_{k,j}$ is the measurement of y_j at time k. $v_{k,j}$ is noise under zero-mean Gaussian distribution with covariance $Q_{k,j}$. Here, $h(\cdot)$ is a pinhole camera model projecting the 3D world onto the 2D image, shown in Fig. 2.

First, the feature $P^{\mathcal{N}} = [X, Y, Z]$ expressed in the frame $\mathcal{N}$ is transferred to $P^{\mathcal{C}} = [X', Y', Z']$ expressed in the frame $\mathcal{C}$. Then it is projected onto the image plane $p^{\mathcal{C}} = [x, y, z]$ under a pinhole camera model, where the feature loses depth information. After that, images are generated on the pixel plane $[u, v]$. The process is calculated by:

$$\begin{aligned} Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= K P^{\mathcal{C}} = [K|0]\, T_{\mathcal{N}}^{\mathcal{C}} \bar{P}^{\mathcal{N}} \\ &= \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{\mathcal{N}}^{\mathcal{C}} & t_{\mathcal{N}}^{\mathcal{C}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \end{aligned} \tag{9}$$

K is the intrinsic matrix of the camera, calibrated by [46], where $f_x = \alpha f$, $f_y = \beta f$, and $[c_x, c_y]$ are the pixel coordinates of the center of the pixel plane. α is the scale factor of the x-axis from the image plane to the pixel plane, β is the scale factor of the y-axis from the image plane to the pixel plane, and f is the focal length.

For the 3D features y_j and its 2D measurement $z_{k,j}$, according to Eq. (9), the projection process is modeled as follows:

$$s\bar{z}_{k,j} = K\left(R_k y_j + t_k\right) + v_{k,j} \tag{10}$$

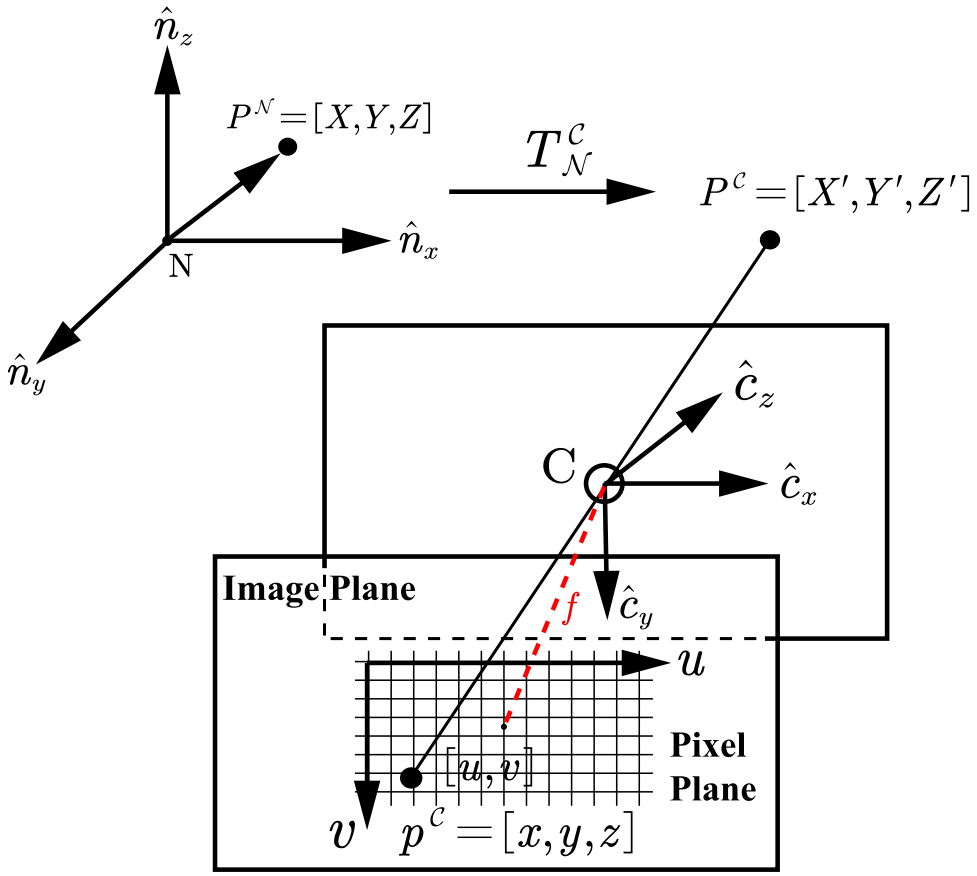


Fig. 2 The measurement model

where s is the depth of the feature y_j, R_k, t_k are the rotation matrix and the translation vector of the target's pose T_k, respectively, and $v_{k,j} \sim N\left(0, Q_{k,j}\right)$.

The state is predicted by the motion model and updated by the measurement model. If the initial state x_0, the control input u_k, and the measurement $z_{k,j}$ are known, the pose x_k and the features y_j can be iteratively estimated according to Eq. (6) and Eq. (7), which is known as the filter approach. For the smoothing approach, a batch of data is utilized for better estimation, which can be divided into full SLAM that processes all historical data (called global optimization) and sliding-window-based SLAM that processes data within the window length (called local optimization). Considering real-time performance, it is impractical to implement global optimization constantly. Thus, global optimization is executed only at some special moments while local optimization is used for ordinary processing.

To model the batch state estimation problem, we define:

$$x = \{x_1, \ldots, x_N\} \tag{11}$$
$$y = \{y_1, \ldots, y_M\} \tag{12}$$
$$u = \{u_1, \ldots, u_k\} \tag{13}$$
$$z = \left\{z_{1,j}, \ldots, z_{k,j}\right\} \tag{14}$$

The SLAM problem is viewed as solving the probability distribution of x, y with known control input u and measurement z, i.e., $P\left(x, y|z, u\right)$. Using the Bayes rule, we have:

$$\begin{aligned} P\left(x, y|z, u\right) &= \frac{P\left(z, u|x, y\right) P\left(x, y\right)}{P\left(z, u\right)} \\ &\propto P\left(z, u|x, y\right) P\left(x, y\right) \end{aligned} \tag{15}$$

It is difficult to solve the posterior probability distribution directly, and we would like to find the optimal estimate of the state. Generally, we do not know the prior $P(x, y)$, and then it turns to a maximum likelihood estimation (MLE) problem (equivalently minimizing the negative log-likelihood):

$$\begin{aligned} (x, y)^* &= \arg\max P\left(z, u|x, y\right) \\ &= \arg\min -\log P\left(z, u|x, y\right) \end{aligned} \tag{16}$$

Assuming that each input and measurement is independent, we have:

$$\begin{aligned} P\left(z, u|x, y\right) = &\prod_k P\left(u_k|x_{k-1}, x_k\right) \\ &\prod_{k,j} P\left(z_{k,j}|x_k, y_j\right) \end{aligned} \tag{17}$$

Furthermore, because $\omega_k \sim N\left(0, R_k\right)$ and $v_{k,j} \sim N\left(0, Q_{k,j}\right)$, we have:

$$P\left(u_k|x_{k-1}, x_k\right) \propto \exp\left(-\frac{1}{2}\left(x_k - f\left(x_{k-1}, u_k\right)\right)^{\mathrm{T}} R_k^{-1}\left(x_k - f\left(x_{k-1}, u_k\right)\right)\right) \tag{18}$$
$$P\left(z_{k,j}|x_k, y_j\right) \propto \exp\left(-\frac{1}{2}\left(z_{k,j} - h\left(x_k, y_j\right)\right)^{\mathrm{T}} Q_{k,j}^{-1}\left(z_{k,j} - h\left(x_k, y_j\right)\right)\right) \tag{19}$$

Substituting Eq. (17), Eq. (18) and Eq. (19) into Eq. (16), the MLE problem turns to the least squares problem:

$$\begin{aligned} (x, y)^* = \arg\min &\sum_k \left(x_k - f\left(x_{k-1}, u_k\right)\right)^{\mathrm{T}} R_k^{-1}\left(x_k\right. \\ &\left. - f\left(x_{k-1}, u_k\right)\right) + \sum_{k,j}\left(z_{k,j} - h\left(x_k, y_j\right)\right)^{\mathrm{T}} Q_{k,j}^{-1} \\ &\left(z_{k,j} - h\left(x_k, y_j\right)\right) \end{aligned} \tag{20}$$

The errors of each input and measurement are defined as follows:

$$e_{u_k} = x_k - f\left(x_{k-1}, u_k\right) \tag{21}$$
$$e_{z_{k,j}} = z_{k,j} - h\left(x_k, y_j\right) \tag{22}$$

Therefore, Eq. (20) can be simplified as follows:

$$(x, y)^* = \arg\min \sum_k e_{u_k}^{\mathrm{T}} R_k^{-1} e_{u_k} + \sum_k e_{z_{k,j}}^{\mathrm{T}} Q_{k,j}^{-1} e_{z_{k,j}} \tag{23}$$

This can be solved by the nonlinear optimization method. Actually, we do not consider the motion error because u_k is unavailable. Despite that, the motion model can provide an initial estimated pose, and then the pose is optimized based on the measurement error, which is detailed in Sect. 3.1.3. Finally, we construct a least squares problem that includes only the measurement error:

$$\min J\left(x, y\right) = \sum_k e_{z_{k,j}}^{\mathrm{T}} Q_{k,j}^{-1} e_{z_{k,j}} \tag{24}$$

3 Methods

This section introduces our method adapted from the ORB-SLAM [23]. Typical SLAM algorithms consist of a front-end tracking thread and a back-end mapping thread, which can simultaneously estimate the camera pose and reconstruct the environment. In addition, a loop closure thread is utilized to detect the place where the camera has been seen before, and then a similarity wrap is performed to correct the cumulative errors. The multi-threaded strategy ensures the system's real-time requirement. ORB-SLAM follows the above framework with a faster running speed. Unfortunately, point features are not robust for spacecraft relative navigation due to illumination changes and weak-textured targets.

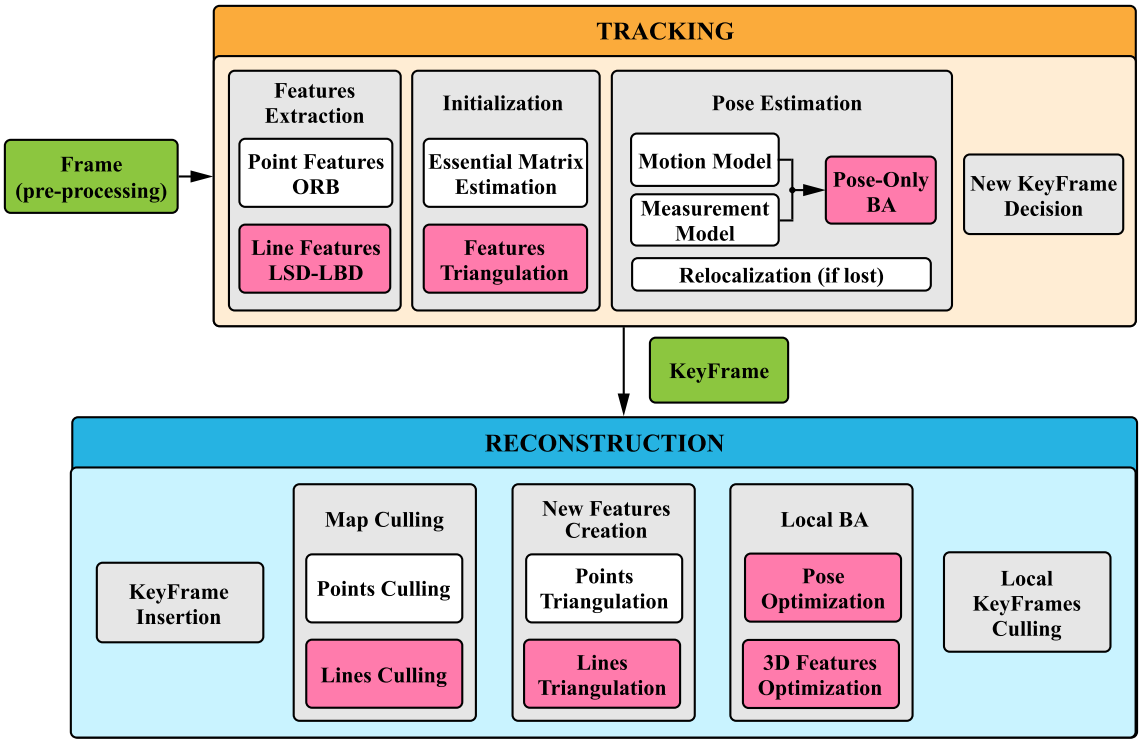


Fig. 3 The proposed architecture

Therefore, the proposed method incorporates point and line features to improve the system's robustness. The algorithm's architecture is shown in Fig. 3. The difference between the ORB-SLAM and our method is especially highlighted in pink. We do not consider loop closure for fair performance comparison because the ORB-SLAM can not work well in our experimental conditions. More details are introduced in the following.

3.1 Front-End: Target Tracking

Some image pre-processing processes, including mask cropping and Gaussian filtering, should be considered. Object detection must be implemented first because the target occupies only a small portion of the image area. In this work, we do not focus on this problem and use a mask to crop the target from the background. After that, Gaussian filtering is employed for image noise. Then, the sequence images are fed into the tracking thread.

3.1.1 Feature Extraction

Implementing ORB features is consistent with the ORB-SLAM, except for assigning features by quadtree. The quadtree divides the image into several small squares, and then the ORB features with high quality are uniformly samples from the squares. This strategy can improve the accuracy of pose estimation. However, we do not adopt the quadtree because the target is weak-textured so that a few ORB features can be detected for pose estimation.

Line features are extracted by Line Segment Detector (LSD) [41]. However, directly employing LSD leads to poor real-time performance. Therefore, we use hidden parameter adjustment to improve the speed of feature extraction. Fu et al. [47] reported that although LSD does not require parameter tuning, several hidden parameters can be modified to speed up detection: the scale of the image $s = (0, 1]$ and the minimal density threshold d. The scale s is used for image downsampling before detecting line segments. The smaller the image size, the fewer pixels need to be processed, i.e., with faster speed. The minimal density threshold d is used to reject pixels with small gradients for line segment generation. The generation speed can be improved by reducing the number of considered pixels. These parameters should be selected considering the trade-off between accuracy and speed.

In this work, we set $s = 0.6$ (default 0.8) and $d = 0.6$ (default 0.7) by controlled variable studies, shown in Fig. 4. We run the algorithm with different parameter settings. Root mean square error (RMSE) of absolute trajectory error (ATE) [48] is used to calculate the accuracy of pose estimation. Notice that decreasing the parameter s can significantly reduce the time consumption with negligible accuracy loss, and the effect of reducing the parameter d is limited because

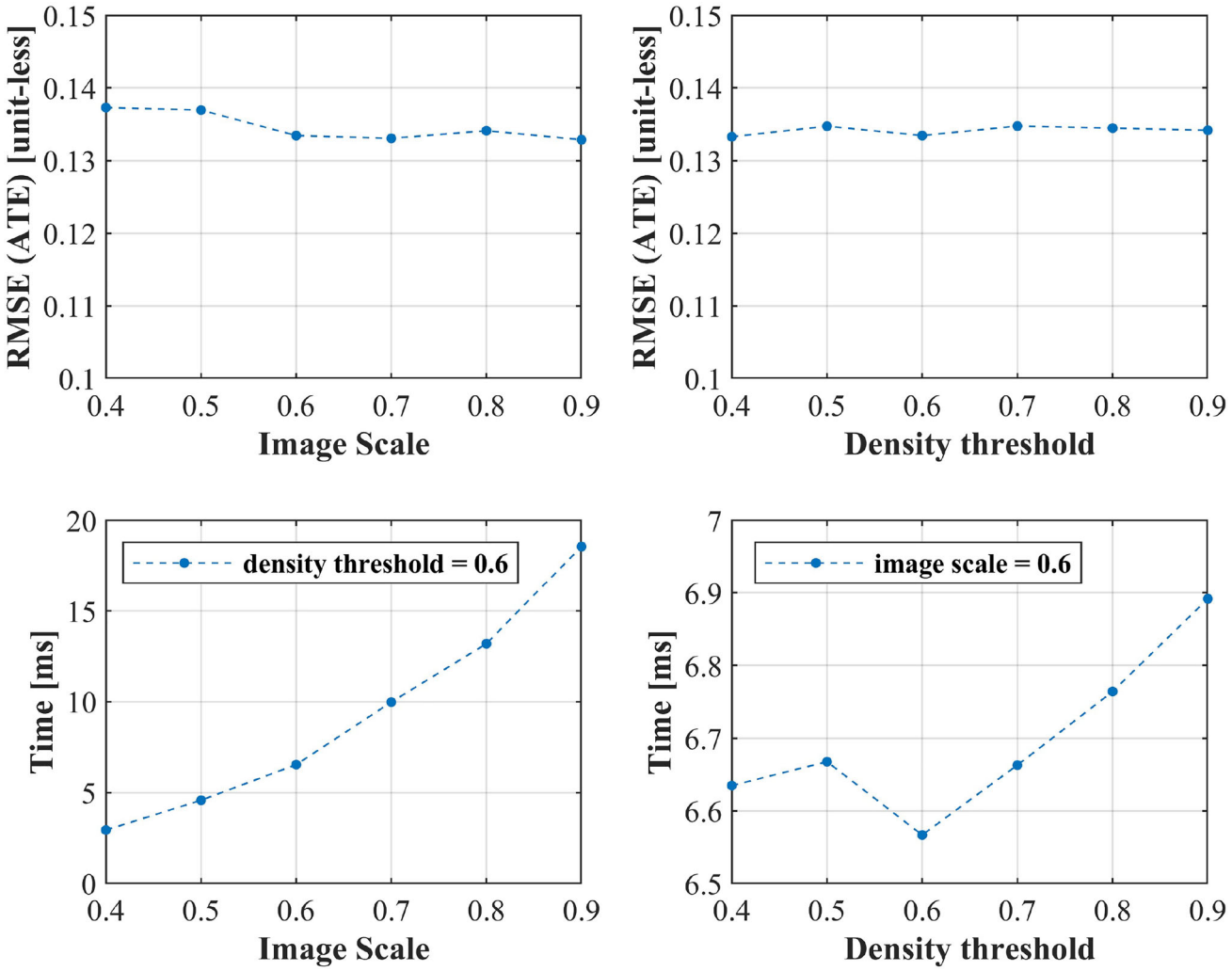


Fig. 4 Hidden parameter adjustment

most areas of images are black space backgrounds with small gradients.

Line features are described by Line Band Descriptor (LBD) [49], which is used for line feature matching. Before computing the LBD of line segments, we implement a rejection strategy and merge disconnected line segments for accurate line feature matching. The rejection strategy includes two stages, i.e., length suppression and response suppression. We consider the line segments whose lengths are less than 30 pixels to be noise. This process also rejects some real short-line segments, but their contribution to state estimation is negligible. Furthermore, we preserve only the top 40 line segments with the largest response, which can decrease the number of line segments to accelerate the matching process. After that, we utilize a merging algorithm to connect line segments from the same line.

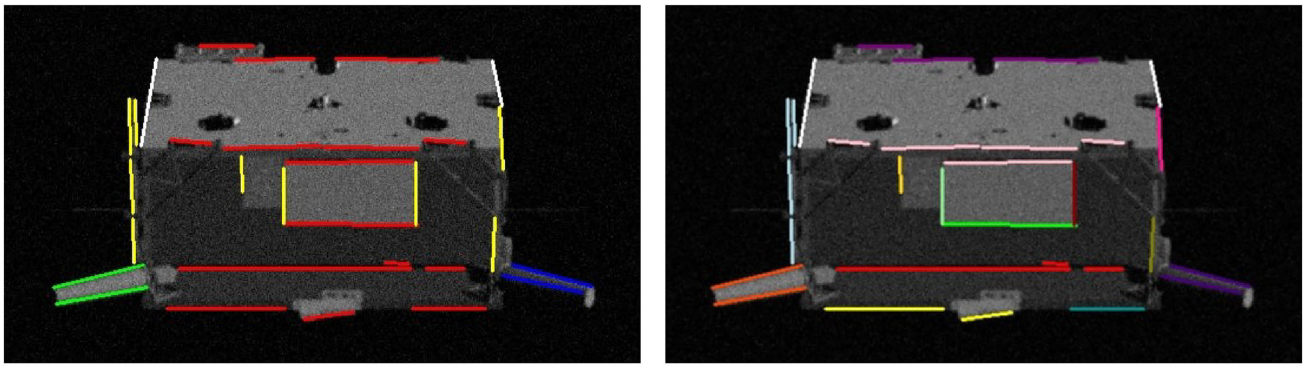
We adopt the merging algorithm [50] to connect line segments. First, line segments are sorted in descending order according to length to reduce the impact of short-line segments because long line segments are credible with continuously strong gradients. Then, line segments are clustered based on angular similarity and spatial proximity, shown in Fig. 5. Line segments are clustered with different colors. In particular, the line segments with white color are not classified.

Suppose all line segments are involved in $\mathscr{L}$. For the line segment L_1, the group G_1 with close angular similarity is selected by:

$$G_1 = \{\forall L_2 \in \mathscr{L} : |\theta_2 - \theta_1| < \tau_\theta\} \tag{25}$$

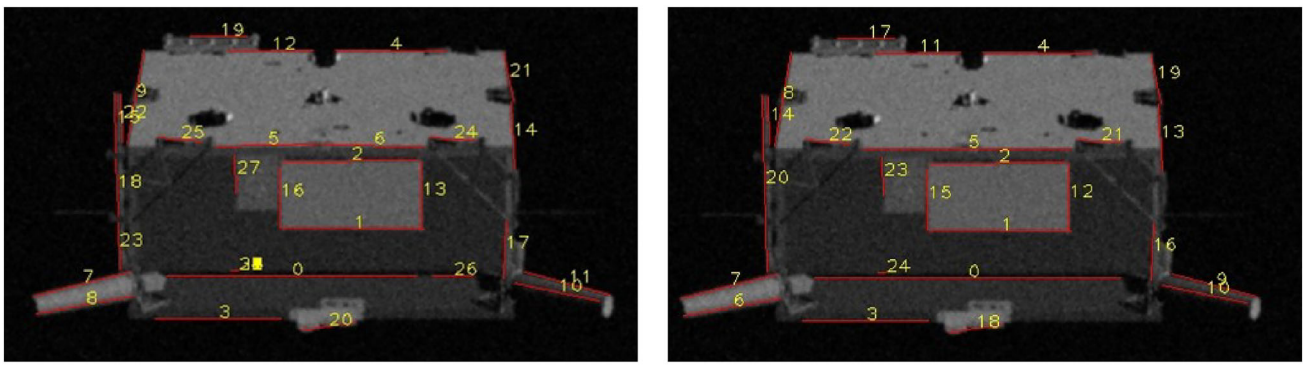
where θ_1, θ_2 are the angles of L_1, L_2, respectively, and τ_θ is set to 4 degrees.

Line segments are further grouped based on spatial proximity. Specifically, the distances between the endpoints of L_1 and the endpoints of L_2 are calculated, and then the group G_2 with close spatial proximity is selected by:



(a) Angular similarity (b) Spatial proximity

Fig. 5 Grouping line segments



(a) Before merging (b) After merging

Fig. 6 Line segment merging

$$G_2 = \{\forall L_2 \in G_1 : d(s_1, s_2) < \tau_s \vee d(e_1, e_2) < \tau_s \vee d(s_1, e_2) < \tau_s \vee d(e_1, s_2) < \tau_s\} \tag{26}$$

where s_n, e_n are the start point and end point of L_n, respectively, $d(\cdot)$ is the Euclidean distance, and $\tau_s = \xi_s |L_1|$ is an adaptive thresholds related to the length of L_1. Here, we set $\xi_s = 0.2$.

Afterward, the line segments in group G_2 are merged according to the perception of mergeability, which is inversely proportional to the length of the shorter line segment, angular difference, and relative spatial distance. Finally, the newly generated line segment is rejected by a threshold τ_θ^*, representing the angular difference between the new line segment and L_1, calculated by:

$$\tau_\theta^* = \left(1 - \frac{1}{1 + e^{-2(\lambda - 1.5)}}\right) \tau_\theta \tag{27}$$

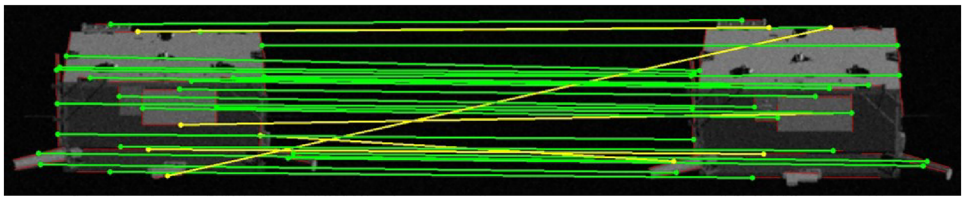
where $\lambda = \frac{|L_2|}{|L_1|} + \frac{d}{\tau_s}$ (d is the closest distance between the endpoints of L_1 and the endpoints of L_2).

The merging algorithm is the post-processing of line feature detection, shown in Fig. 6. There are 40 line segments detected at first. After merging, 25 line segments are reserved.

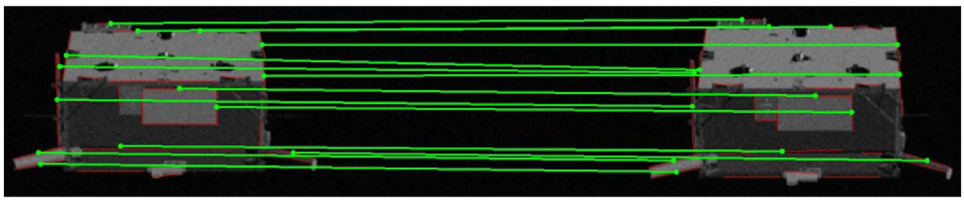
In the line feature matching process, mismatches may occur. Line segment merging algorithm is helpful to improve matching performance, shown in Fig. 7. We use the angle histogram to detect outliers, whose angles are significantly different from the overall. The outliers are colored in yellow. There are 5 outliers before merging among 37 matching lines. After merging, there is no outlier among 14 matching lines. In this way, the accuracy of feature matching can be improved.

3.1.2 Initialization

If there are enough point features with accurate matching between two images, the initialization process executes the same as the ORB-SLAM. We briefly introduce the estimation of the transformation R, t between two image frames. Assuming that pixels on the first frame is $p_1 = [u_1, v_1]^{\mathrm{T}}$, and pixels on the second frame is $p_2 = [u_2, v_2]^{\mathrm{T}}$, they satisfy the epipolar constraint:



(a) The result of matching before merging



(b) The result of matching after merging

Fig. 7 Line feature matching by LBD

$$\bar{p}_2^{\mathrm{T}} K^{-\mathrm{T}} t^{\wedge} R K^{-1} \bar{p}_1 = 0 \tag{28}$$

where K is the camera intrinsic matrix and $t^{\wedge} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ t_2 & t_1 & 0 \end{bmatrix}$ is the skew-symmetric matrix of $t = [t_1, t_3, t_3]^{\mathrm{T}}$. Eq. (28) can be simplified as follows:

$$x_2^{\mathrm{T}} E x_1 = 0 \tag{29}$$

where $x_1 = K^{-1}\bar{p}_1$, $x_2 = \bar{p}_2^{\mathrm{T}} K^{-\mathrm{T}}$ and $E = t^{\wedge} R$, called essential matrix. The unknown parameters of the matrix E are eight up to a scale for the monocular camera. It is calculated by the eight-point algorithm [51]. Finally, the singular value decomposition (SVD) is used to recover the transformation R, t from the matrix E with positive depth verification.

After successful initialization, the initial two images' 3D point and line features are constructed by triangulation, which is detailed in Section 3.2.1. The feature frame $\mathcal{N}$ is considered to coincide with the first camera frame. Once we have defined the feature frame $\mathcal{N}$, we can estimate the target's pose $T_{\mathcal{N}}^{\mathcal{C}}$ according to the motion model and the measurement model. Meanwhile, the target's 3D shape is simultaneously reconstructed from the keyframes.

3.1.3 Pose Estimation

As mentioned above, the initial pose is provided by the motion model, and then the refined target's pose is optimized by the measurement error. The measurement error is calculated by the reprojection process, shown in Fig. 8. p, $p\prime$ are the projected point and the matching point, respectively. The projected line segment l is represented by the endpoints x_s and x_e. Its matching line segment l' is represented by the endpoints x_s' and x_e'.

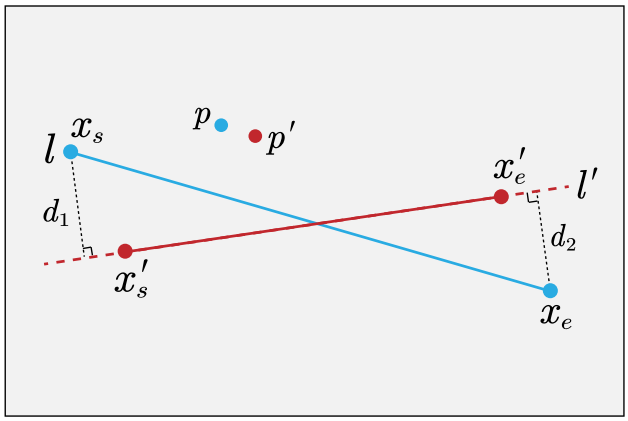


Fig. 8 The reprojection error

When 3D features are projected onto the current frame, the projected 2D features are inconsistent with their corresponding measurements. The correspondence 2D features with the minimum Hamming distance are searched by ORB for points and LBD for line segments. Then, the reprojection errors of point and line features are defined as:

$$E_p : \left\| p - p' \right\| \tag{30}$$

$$E_l : \{d_1, d_2\} = \left\{ x_s^T l_c', x_e^T l_c' \right\} \tag{31}$$

where $\|\cdot\|$ is the Euclidean distance, and l_c' is the line coefficient calculated by:

$$l_c' = \frac{\bar{x}_s' \times \bar{x}_e'}{\left\| \bar{x}_s' \times \bar{x}_e' \right\|} \tag{32}$$

We utilize a unified cost function including point and line features [52] to represent the reprojection error of the current frame. The error terms are comparable when lines are represented by their endpoints. Considering the reprojection

errors of 3D points X_i on the image I_k represented by $e_{i,j}$, and the reprojection errors of 3D lines L_j on the image I_k represented by $e'_{k,j} = d_1$ and $e''_{k,j} = d_2$, the cost function is given by:

$$J = e_{k,i}^T \Omega_{k,i}^{-1} e_{k,i} + e_{k,j}'^T \Omega'_{k,j}{}^{-1} e'_{k,j} + e_{k,j}''^T \Omega''_{k,j}{}^{-1} e''_{k,j} \quad (33)$$

$\Omega_{k,i}, \Omega'_{k,j}, \Omega''_{k,j}$ are the covariance matrices related to the detected scale levels of the point and line features. Eq. (33) is solved by the Levenberg–Marquardt algorithm [53]. In the tracking thread, we optimize only the target's current pose, while in the reconstruction thread, we use bundle adjustment to optimize both the target's pose and the 3D features, which is detailed in Sect. 3.2.2.

In the optimization process, we detect outliers using the chi-square test. The reprojection error e obeys zero-mean Gaussian distribution with its correspondence covariance Σ. The higher the number of pyramid layers is, the lower the feature extraction accuracy with larger covariance. The statistic of the chi-square test is calculated by $r = e^{\mathrm{T}} \Sigma^{-1} e$. For the monocular camera, the statistical threshold is set to $r_t = 5.99$ at a significance level of 0.05. Thus, we consider the features as outliers if $r > r_t$.

If the tracking of the target is lost, we perform relocalization. All 3D point features project onto the current frame, and the target's pose is estimated by the EPnP algorithm [54].

3.2 Back-End: Target Reconstruction

The target is tracked continuously while its 3D shape is estimated by the reconstruction thread. For real-time performance, the keyframe strategy is utilized as a sliding window. Keyframes are selected according to the number of observed features of the current frame and the duration between the last keyframe and the current frame. More observed features and longer durations vote for the current frame as the keyframe. We only operate the keyframes for the target's 3D shape reconstruction.

The number of keyframes increases with the time passing by. Thus, we conduct keyframe culling to reduce redundancy in the final process of the reconstruction thread. If the features of the current keyframe can be abundantly observed for the other keyframes, we consider the current keyframe to be redundant and delete it.

3.2.1 Map Creation and Culling

The map concept for typical SLAM is the same as the target's 3D shape in our problem. The map module manages 3D features, keyframes and covisibility graph. Specifically, the covisibility graph is created to save the co-view relationship between keyframes. Two keyframes have a co-view relationship if they observe several of the same features. It is used in the optimization process to search related keyframes and 3D features.

In the map, 3D features are reconstructed by triangulation. For line features, we can directly triangulate their endpoints like point features. However, it will cause a significant error because differences in the matching endpoints are pronounced under illumination changes, shown in Fig. 9(a). To solve the endpoint inconsistency problem, we utilize Plücker coordinates to represent an infinite 3D line. In Plücker coordinates, a 6D vector $L = \left(m^T, d^T\right)$ is used to describe a 3D line, where $m \in \mathbb{R}^3$, called the momentum vector, is normal to the plane of the line L intersecting the origin, and $d \in \mathbb{R}^3$ represents the line direction, shown in Fig. 9(b). The 3D line can be reconstructed by forward projecting the matching 2D line segments l_1 and l_2 from two images [55]. There are two 3D planes π_1 and π_2 across the line segments and the viewpoints, shown in Fig. 9(c), which are calculated by:

$$\pi_1 = P_1^{\mathrm{T}} l_1, \quad \pi_2 = P_2^{\mathrm{T}} l_2 \quad (34)$$

where $P_i = [K|0]\, T_i$ is the 3×4 projection matrix involving the camera's intrinsic and its pose, and $l_i = \bar{x}_{is} \times \bar{x}_{ie}$.

Then, the Plücker coordinates of the 3D line can be extracted from the dual Plücker matrix L^*, given by:

$$L^* = \pi_1 \pi_2^T - \pi_2 \pi_1^T = \begin{bmatrix} [d]_\times & m \\ -m^T & 0 \end{bmatrix} \quad (35)$$

Finally, we covert the Plücker coordinates to the endpoints representation by endpoint trimming [56], shown in Fig. 9(d), which is implemented by selecting a line segment l_t parallel to the image's x-axis and creating a plane π_t for cutting:

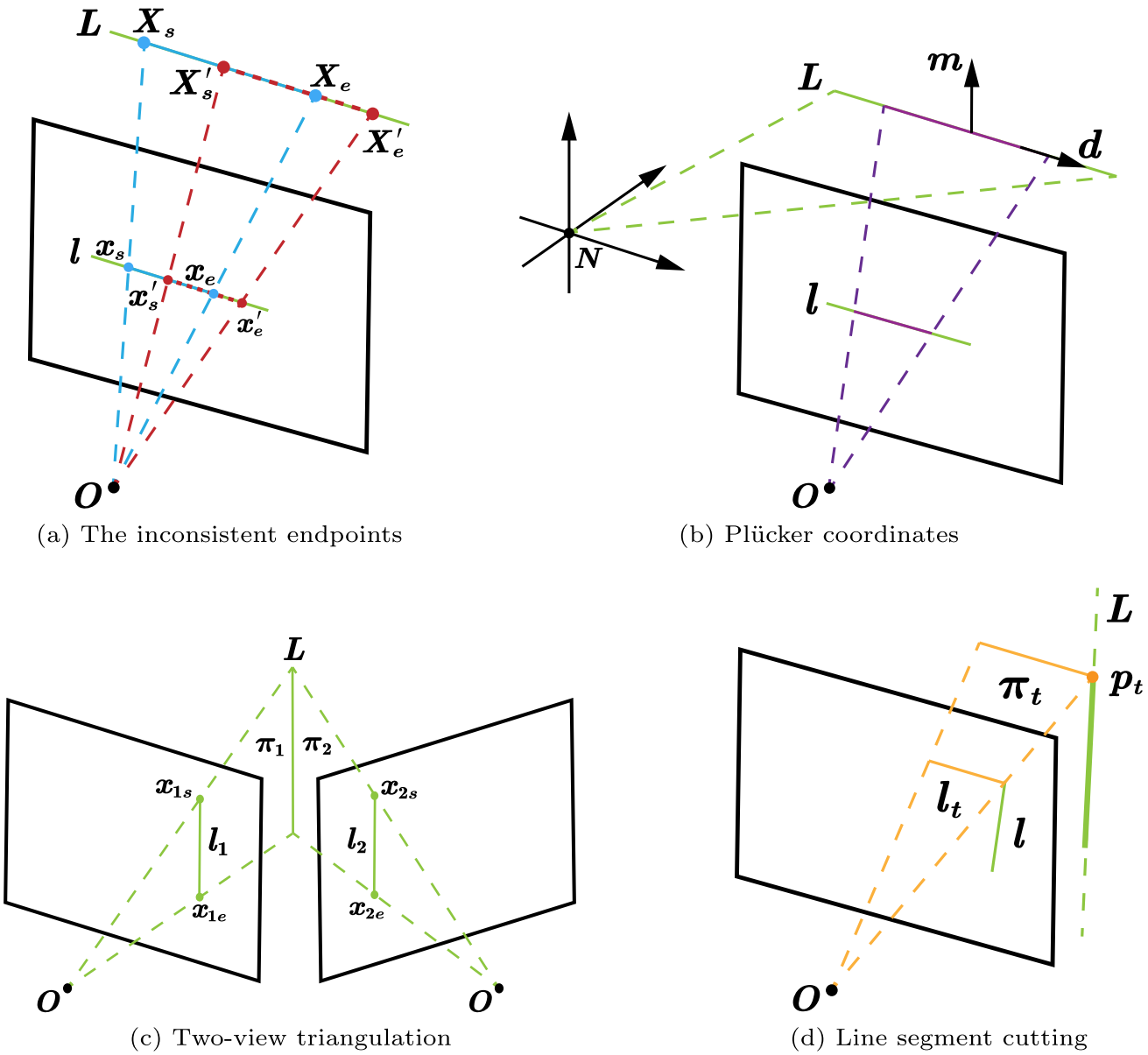
$$\pi_t = l_t^{\mathrm{T}} P \quad (36)$$

$$\bar{p}_t = L^* \pi_t \quad (37)$$

where P is the projection matrix, and p_t is the 3D trimming point. In this way, the target's 3D shape is intuitively visible, and the line projection process is much easier when directly projecting the endpoints.

We also perform outlier detection for line reconstruction by checking the depth and the reprojection error of the 3D line. If the depth of the endpoint is negative or far from the 3D point features, we delete the line segment. In addition, if the reprojection error of the midpoint of the line segment is large, we also delete the line segment.

For map culling, 3D features are compared with their nearby features via descriptors. Similar features are fused by deleting redundant features. Thus, the number of features can be kept at a reasonable size.



(a) The inconsistent endpoints (b) Plücker coordinates

(c) Two-view triangulation (d) Line segment cutting

Fig. 9 3D line reconstruction

3.2.2 Bundle Adjustment

Bundle adjustment (BA) is used to optimize both the target's pose and the location of the 3D features by minimizing the measurement error. BA considers a batch of poses T_k and features y_j selected by the covisibility graph, and the optimization problem is defined as follows:

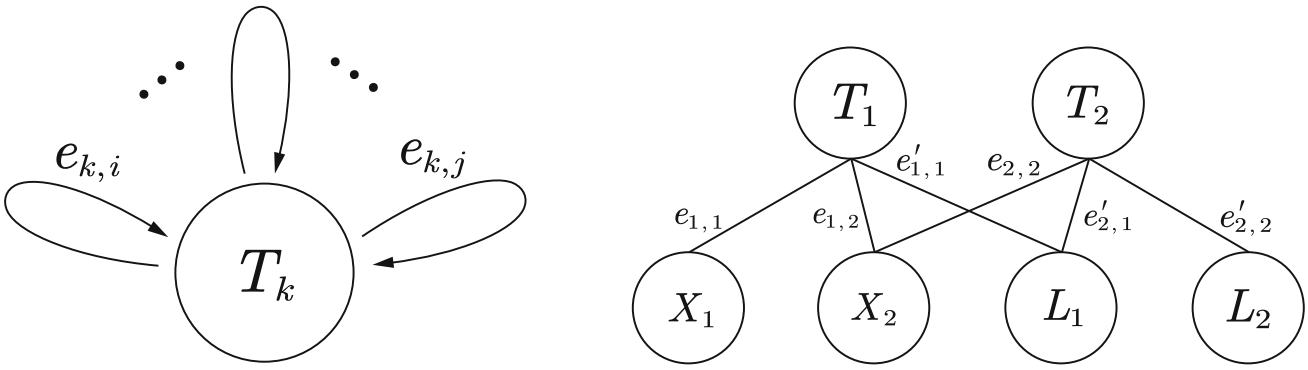
$$\min \sum_{k,j} \left\| z_{k,j} - h\left(T_k, y_j\right) \right\|^2 \tag{38}$$

According to Eq. (33), the cost function of BA is given by:

$$J = \sum_{i,j,k} \rho\left(e_{k,i}^T \Omega_{k,i}^{-1} e_{k,i} + e_{k,j}^{\prime T} \Omega_{k,j}^{\prime\ -1} e_{k,j}^{\prime} + e_{k,j}^{\prime\prime T} \Omega_{k,j}^{\prime\prime\ -1} e_{k,j}^{\prime\prime} \right) \tag{39}$$

where $\rho(\cdot)$ is .

We utilize a factor graph to model the optimization problem, shown in Fig. 10. If we only optimize the pose with fixed features, the problem is modeled as a hypergraph consisting of unary edges and one vertex. T_k is the estimated pose, and $e_{k,i}$, $e_{k,j}$ are the measurement errors. When we simultaneously optimize poses and features, vertices include poses and features, and binary edges represent the measurement errors if the poses observe the features. Graph optimization can be



(a) Pose-only optimization (b) Simultaneously optimize poses and features

Fig. 10 Graph optimization

solved by some libraries [57]. For real-time performance, we operate up to 20 keyframes for the BA.

4 Experiments

The proposed method is validated on a challenging dataset, the Satellite Hardware-In-the-loop Rendezvous Trajectories (SHIRT) dataset [58], which includes sequential images of a mockup satellite in simulated rendezvous trajectories. We use the subset of the SHIRT dataset containing 230 synthetic images from OpenGL in the ROE1 rendezvous scenario, where the chaser maintains the along-track separation while the target spins around one principal axis, shown in Fig. 11. When the light condition changes, it is hard to detect enough point features on the textured-less Tango spacecraft. However, our algorithm can still work with the help of line features.

Experiments are conducted by an Intel Core i5-10400F CPU (2.90 GHz) with 8GB of memory, and the codes are written in C++. Due to the randomness, we run each algorithm five times. We show the quantitative analysis of the estimated pose and the qualitative analysis of the estimated 3D shape. In addition, we test the running time for the real-time requirement. All trajectories are derived from the estimated pose and aligned by a similarity warp with respect to the ground truth (GT).

For comparison, the ORB-SLAM is conducted on the same dataset. The results of the pose estimation of the ORB-SLAM are shown in Fig. 12. The estimated pose is drawn in solid lines, while the GT is drawn in dashed lines. The GTs show that the target is rotating around the z-axis, causing periodic motions of the y-axis and the z-axis, and the r-angle varying from -180° to 180°. Meanwhile, the chaser is approaching the target along the x-axis, causing the distance to decrease slowly.

Until there are enough point features for initialization, the chaser continuously estimates the target's pose from the sequential images. Because the ORB-SLAM performs poorly on weak-textured targets under illumination changes, tracking of the target is frequently lost. As shown in experiments 1, 2, 3 and 4, the tracking of the target is almost lost at about 2 s to 3 s. In experiment 5, although the chaser can relocate and keep tracking with the help of relocalization, the estimated pose has significant errors. The largest offset of the translation part is 0.179 m (up to a scale), and the largest offset of the rotation part is 1.776 (unit-less). Large errors always appear around tracking loss, which is unacceptable for relative navigation. The results show that using only point features for SLAM is not robust and has low accuracy in harsh space environments.

To improve the system's performance, we add line features to the ORB-SLAM. The same configurations for ORB features are employed. The result of the pose estimation of our method is shown in Fig. 13. Compared with the ORB-SLAM, our method can effectively improve the system's robustness. Even though the tracking of the target is lost, the chaser can relocate quickly by EPnP due to accurate 3D shape reconstruction with the help of line features. Thus, large pose errors would not appear around tracking loss. Overall, the estimated pose is basically consistent with the GT. There are fluctuations of the x-axis, the pitch angle, and the yaw angle, but the errors are still small. The largest offset of the translation part decreases to 0.026 m (up to a scale), and the largest offset of the rotation part decreases to 0.343 (unit-less). The result illustrates that although fluctuations occur when point features decrease abundantly, our method can robustly track the target with the help of line features.

To further analyze the accuracy of the pose estimation, we utilize the metric of Absolute Trajectory Error (ATE) [48]. ATE is used to evaluate the overall consistency between the estimated trajectory and the GT. We consider the full trans-

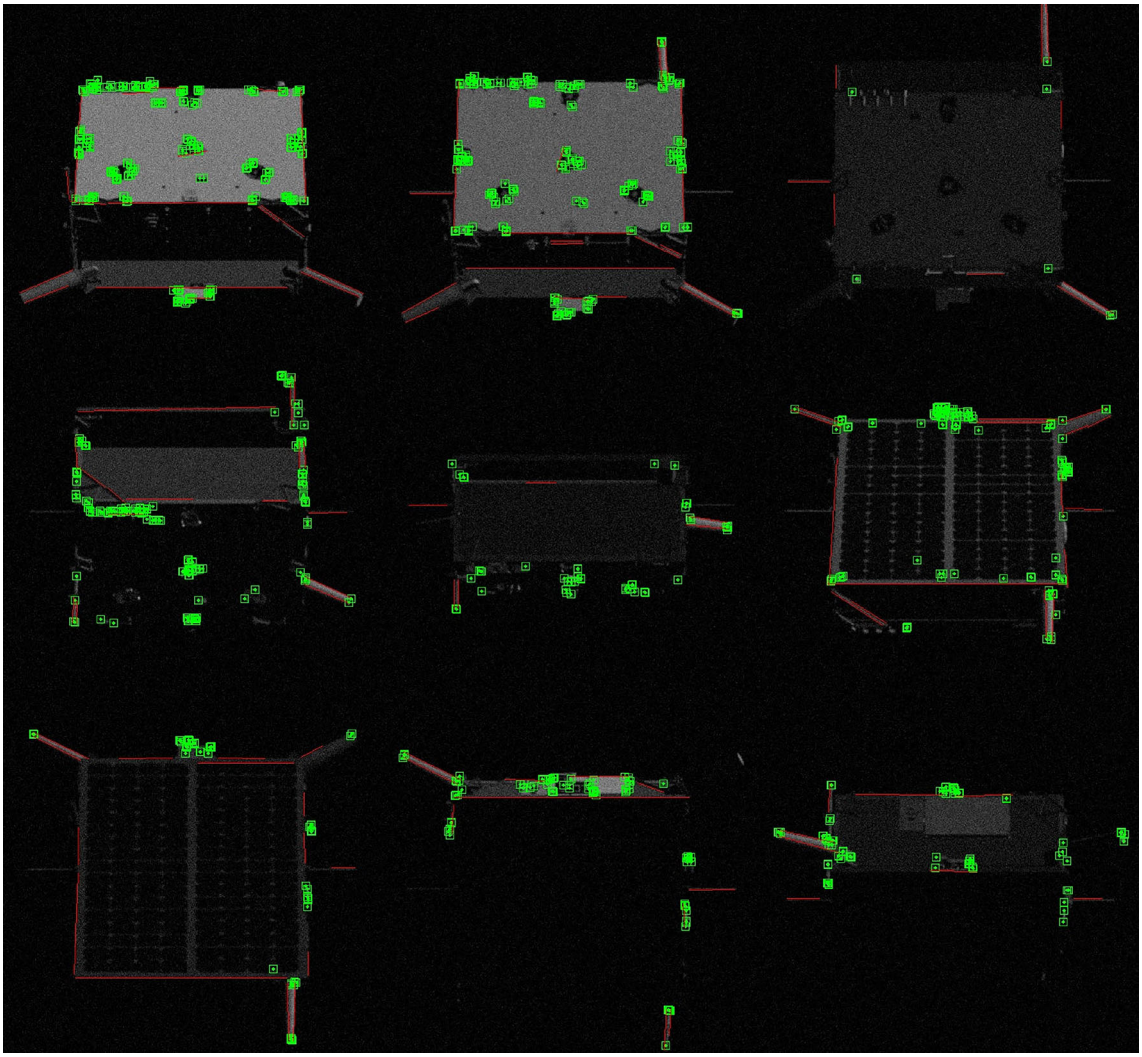
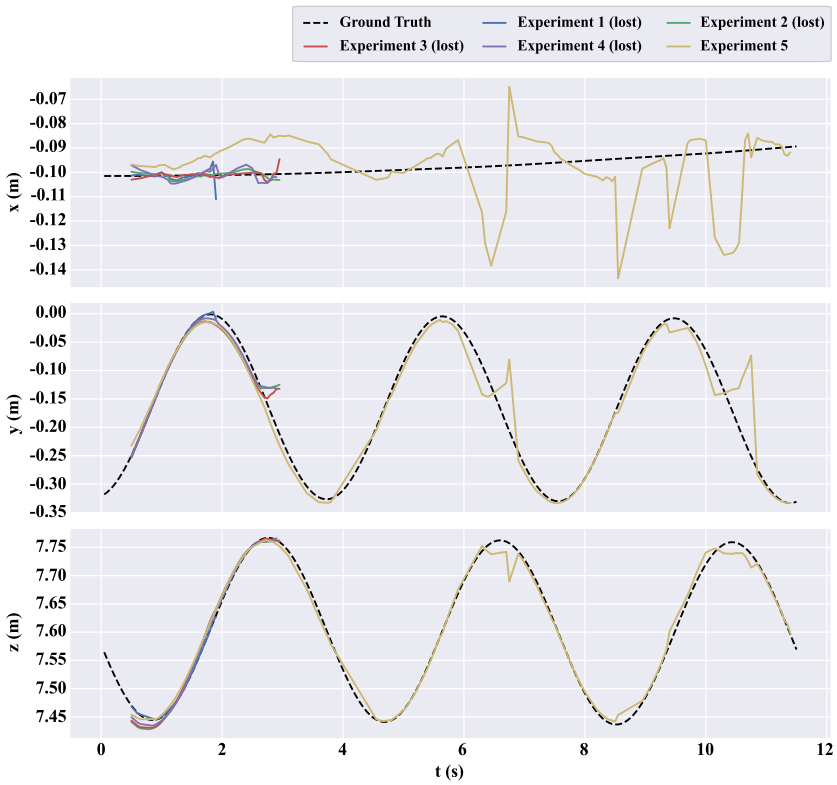


Fig. 11 The performance of feature tracking

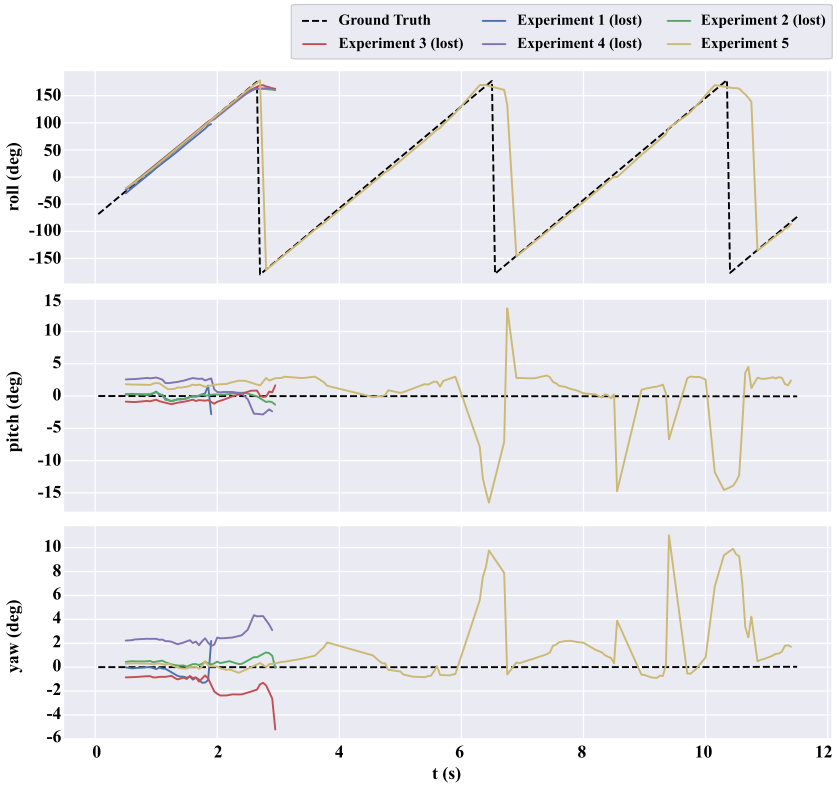
formation, including the translation part and the rotation part, to calculate the unit-less ATE. In Fig. 14, the curve of the ATE shows that the ORB-SLAM performs well at the beginning because there are enough point features. However, under the conditions of illumination changes and the weak-texture target, the number of detected point features reduces, and the estimated pose has significant errors, especially around the moments of tracking loss, which is reflected in the peaks of the ATE.

For our method, we report the ATE of the median result of the five executions, shown in Fig. 15. For quantitative analysis, we calculate the statistics of the ATE. The accuracy of the ORB-SLAM is strongly affected by tracking loss with a large maximum and standard deviation of the ATE. However, the minimum and the median of the ATE of the ORB-SLAM are smaller than our method, demonstrating that point-feature-based pose estimation has the best accuracy if there are enough point features. In another perspective, when the endpoints of lines are treated equally as points, the number of endpoints is much less than the number of point features, so point features still play a dominant role in pose estimation. However, we have a better performance considering the error distribution reflected in the root mean squared error (RMSE) of the ATE. RMSE of the ATE is always used to represent the overall accuracy. The RMSE of the ORB-SLAM is 0.374, and the RMSE of our method is 0.136. Our method improves the accuracy of the pose estimation by 63.6%.

Next, the target's 3D shape estimation is evaluated. Due to the lack of the GT of the 3D model, a qualitative analysis is given by visualization of 3D features, shown in Fig. 16. The estimated 3D shape by the ORB-SLAM in Fig. 16(a) is only with point features, making it difficult to recognize the target's structure. There are many outliers of points around the target, which may be caused by feature mismatching and map creation from larger pose errors. In comparison, our



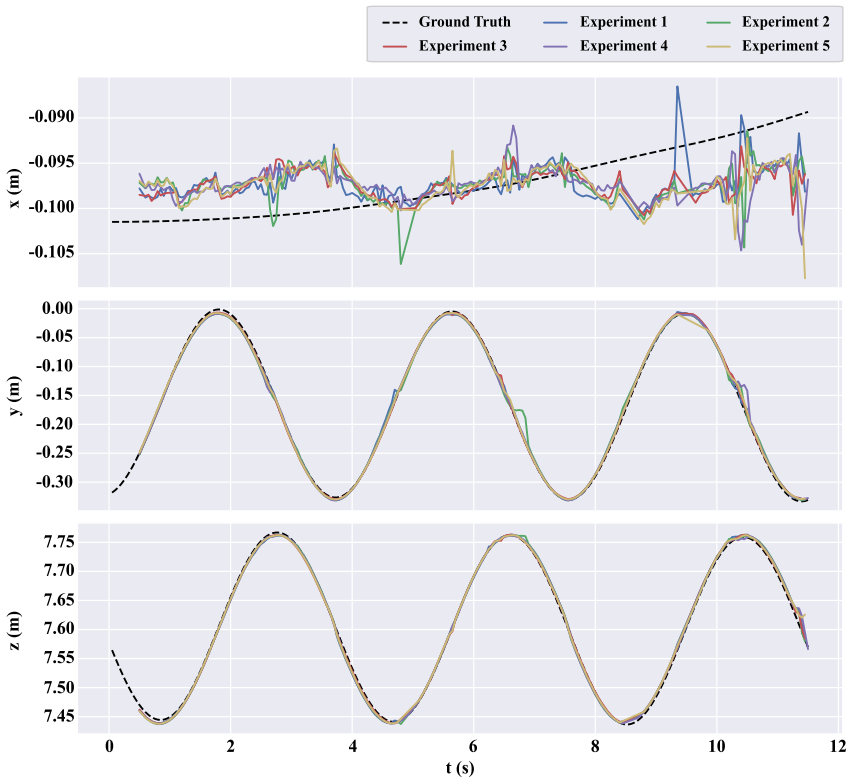
(a) Estimated translation



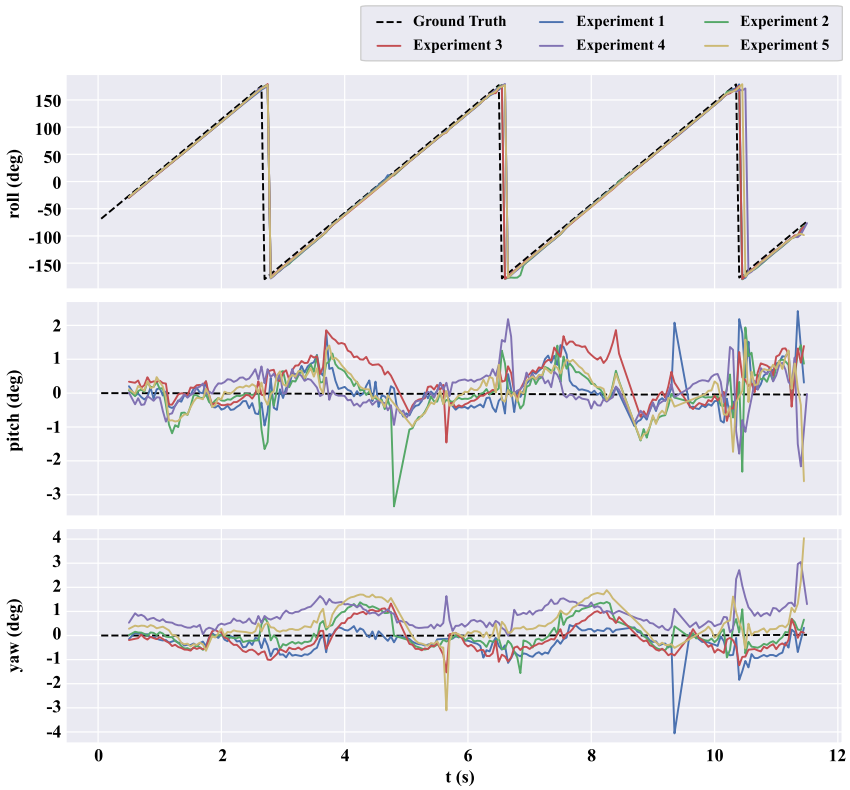
(b) Estimated rotation

Fig. 12 The results of pose estimation by the ORB-SLAM

Fig. 13 Pose estimation by our method



(a) Estimated translation



(b) Estimated rotation

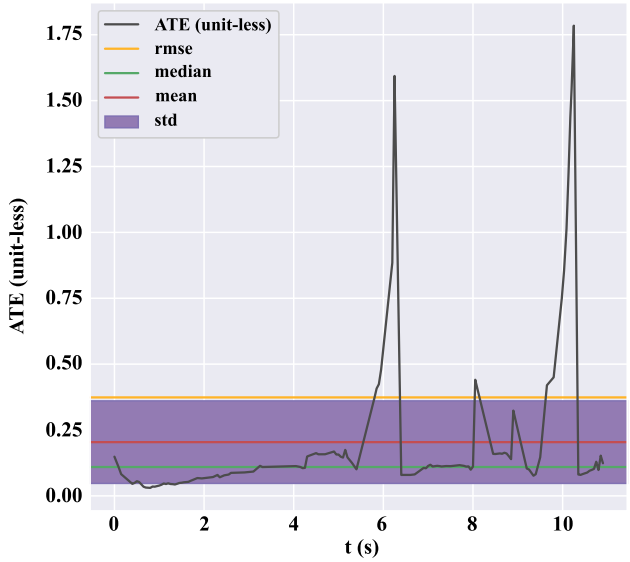


Fig. 14 ATE of the ORB-SLAM

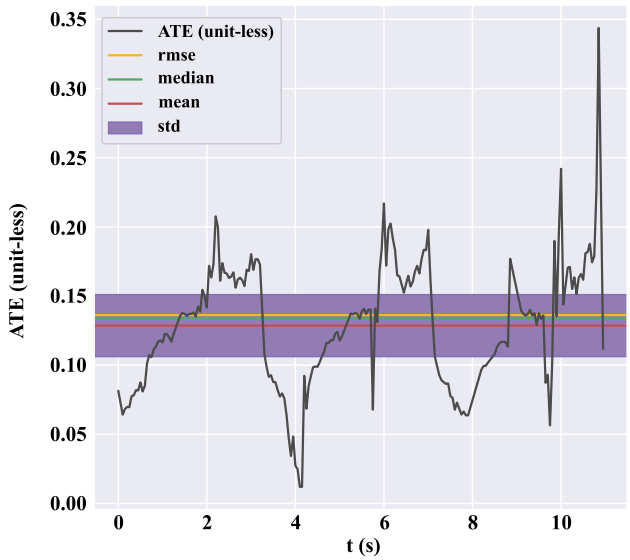


Fig. 15 ATE of our method

method in Fig. 16(b) shows that point features are distributed in the corners of the target with fewer outliers, and utilizing line features helps to gather structural information. It is clear that there is a plane on the bottom of the 3D shape, and three cylindrical antennas can be recognized, which can guide downstream on-orbit tasks. Due to the light conditions, some reconstructed 3D lines may not be detected in subsequent frames. These lines are deleted by map culling so that a few 3D lines can be preserved in the map. This proves that a few line features can improve the system's performance. In addition, although there are outliers of 3D lines triangulated by mismatching of 2D line segments, the tracking of the target is still robust with the impact on these incorrect 3D features because we reject the outliers with significant reprojection errors in the optimization process.

For the real-time requirement, the mean execution times of each operation are given in Table 1. Although adding line features can improve the system's robustness and accuracy, it also increases computational complexity. Compared with the ORB-SLAM, the running time of our method mainly increases in feature extraction and optimization, as expected. Although we utilize hidden parameter adjustment to improve the speed of feature extraction, the computational complexity is still high. In pose estimation and bundle adjustment, employing line features for optimization inevitably increases the running time. The frame rate of the ORB-SLAM is approximately 28.00 Hz, and the frame rate of our method is approximately 17.83 Hz, which is acceptable for online running (larger than 10 Hz). Our method sacrifices a certain amount of operating efficiency in exchange for significant improvements in the robustness and accuracy of the system.

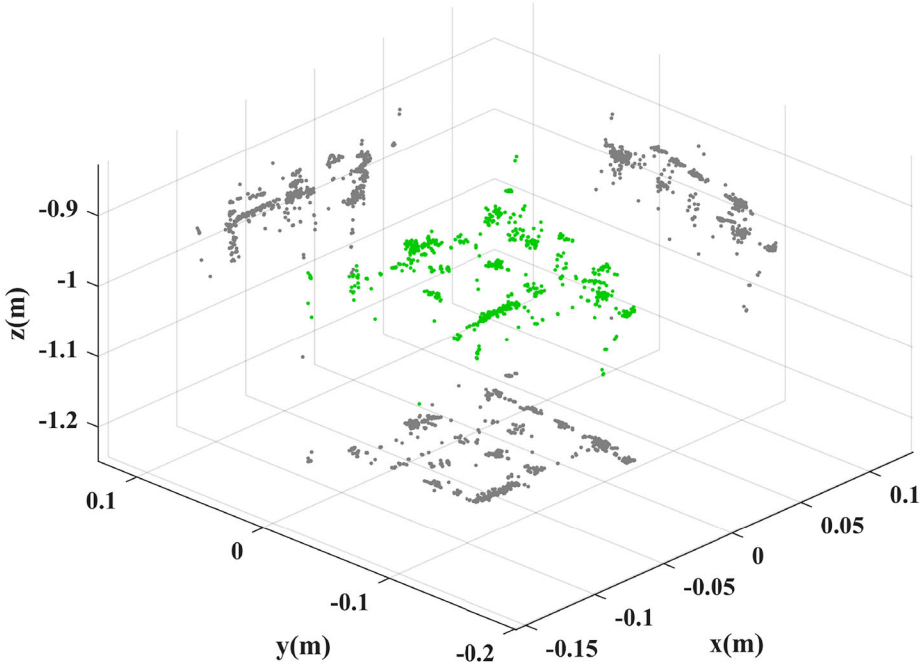
5 Conclusion

For uncooperative and unknown targets, visual SLAM shows excellent potential in solving the relative navigation problem. Dynamics-based state estimation is difficult when there is no maneuver of the chaser. Vision-based SLAM methods are powerful among the solutions for spacecraft relative navigation, which estimate relative pose (up to a scale) by simply leveraging camera images.

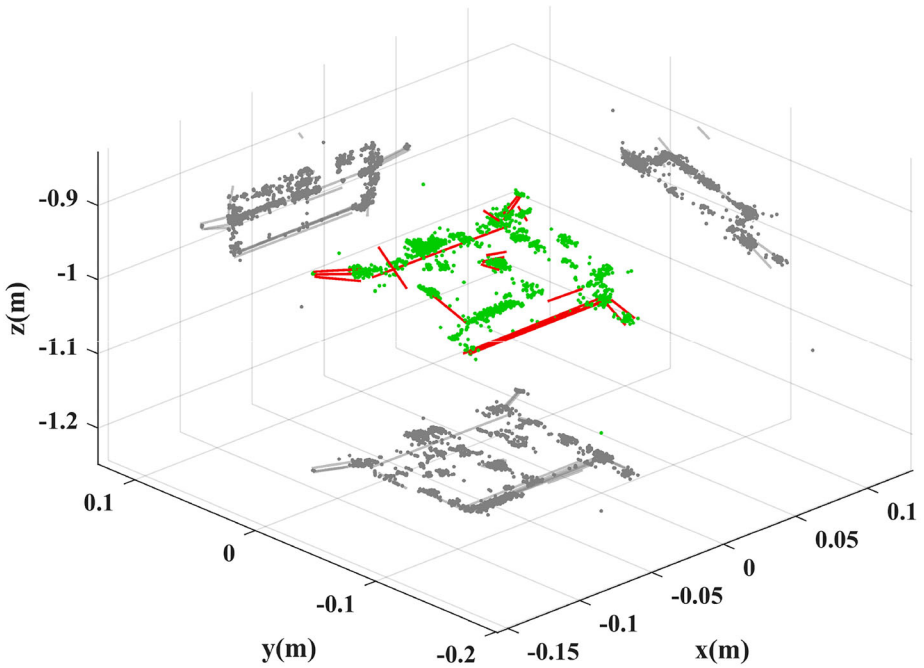
In this work, we add line features to monocular point-feature-based SLAM for spacecraft rendezvous to a weak-textured target under illumination changes. It is a real and difficult scenario in which existing research has not been carefully considered. Specifically, in the tracking thread, fast and accurate line feature extraction and matching are designed for the real-time requirement. In the reconstruction thread, the endpoint inconsistency problem is solved by Plücker coordinates. In addition, a unified cost function including point and line features are utilized for state estimation. The results show that our method strongly improves the system's robustness and pose estimation accuracy by 63.6%. The reconstruction of the 3D shape becomes better with fewer outliers and easier identification with the help of line features. Even though the real-time performance is reduced, the algorithm still guarantees real-time performance at 17.83 Hz.

We hope that this work can inspire researchers to utilize more image features for robust spacecraft relative navigation in challenging space environments. Moreover, the smoothing approach should be explored further. Even though we use a simple motion model, the smoothing approach still performs well with outlier detection. We believe that with the development of onboard computing resources, advanced

Fig. 16 3D shape estimation



(a) 3D shape estimation by the ORB-SLAM



(b) 3D shape estimation by our method

Table 1 Running times

Operations	ORB-SLAM (ms)	Ours (ms)
Feature extraction	2.16	10.06
Initialization	8.36	9.33
Pose estimation	9.23	13.83
Map creation and culling	2.21	4.08
Bundle Adjustment	13.63	18.58
Keyframes culling	0.12	0.22
Total	35.71	56.10

smoothing-based SLAM methods from the robotics community can be applied to spacecraft relative navigation.

In future work, considering the system's accuracy and real-time performance, we plan to utilize faster line extraction methods, such as CNN-based feature extraction, to reduce computational complexity and explore other line representations for optimization to evaluate the system's performance. In addition, loop detection and correction will be included to enhance the system's accuracy.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grant 52105480 and U22B2013, in part by the Xi'an Postdoctoral Innovation Base Project, and in part by the "Two Chain" Integration Key Project of Shaanxi Province under Grant 2021LLRH-03.

Data Availability The data that support the findings of this study are available on request from the corresponding author, Chenxi Wang, upon reasonable request.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest

References

1. Li W-J, Cheng D-Y, Liu X-G, Wang Y-B, Shi W-H, Tang Z-X, Gao F, Zeng F-M, Chai H-Y, Luo W-B (2019) On-orbit service (OOS) of spacecraft: A review of engineering developments. Prog Aerosp Sci 108:32–120
2. You Y, Wang H (2019) Onboard target searching strategy during lost in space situations in angles-only navigation active space debris removal. Int J Aeronaut Space Sci 20:815–829
3. Ma B, Jiang Z, Liu Y, Xie Z (2023) Advances in space robots for on-orbit servicing: A comprehensive review. Adv Intell Syst 2200397
4. Opromolla R, Fasano G, Rufino G, Grassi M (2017) A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. Prog Aerosp Sci 93:53–72
5. Cassinis LP, Fonod R, Gill E (2019) Review of the robustness and applicability of monocular pose estimation systems for relative navigation with an uncooperative spacecraft. Prog Aerosp Sci 110:100548
6. Kaplan M, Boone B, Brown R, Criss T, Tunstel E (2010) Engineering issues for all major modes of in situ space debris capture. In: AIAA Space 2010 Conference & Exposition, p. 8863
7. Tourani A, Bavle H, Sanchez-Lopez JL, Voos H (2022) Visual SLAM: what are the current trends and what to expect? Sensors 22(23):9297
8. Song J, Rondao D, Aouf N (2022) Deep learning-based spacecraft relative navigation methods: A survey. Acta Astronaut 191:22–40
9. Augenstein S, Rock S (2009) Simultaneous estimaton of target pose and 3-D shape using the FastSLAM algorithm. In: AIAA Guidance, Navigation, and Control Conference, p. 5782
10. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vision 60:91–110
11. Augenstein S, Rock SM (2011) Improved frame-to-frame pose tracking during vision-only SLAM/SFM with a tumbling target. In: 2011 IEEE International Conference on Robotics and Automation, pp. 3131–3138. IEEE
12. Sonnenburg A, Tkocz M, Janschek K (2010) EKF-SLAM based approach for spacecraft rendezvous navigation with unknown target spacecraft. IFAC Proc 43(15):339–344
13. Barbier T, Gao Y (2023) Relative visual navigation around an unknown and uncooperative spacecraft. Acta Astronaut 206:144–155
14. Schnitzer F, Janschek K, Willich G (2012) Experimental results for image-based geometrical reconstruction for spacecraft rendezvous navigation with unknown and uncooperative target spacecraft. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5040–5045. IEEE
15. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24(6):381–395
16. Capuano V, Kim K, Harvard A, Chung S-J (2020) Monocular-based pose determination of uncooperative space objects. Acta Astronaut 166:493–506
17. Na Y, Bang H, Mok S-H (2019) Vision-based relative navigation using dual quaternion for spacecraft proximity operations. Int J Aeronaut Space Sci 20:1010–1023
18. Strasdat H, Montiel J, Davison AJ (2010) Real-time monocular SLAM: Why filter? In: 2010 IEEE International Conference on Robotics and Automation, pp. 2657–2664. IEEE
19. Tweddle BE, Saenz-Otero A, Leonard JJ, Miller DW (2015) Factor graph modeling of rigid-body dynamics for localization, mapping, and parameter estimation of a spinning object in space. J Field Robot 32(6):897–933
20. Kaess M, Ranganathan A, Dellaert F (2008) iSAM: Incremental smoothing and mapping. IEEE Trans Robot 24(6):1365–1378
21. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Comput Vision Image Understand 110(3):346–359
22. Rublee E, Rabaud V, Konolige K, Bradski G (2011) ORB: An efficient alternative to SIFT or SURF. In: 2011 International Conference on Computer Vision, pp. 2564–2571. Ieee
23. Mur-Artal R, Montiel JMM, Tardos JD (2015) ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans Robot 31(5):1147–1163
24. Dor M, Tsiotras P (2018) ORB-SLAM applied to spacecraft non-cooperative rendezvous. In: 2018 Space Flight Mechanics Meeting, p. 1963
25. Naasz B, Eepoel JV, Queen S, Southward CM, Hannah J (2010) Flight results from the HST SM4 relative navigation sensor system. In: 33rd Annual AAS Guidance and Control Conference
26. Yan Z, Wang H, Ning Q, Lu Y (2022) Robust image matching based on image feature and depth information fusion. Machines 10(6):456

27. Yan Z, Wang H, Ze L, Ning Q, Lu Y (2023) A pose estimation method of space non-cooperative target based on ORBFPFH SLAM. Optik 286:171025
28. Engel J, Schöps T, Cremers D (2014) LSD-SLAM: Large-scale direct monocular SLAM. In: European Conference on Computer Vision, pp. 834–849. Springer
29. Lei T, Liu X-F, Cai G-P, Liu Y-M, Liu P (2019) Pose estimation of a noncooperative target based on monocular visual SLAM. Int J Aerosp Eng 2019:1–14
30. Meng C, Li Z, Sun H, Yuan D, Bai X, Zhou F (2018) Satellite pose estimation via single perspective circle and line. IEEE Trans Aerosp Elect Syst 54(6):3084–3095
31. Hu Q, Jiang C (2021) Relative stereovision-based navigation for noncooperative spacecraft via feature extraction. IEEE/ASME Trans Mechatr 27(5):2942–2952
32. Liu Z, Liu H, Zhu Z, Song J (2021) Relative pose estimation of uncooperative spacecraft using 2D–3D line correspondences. Appl Opt 60(22):6479–6486
33. Bechini M, Gu G, Lunghi P, Lavagna M (2024) Robust spacecraft relative pose estimation via CNN-aided line segments detection in monocular images. Acta Astronaut 215:20–43
34. Zhang X, Jiang Z, Zhang H, Wei Q (2018) Vision-based pose estimation for textureless space objects by contour points matching. IEEE Trans Aerosp Elect Syst 54(5):2342–2355
35. Liu C, Guo W, Hu W, Chen R, Liu J (2020) Real-time vision-based pose tracking of spacecraft in close range using geometric curve fitting. IEEE Trans Aerosp Elect Syst 56(6):4567–4593
36. Luckett JA. Comparison of Three Machine Vision Pose Estimation Systems Based on Corner, Line, and Ellipse Extraction for Satellite Grasping
37. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 6:679–698
38. Ballard DH (1981) Generalizing the hough transform to detect arbitrary shapes. Pattern Recogn 13(2):111–122
39. D'Amico S, Benn M, Jørgensen JL (2014) Pose estimation of an uncooperative spacecraft from actual space imagery. Int J Space Sci Eng 2(2):171–189
40. Sharma S, Ventura J, D'Amico S (2018) Robust model-based monocular pose initialization for noncooperative spacecraft rendezvous. J Spacecr Rockets 55(6):1414–1429
41. Von Gioi RG, Jakubowicz J, Morel J-M, Randall G (2008) LSD: A fast line segment detector with a false detection control. IEEE Trans Pattern Anal Mach Intell 32(4):722–732
42. Akinlar C, Topal C (2011) EDLines: A real-time line segment detector with a false detection control. Pattern Recogn Lett 32(13):1633–1642
43. Capuano V, Alimo SR, Ho AQ, Chung S-J (2019) Robust features extraction for on-board monocular-based spacecraft pose acquisition. In: AIAA Scitech 2019 Forum, p. 2005
44. Shi J (1994) Good features to track. In: 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600. IEEE
45. Gu G, Ko B, Go S, Lee S-H, Lee J, Shin M (2022) Towards lightweight and real-time line segment detection. Proceedings of the AAAI Conference on Artificial Intelligence 36:726–734
46. Zhang Z (1999) Flexible camera calibration by viewing a plane from unknown orientations. In: Proceedings of the Seventh Ieee International Conference on Computer Vision, vol. 1, pp. 666–673. Ieee
47. Fu Q, Wang J, Yu H, Ali I, Guo F, He Y, Zhang H (2020) Pl-vins: Real-time monocular visual-inertial slam with point and line features. arXiv preprint arXiv:2009.07462
48. Zhang Z, Scaramuzza D (2018) A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 7244–7251. IEEE
49. Zhang L, Koch R (2013) An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. J Vis Commun Image Represent 24(7):794–805
50. Hamid N, Khan N (2016) Lsm: perceptually accurate line segment merging. J Elect Imaging 25(6):061620
51. Hartley RI (1997) In defense of the eight-point algorithm. IEEE Trans Pattern Anal Mach Intell 19(6):580–593
52. Pumarola A, Vakhitov A, Agudo A, Sanfeliu A, Moreno-Noguer F (2017) Pl-slam: Real-time monocular visual slam with points and lines. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 4503–4508. IEEE
53. Piazza M, Maestrini M, Di Lizia P (2022) Monocular relative pose estimation pipeline for uncooperative resident space objects. J Aerosp Inf Syst 19(9):613–632
54. Lepetit V, Moreno-Noguer F, Fua P (2009) EPnP: An accurate O(n) solution to the PnP problem. Int J Comput Vis 81:155–166
55. Jia Y-B (2020) Plücker coordinates for lines in the space. Problem Solver Techniques for Applied Computer Science, Com-S-477/577 Course Handout **3**
56. Zhang G, Lee JH, Lim J, Suh IH (2015) Building a 3-d line-based map using stereo slam. IEEE Trans Robot 31(6):1364–1377
57. Jurić A, Kendeš F, Marković I, Petrović I (2021) A comparison of graph optimization approaches for pose estimation in slam. In: 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), pp. 1113–1118. IEEE
58. Park TH, D'Amico S (2023) Adaptive neural-network-based unscented kalman filter for robust pose tracking of noncooperative spacecraft. J Guid Control Dyn 46(9):1671–1688