Aerospace Systems
https://doi.org/10.1007/s42401-020-00057-8

ORIGINAL PAPER

Feature-based visual navigation integrity monitoring for urban autonomous platforms

Shizhuang Wang[1] · Xingqun Zhan[1] · Yuanwen Fu[1] · Yawei Zhai[1]

Received: 14 May 2020 / Revised: 13 July 2020 / Accepted: 29 July 2020
© Shanghai Jiao Tong University 2020

Abstract

Visual navigation systems have increasingly been adopted in many urban safety–critical applications, such as urban air mobility and highly automated vehicle, for which they must continuously provide accurate and safety-assured pose estimates. Extensive studies have focused on improving visual navigation accuracy and robustness in complex environment, while insufficient attention has been paid to ensuring navigation safety in the presence of outliers. From safety perspective, integrity is the most important navigation performance criterion because it measures the trust that can be placed in the correctness of the navigation output. Through leveraging the concept of integrity, this paper develops an integrity monitoring framework to protect visual navigation system against misleading measurements and to quantify the reliability of the navigation output. We firstly present the iterative least squares (LS)-based pose estimation algorithm and derive the associated covariance estimation methodology. Then we develop a two-layer fault detection scheme through combining random sampling consensus (RANSAC) with multiple hypotheses solution separation (MHSS) to achieve high efficiency and high reliability. Finally, the framework determines the probabilistic error bound of the navigation output that rigorously captures the undetected faults and the measurement uncertainty. The proposed algorithms are validated using various simulations, and the results suggest the promising performance.

Keywords Visual navigation · Safety · Integrity · Fault detection · Autonomous systems

1 Introduction

Vision-based or -aided navigation systems have attracted wide interest because of their outstanding performance in urban environments where Global Navigation Satellite System (GNSS) becomes seriously vulnerable [12]. Accordingly, visual navigation is an additional choice to realize the localization in urban safety–critical applications, such as urban air mobility (UAM) and highly automated vehicle (HAV), which are expected to bring great benefits to the society [1].

For these applications, ensuring the safety of visual localization is on the top priority when designing the navigation algorithms. This is because, on the one hand, visual navigation is highly sensitive to operating and environmental conditions, such as textures, presence of blurs and illumination changes. Therefore, the navigation system may perform well under some conditions, but in other environments, it might become unreliable. On the other hand, failure to correctly perform the localization task might lead to catastrophic damages to the vehicles, the passengers and people in the vicinity.

Consequently, it is highly imperative to not only improve navigation accuracy and robustness, but also to quantify the correctness of the navigation output. Up to now, prior work has mostly focused on the former task through optimizing the algorithms and developing some outlier rejection techniques [9, 11, 13]. Though these algorithms greatly reduce the failure rate, it remains possible to further improve navigation safety by performing the latter task, which is actually the objective of this study.

To this end, this paper develops a novel approach to realize autonomous integrity monitoring of visual navigation, which is rarely addressed in the literature. We leverage the integrity concept developed in aviation to assess the safety of visual navigation in the UAM and HAV deployment. Integrity measures the trust that can be placed on the correctness

✉ Xingqun Zhan
xqzhan@sjtu.edu.cn

1 School of Aeronautics and Astronautics, Shanghai Jiao Tong University, Shanghai 200240, China

of the navigation output, and integrity risk describes the probability that navigation system provides a hazardous misleading information without warning the user [15]. Obviously, integrity provides a direct way to quantify the level of safety from navigation perspective, and it is therefore employed to measure localization safety for urban autonomous systems. Furthermore, this metric and the associated analytical method perfectly complement the testing-based vehicle safety demonstration approaches [8].

Integrity monitoring is an essential approach to ensure navigation integrity, which involves two basic functions: one is to detect the faulted measurements, and the other is to quantify the safety risk. Various integrity monitoring methods have been established to ensure the integrity of GNSS-based navigation in civil aviation applications. The most representative approaches include receiver autonomous integrity monitoring (RAIM) and advanced RAIM (ARAIM), both of which are based on consistency check of redundant range measurements [4, 5].

Introducing these techniques into other navigation systems and other applications is more and more attractive, and some remarkable efforts have been made [2, 14]. However, to the best of our knowledge, only a few studies have focused on visual navigation integrity monitoring. Joerger et al. proposed an integrity monitoring method for laser-based navigation with an emphasis on feature extraction and data association [8]. In addition, Bhamidipati et al. developed a simultaneous localization and mapping (SLAM)-based integrity monitoring scheme for GNSS/vision integrated system [3]. Li et al. presented a RAIM-based integrity monitoring scheme for visual navigation by assuming that there is at most one faulted measurement (i.e., feature) after outlier rejection [10]. However, these existing approaches do not represent the general case and cannot address the challenges described as follows.

The significant difference between radio navigation and visual navigation makes it difficult to introduce the algorithms developed in the GNSS domain into visual navigation context. In urban scenarios, feature-based visual navigation generally performs better than other approaches, such as optical flow method and template matching method, because of the rich textures in the environments [1]. The number of features in an image is usually close to or even over 100, and it is much larger than the number of pseudoranges used in GNSS applications. In addition, the fault ratio could be over 10%, which is severely beyond the capability of the traditional integrity monitoring algorithms. Furthermore, the spatial dependence in measurement failures cannot be negligible in visual navigation [3], and thus the fault-independent assumption adopted by RAIM and ARAIM is not true here. Moreover, the pose estimation is usually implemented in an optimization-based approach in visual navigation while the existing integrity monitoring algorithms are mainly developed for least-squares (LS)- or filter-based navigation systems.

To account for all the challenges above, this paper develops a novel approach to realize visual navigation integrity monitoring. In this work, we firstly describe the general pipeline of feature-based visual navigation, highlighting the estimation of navigation states with matched features. Then we present the iterative LS-based pose estimation algorithm and derive the associated real-time covariance estimation methodology, based on which the integrity monitoring framework is developed. This framework includes a two-layer fault detection scheme which employs both random sampling consensus (RANSAC) and multiple hypotheses solution separation (MHSS) to achieve both high efficiency and high reliability. In this scheme, we also exploit the fault grouping technique to reduce computational complexity and to cope with spatial dependence in the faults. In addition, to assess navigation integrity, this framework determines the probabilistic error bound that captures the undetected faults and the measurement uncertainty.

The layout of this paper is arranged as follows. Section 2 describes the basic principle of visual navigation and clarifies the research scope. The pose estimation algorithm and the associated covariance estimation method are presented in Sect. 3. The details of the integrity monitoring framework are discussed in Sect. 4. In Sect. 5, we validate the proposed algorithm with simulations. Finally, the conclusions are drawn in Sect. 6.

2 Basic principle of visual navigation

Visual navigation is an accurate and robust technique to estimate the egomotion (translation and orientation of an agent) through utilizing the output from a camera or a laser sensor. Many approaches have been developed since the early 1980s [7] and they can be roughly divided into the categories shown in Table 1.

Since different approaches exhibit noticeable difference in their implementations, this paper does not intend to provide an integrity monitoring framework that covers all the approaches. For autonomous platforms operating in urban environments, feature-based approaches have been the dominant visual navigation solutions for a long time. Under urban scenarios, there are rich distinctive features that can be extracted from an image, and thus feature-based approaches show superior performance to others in terms of efficiency, accuracy, and robustness. As for the sensors, as monocular vision systems suffer from scale uncertainty, stereo cameras are preferred when there is no assistance from external sensors. Therefore, this paper mainly focuses on feature-based stereo visual navigation systems. We will expand this work to support laser-based navigation in the future.

Table 1 Categorization of typical visual navigation approaches

Classification ways	Categories	Notes
Sensor	Optical camera	Monocular, stereo
	Laser sensor	Light Detection And Ranging (LiDAR)
Measurement	Features	Using features extracted from the images
	Pixels	Using pixels or optical flow
Architecture	Visual odometry (VO)	Relative motion estimation between frames
	SLAM	Motion estimation and mapping
	Map-aided localization (MAL)	Pose estimation with pre-established map
Estimation	Optimization	Nonlinear optimization
	Filter	Kalman filter (KF), extended KF, etc
	Singular value decomposition (SVD)	Motion estimation with matched points

As shown in Table 1, feature-based visual navigation systems have three operational modes: VO, SLAM, and MAL (also called relocalization in the literature). Figure 1 presents the required procedures in these modes. VO provides an incremental online estimation of the vehicle pose by analyzing the image sequences captured by the camera. This is an efficient and convenient approach without any dependency on the prior knowledge about the environments, but the localization error will accumulate over time. SLAM is a technique that allows the vehicle localize itself in an unknown environment and incrementally generate a map. As shown in Fig. 1, VO is an essential part in SLAM while the latter also contains back-end optimization, loop closure, and mapping. These additional procedures help improve the localization accuracy, but meanwhile they make SLAM much more complicated than VO. Through employing the pre-established map, MAL allows the vehicle to estimate its pose relative to this map. In practice, the map is usually accessed from either a map provider or the past-time mapping process in SLAM.

In feature-based approaches, feature extraction is a key step in which the distinctive features (e.g., lines, curves and corners) are extracted from the images. This process can be implemented in a variety of ways with different performance [1, 7]. The proposed method is generally applicable to various features, and we take Oriented FAST and rotated BRIEF (ORB) feature as an example to validate our algorithm. As a widely used feature descriptor, ORB feature keeps invariant with rotation, translation and scale, and it also presents strong robustness to camera settings and illumination changes [13]. For the features extracted from the current image, data association attempts to find the corresponding

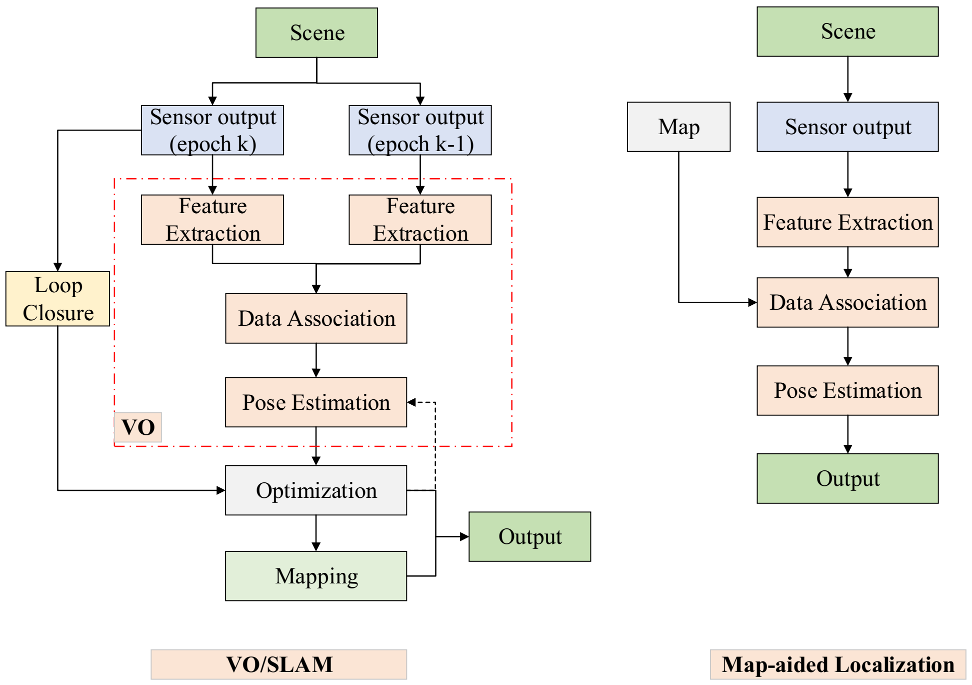

Fig. 1 Flowchart of the required steps in feature-based visual navigation systems

points in either the previous frame (VO mode) or the local map (MAL mode). Using a set of matched feature points, VO estimates the motion of the camera between epochs, while MAL computes the agent's pose relative to the map. This paper focuses on ensuring localization safety in MAL mode, but the methodology can be easily extended to VO context because of the great similarity between them.

For MAL, the direct input to the pose estimation module is the matched 3D points. In safety–critical applications, it is of great importance to pay sufficient attention to the potential faults in visual measurements (i.e., point pairs). The faults are mainly attributed to two aspects: one is incorrect data association due to unfavorable environment conditions, and the other is the changes occurring in real world scenes after map construction. These faults have the potential to result in large navigation error and could place the vehicles in hazardous situations. To ensure operation safety, the navigation system must continuously provide accurate and reliable pose estimates. To this end, this paper aims at developing an integrity monitoring framework against measurement faults so as to provide safety-assured navigation output.

This section defines the scope of this work and describes the required procedures involved in representative visual navigation approaches. Without loss of generality, this paper assumes that some distinctive features are extracted from the outputs of a stereo camera and they are matched with the points in the pre-established 3D map. For clarity, the detailed implementations of this process are not presented here and they do not influence the following derivations.

3 Pose determination and covariance estimation

3.1 Measurement equation

The objective of MAL is to determine the agent's pose, i.e., position and orientation, relative to the local map. The pose is usually defined as $\boldsymbol{x} = \left[\boldsymbol{\varphi}^T, \boldsymbol{t}^T\right]^T$, where $\boldsymbol{\varphi} = [\alpha\beta\gamma]^T$ and $\boldsymbol{t} = \left[t_x t_y t_z\right]^T$ denote the rotational (in the form of Euler angles) and translational parameters, respectively. The state vector $\boldsymbol{x}$ corresponds to a 4×4 transformation matrix $\boldsymbol{T}$ as follows:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \tag{1}$$

where the rotational matrix R is calculated with the roll (α), pitch (β) and yaw (γ) angles:

$$\boldsymbol{R}(\alpha,\beta,\gamma) = \begin{bmatrix} c\beta c\gamma & -c\alpha s\gamma + s\alpha s\beta c\gamma & s\alpha s\gamma + c\alpha s\beta c\gamma \\ c\beta s\gamma & c\alpha c\gamma + s\alpha s\beta s\gamma & -s\alpha c\gamma + c\alpha s\beta s\gamma \\ -s\beta & s\alpha c\beta & c\alpha c\beta \end{bmatrix}, \tag{2}$$

where s and c denote sine and cosine functions, respectively.

To estimate the pose, we assume that there are N pairs of matched feature points as follows:

$$\boldsymbol{P} = \left\{\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_N\right\}, \boldsymbol{Q} = \left\{\boldsymbol{q}_1, \boldsymbol{q}_2, \dots, \boldsymbol{q}_N\right\}, \tag{3}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{q}_i$ represent the 3D coordinates of the i-th feature point in the camera frame and in the navigation frame, respectively. The two frames are linked through $\boldsymbol{R}$ and $\boldsymbol{t}$, and thus we have the following measurement equation:

$$\forall i, \boldsymbol{R} \bullet \left(\boldsymbol{p}_i - \boldsymbol{\epsilon}_{pi}\right) + \boldsymbol{t} = \boldsymbol{q}_i - \boldsymbol{\epsilon}_{qi,} \tag{4}$$

$$\boldsymbol{\epsilon}_{pi} \sim \mathbb{N}\left(0, \boldsymbol{C}_{pi}\right), \tag{5}$$

$$\boldsymbol{\epsilon}_{qi} \sim \mathbb{N}\left(0, \boldsymbol{C}_{qi}\right), \tag{6}$$

where $\boldsymbol{\epsilon}_{pi}$ and $\boldsymbol{\epsilon}_{qi}$ denote the zero-mean Gaussian noise terms with covariance matrix $\boldsymbol{C}_{pi}$ and $\boldsymbol{C}_{qi}$, respectively. Note that the errors of the feature points are usually assumed to be zero-mean Gaussian noises in prior studies [10]. This work is also carried out under this assumption, and we will examine the error model in our future work.

Then we formulate the pose estimation as a nonlinear optimization. Specifically, the navigation states, $\alpha, \beta, \gamma, t_x, t_y, t_z$, are estimated by minimizing the reprojection error of features as:

$$\left\{\boldsymbol{R}^*, \boldsymbol{t}^*\right\} = \underset{\boldsymbol{R},\boldsymbol{t}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\boldsymbol{R} \bullet \boldsymbol{p}_i + \boldsymbol{t} - \boldsymbol{q}_i\|_{2.}^2 \tag{7}$$

Equation (7) is essentially a least-squares optimization problem. However, since α, β, and γ are arguments of nonlinear trigonometric functions in the rotation matrix $\boldsymbol{R}$, the linear LS estimation method cannot be directly applied to obtain the solution. In the next section, we present how to modify this optimization problem such that it can be solved via a linear LS approach.

3.2 Pose determination by iterative LS estimation

The major difficulty in linearly solving Eq. (7) is the strong nonlinearity due to the rotation matrix. Fortunately, small-angle approximation (i.e., s$\theta \approx \theta$, c$\theta \approx 1$ when $\theta \approx 0$) is an effective approach to eliminate the nonlinearity. When $\Delta\alpha$, $\Delta\beta$, $\Delta\gamma \approx 0$, we have:

$$\Delta\boldsymbol{R} \approx \begin{bmatrix} 1 & -\Delta\gamma & \Delta\beta \\ \Delta\gamma & 1 & -\Delta\alpha \\ -\Delta\beta & \Delta\alpha & 1 \end{bmatrix} \triangleq \boldsymbol{I} - (\Delta\boldsymbol{\varphi}\times), \tag{8}$$

where $(\Delta\boldsymbol{\varphi}\times)$ is the cross product matrix and $\boldsymbol{I}$ denotes the 3×3 identity matrix. With this approximation, we can linearize the measurement equation at a linearization point $\{\boldsymbol{R}_l, \boldsymbol{t}_l\}$ as:

$$\forall i, \left(\boldsymbol{p}_i|_l \times\right) \bullet \Delta\boldsymbol{\varphi} + \Delta\boldsymbol{t} = \Delta\boldsymbol{p}_i, \tag{9}$$

$$\forall i, \boldsymbol{p}_i|_l = R_l \bullet \boldsymbol{p}_i + \boldsymbol{t}_l. \tag{10}$$

Accordingly, the original nonlinear optimization problem can be solved by iterative refinement as follows. At each iteration, we linearize the measurement equations and stack them as:

$$\Delta\mathbf{y} = \boldsymbol{G} \bullet \Delta\boldsymbol{x} + \boldsymbol{\epsilon}, \tag{11}$$

where the geometry (Jacobian) matrix $\boldsymbol{G}$, the reprojection error vector $\Delta\mathbf{y}$, and the measurement noise $\boldsymbol{\epsilon}$ are given by the following:

$$\boldsymbol{G} = \begin{bmatrix} (\boldsymbol{p}_1^{'}\times) & \boldsymbol{I} \\ \vdots & \vdots \\ (\boldsymbol{p}_N^{'}\times) & \boldsymbol{I} \end{bmatrix}, \tag{12}$$

$$\Delta\mathbf{y} = \begin{bmatrix} \boldsymbol{q}_1 - \boldsymbol{p}_1^{'} \\ \vdots \\ \boldsymbol{q}_N - \boldsymbol{p}_N^{'} \end{bmatrix}, \tag{13}$$

$$\boldsymbol{p}_i^{'} = \widehat{\boldsymbol{R}}^{'} \bullet \boldsymbol{p}_i + \widehat{\boldsymbol{t}}^{'}, \tag{14}$$

$$\boldsymbol{\epsilon} = \begin{bmatrix} \widehat{\boldsymbol{R}}^{'} \bullet \boldsymbol{\epsilon}_{p1} - \boldsymbol{\epsilon}_{q1} \\ \vdots \\ \widehat{\boldsymbol{R}}^{'} \bullet \boldsymbol{\epsilon}_{pN} - \boldsymbol{\epsilon}_{qN} \end{bmatrix}, \tag{15}$$

in which $\widehat{\boldsymbol{R}}^{'}$ and $\widehat{\boldsymbol{t}}^{'}$ denote the estimated transformation parameters given by the previous iteration.

Secondly, the least-squares estimation is performed and the update for $\Delta\widehat{\boldsymbol{x}}$ is calculated by:

$$\Delta\widehat{\boldsymbol{x}} = \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}\right)^{-1}\boldsymbol{G}^{\mathrm{T}} \bullet \Delta\mathbf{y} \tag{16}$$

Then we update the transformation matrix with $\Delta\widehat{\boldsymbol{x}}$ as:

$$\begin{bmatrix} \widehat{\boldsymbol{R}} & \widehat{\boldsymbol{t}} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \Delta\widehat{\boldsymbol{R}} & \Delta\widehat{\boldsymbol{t}} \\ 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \widehat{\boldsymbol{R}}^{'} & \widehat{\boldsymbol{t}}^{'} \\ 0 & 1 \end{bmatrix} \tag{17}$$

Finally, we output the estimated pose $\widehat{\boldsymbol{x}}$ when the solution has converged, and the last $\Delta\mathbf{y}$ is defined as the residual vector $\mathbf{y}$.

3.3 Covariance calculation

In navigation context, covariance matrix is often used as a measure of the uncertainty in the state estimates and covariance estimation lays a foundation for accuracy evaluation and integrity monitoring. The pose uncertainty is caused by the measurement errors, and the key of covariance estimation is to derive the error propagation formula. According to Eq. (15), the covariance matrix $\boldsymbol{C}_i$ of the measurement noise corresponding to the i-th feature is computed as:

$$\boldsymbol{C}_i = \widehat{\boldsymbol{R}} \bullet \boldsymbol{C}_{pi} \bullet \widehat{\boldsymbol{R}}^{\mathrm{T}} + \boldsymbol{C}_{qi} \tag{18}$$

Assuming that the measurement errors of different features are uncorrelated, the stacked covariance matrix for all the features is a block diagonal matrix as follows:

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_1 & & \\ & \ddots & \\ & & \boldsymbol{C}_N \end{bmatrix} \tag{19}$$

Based on Eq. (16), we can directly determine the state perturbation $\delta\boldsymbol{x}$ as follows:

$$\delta\boldsymbol{x} = \left[\delta\boldsymbol{\varphi}^{\mathrm{T}}, \delta\boldsymbol{t}^{\mathrm{T}}\right]^{\mathrm{T}} = \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}\right)^{-1}\boldsymbol{G}^{\mathrm{T}} \bullet \boldsymbol{\epsilon} \tag{20}$$

The covariance matrix of $\delta\boldsymbol{x}$ is given by:

$$\boldsymbol{C}_\delta = \boldsymbol{S}_\delta \bullet \boldsymbol{C} \bullet \boldsymbol{S}_{\delta,}^{\mathrm{T}} \tag{21}$$

where $\boldsymbol{S}_\delta = \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}\right)^{-1}\boldsymbol{G}^{\mathrm{T}}$.

However, $\boldsymbol{C}_\delta$ is not the error covariance matrix of the pose estimates because the pose error $\boldsymbol{\varepsilon}_x = \widehat{\boldsymbol{x}} - \boldsymbol{x}$ is not equal to the state perturbation $\delta\boldsymbol{x}$. The relationship between $\widehat{\boldsymbol{x}}$ and $\delta\boldsymbol{x}$ is established through the transformation matrix:

$$\begin{bmatrix} \widehat{\boldsymbol{R}} & \widehat{\boldsymbol{t}} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \delta\boldsymbol{R} & \delta\boldsymbol{t} \\ 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}. \tag{22}$$

This equation can be further rewritten as:

$$\begin{bmatrix} \widehat{\boldsymbol{R}} & \widehat{\boldsymbol{t}} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \delta\boldsymbol{R} \bullet \boldsymbol{R} & \delta\boldsymbol{R} \bullet \boldsymbol{t} + \delta\boldsymbol{t} \\ 0 & 1 \end{bmatrix}. \tag{23}$$

Then we have:

$$\boldsymbol{\varepsilon}_\varphi = \widehat{\boldsymbol{\varphi}} - \boldsymbol{\varphi} = \begin{bmatrix} c\gamma/c\beta & s\gamma/c\beta & 0 \\ -s\gamma & c\gamma & 0 \\ c\gamma s\beta/c\beta & s\gamma s\beta/c\beta & 1 \end{bmatrix} \cdot \delta\boldsymbol{\varphi} \cdot \boldsymbol{A}_\varphi \cdot \delta\boldsymbol{x} \tag{24}$$

$$\boldsymbol{\varepsilon}_t = \widehat{\boldsymbol{t}} - \boldsymbol{t} = (\boldsymbol{t}\times) \bullet \delta\boldsymbol{\varphi} + \delta\boldsymbol{t} \triangleq \boldsymbol{A}_t \bullet \delta\boldsymbol{x}. \tag{25}$$

The detailed proof of Eq. (24) is given in "Appendix".

Accordingly, the covariance matrices of $\boldsymbol{\varepsilon}_\varphi$ and $\boldsymbol{\varepsilon}_t$ are, respectively, given by:

$$\boldsymbol{C}_\varphi = \boldsymbol{A}_\varphi \bullet \boldsymbol{C}_\delta \bullet \boldsymbol{A}_\varphi^{\mathrm{T}}, \tag{26}$$

$$\boldsymbol{C}_t = \boldsymbol{A}_t \bullet \boldsymbol{C}_\delta \bullet \boldsymbol{A}_t^{\mathrm{T}}. \tag{27}$$

4 Integrity monitoring framework for visual navigation

4.1 Input parameters

As mentioned earlier, feature-based visual navigation systems will occasionally encounter measurement faults which greatly threaten the operation safety. This paper leverages the concept of integrity into visual navigation context and develops an integrity monitoring framework correspondingly. This framework can not only cope with the misleading features, but also determine the probabilistic error bound of the pose estimate. The framework requires some essential inputs to ensure its reliability, including measurement error models, prior fault probabilities, and the predefined navigation performance requirements.

Table 2 presents the error models and the prior fault probabilities of each feature. The error models stem from a preliminary analysis on measurement errors based on KITTI dataset which provides ground truth and outputs from a stereo camera. In addition, it should be noted that the fault probability here represents the probability of an "inlier" being faulted after the first-layer outlier rejection (i.e., RANSAC). The two-layer fault detection scheme will be illustrated in detail in Sect. 4.2. Since the prior fault probability may vary with the settings in RANSAC, this work also carries out sensitivity analysis over this parameter with the values given in Table 2.

The navigation requirement parameters mainly consist of the target integrity risk and the continuity risk coming from false alert. Integrity risk is defined as the probability that the navigation system provides hazardous misleading information (HMI) without warning the user. False alert is an event occurring when the fault detector indicates a fault state in a fault-free condition and it is a major cause of the loss of navigation continuity. Table 3 shows the preliminary values of these parameters, which are representative of typical requirements in safety–critical applications. Since the requirements are actually application dependent, we will determine the navigation requirements for specific vehicular applications in future work.

4.2 Two-layer fault detection scheme design

For vehicular applications in urban environment, visual navigation system may work in a situation where there is a large number of measurements and the fault ratio is considerably high. This scenario is beyond the capability of traditional integrity monitoring schemes developed in GNSS applications. In response, we develop a two-layer fault detection scheme which combines the advantages of RANSAC and MHSS to improve detection efficiency and ensure navigation integrity.

Before the description of this scheme, we first clarify the basic assumptions. We assume that quality control is performed in the map generation process. To be more specific, the non-static objects are a big threat to the visual navigation system, and we assume that most of these unwanted objects are excluded from the map. This exclusion step can be realized by the map provider using ground truth data and some object classification

Table 2 Error models and prior fault probabilities

Parameter	Description	Value (Preliminary)
$\boldsymbol{C}_{pi}$	Error covariance matrix for $\boldsymbol{p}_i$	diag([0.5^2, 0.5^2, 1^2]), unit: m^2
$\boldsymbol{C}_{qi}$	Error covariance matrix for $\boldsymbol{q}_i$	diag([0.5^2, 0.5^2, 1^2]), unit: m^2
p_{fi}	Prior fault probability for feature i	$\{10^{-3}, 10^{-4}, 10^{-5}\}$

Note: diag() returns a square diagonal matrix with the elements inside the brackets

Table 3 List of navigation performance requirements

Parameter	Description	Value (preliminary)
P_{HMI}	Total integrity budget	6×10^{-7}
$P_{\mathrm{HMI},R}$	Integrity budget for each rotational component	10^{-7}
$P_{\mathrm{HMI},T}$	Integrity budget for each translational component	10^{-7}
P_{THRES}	Threshold for integrity risk from unmonitored faults	10^{-8}
P_{FA}	Continuity budget allocated to false alarm	6×10^{-6}
$P_{\mathrm{FA},R}$	False alarm probability for each rotational component	10^{-6}
$P_{\mathrm{FA},T}$	False alarm probability for each translational component	10^{-6}

techniques. Detailed implementation of this step is beyond the scope of this paper. Under this assumption, it is very unlikely that there are significantly many outliers with strong consistency.

In the first layer, RANSAC is employed to exclude the outliers from raw visual measurements. RANSAC achieves its goal by repeating the following steps [9]:

1. Randomly choose a subset of the feature correspondences as the initial consensus set.
2. Estimate the navigation parameters using this subset.
3. All other features are then tested against the estimate and those points that fit the estimated parameters well (comparing with a predefined threshold) are added to the consensus set.

This procedure is repeated a fixed number of times and the consensus set with the most inliers is output as the inlier set. RANSAC is an effective approach to remove most of the outliers even though the outlier ratio is high, and thus it is widely used in vision-based navigation. Under the aforementioned assumption, RANSAC will not be misled by the outliers and can reduce the outlier rate to a sufficiently low value. The number of iterations and the threshold for outlier identification greatly impact the fault probability after RANSAC, i.e., the probability of an inlier being faulted. Therefore, we need to properly determine the fault probability by specifying the RANSAC settings. This is beyond the scope of this paper but involved in another work of us.

The fault probability is expected to be significantly low (e.g., 10^{-3}, 10^{-4} or 10^{-5}) with appropriate RANSAC settings. However, we cannot guarantee the safety of visual navigation system by only using RANSAC. This is because, on the one hand, the low-probability events still need to be monitored in safety–critical applications. On the other hand, though the navigation errors can be greatly reduced after performing RANSAC, it is very difficult to derive the associated probabilistic error bound as required by integrity monitoring algorithm.

MHSS realizes the detection of both single and multiple faults through establishing a list of subsets, each of which corresponds to a fault mode. It should be noted that there is an obvious difference between the process of subset determination in RANSAC and in MHSS. RANSAC intends to find a very small subset of feature points so that all of them are inliers. In MHSS, each subset contains most of the observations because it is assumed that there are only a few faulted features after performing RANSAC.

When utilizing MHSS, one can determine the subsets that need to be monitored based on the prior fault probability of each feature. The associated probability of each fault mode (or subset) is determined by multiplying the fault probabilities of the features that are assumed faulted in this mode. The fault modes with relatively low probability are not monitored, and we use a predefined threshold P_{THRES} to bound the integrity risk coming from these unmonitored modes. In this process, we can also obtain the maximum number $N_{f,\max}$ of simultaneous faults that need to be monitored. One can implement the subset determination following the detailed procedures shown in the ARAIM baseline algorithm [4], by regarding each feature as a "satellite" and setting the constellation fault probability to 0.The outputs of this step include: the monitored subsets (indexed by $j = 0,1,\dots,N_s$), $N_{f,\max}$, the fault probability $p_{\mathrm{fs}}^{(j)}$ of each subset, and the total integrity risk p_{nm} coming from the unmonitored modes.

Then we calculate the test statistics and the corresponding thresholds. For subset j, the difference $\Delta\boldsymbol{x}^{(j)}$ between the fault-tolerant solution $\boldsymbol{x}^{(j)}$ and the all-in-view solution $\boldsymbol{x}^{(0)}$ is given by:

$$\Delta\boldsymbol{x}^{(j)} = \boldsymbol{x}^{(j)} - \boldsymbol{x}^{(0)} = \left(\boldsymbol{S}^{(j)} - \boldsymbol{S}^{(0)}\right) \cdot \boldsymbol{y}, \tag{28}$$

where the coefficient matrix $\boldsymbol{S}^{(j)}$ is calculated by the following:

$$\boldsymbol{S}^{(j)} = \begin{bmatrix} \boldsymbol{A}_\varphi \\ \boldsymbol{A}_t \end{bmatrix} \left(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{W}^{(j)}\boldsymbol{G}\right)^{-1}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{W}^{(j)}, \tag{29}$$

in which the $3n \times 3n$ diagonal matrix $\boldsymbol{W}^{(j)}$ is obtained by:

$$\boldsymbol{W}^{(j)}(3i-2:3i, 3i-2:3i) = \begin{cases} 0, & \text{when feature } i \text{ is assumed faulted in subset } j \\ 1, & \text{otherwise} \end{cases} \tag{30}$$

As a result, we employ the MHSS technique as the second layer to further detect the faults, and more importantly, to make it convenient for quantifying navigation safety. MHSS has been widely implemented in integrity monitoring algorithms because it offers two significant advantages: it is effective in coping with the multiple simultaneous faults, and it provides a straightforward proof of integrity. The step-by-step procedures of MHSS are illustrated in detail as follows.

Let the index $q = 1, 2$ and 3 designate three attitude components, and $q = 4, 5$ and 6 represent the translational components. The variances of $\boldsymbol{x}^{(j)}$ and $\Delta\boldsymbol{x}^{(j)}$ are, respectively, given by:

$$\sigma_q^{(j)2} = \operatorname{cov}\left(x_q^{(j)}, x_q^{(j)}\right) = \boldsymbol{e}_q^{\mathrm{T}}\boldsymbol{S}^{(j)}\boldsymbol{C}\boldsymbol{S}^{(j)\mathrm{T}}\boldsymbol{e}_q, \tag{31}$$

$$\sigma_{ss,q}^{(j)2} = \mathrm{cov}\left(\Delta x_q^{(j)}, \Delta x_q^{(j)}\right) = \boldsymbol{e}_q^T\left(\boldsymbol{S}^{(j)} - \boldsymbol{S}^{(0)}\right)\boldsymbol{C}\left(\boldsymbol{S}^{(j)} - \boldsymbol{S}^{(0)}\right)^{\mathrm{T}}\boldsymbol{e}_q, \tag{32}$$

in which $\boldsymbol{e}_q$ is a vector whose q-th entry is 1 and all others are 0.

For each fault mode, there are six solution separation tests, one for each rotational or translational component. The corresponding thresholds are calculated by the following:

$$T_q^{(j)} = K_{fa,q} \cdot \sigma_{ss,q}^{(j)} \tag{33}$$

where,

$$K_{fa,q} = \begin{cases} Q^{-1}\left(\frac{P_{FA,R}}{2N_s}\right), & q = 1,2,3 \\ Q^{-1}\left(\frac{P_{FA,T}}{2N_s}\right), & q = 4,5,6 \end{cases} \tag{34}$$

$Q^{-1}(p)$ is the $(1-p)$ quantile of a zero-mean unit-variance Gaussian distribution. The coefficients $K_{fa,q}$ reflect the probability of false alarm. The navigation system is declared to pass the second-layer fault detection only if for all j and q we have the following:

$$\tau_q^{(j)} = \frac{\left|x_q^{(j)} - x_q^{(0)}\right|}{T_q^{(j)}} \le 1. \tag{35}$$

If any of the tests fails, the navigation system will lose its continuity. To improve navigation continuity under fault conditions, fault exclusion should be attempted. We do not provide an exclusion scheme here, but we will develop an accurate and efficient exclusion algorithm in future work.

For real-time applications, it is imperative to take the algorithm complexity into consideration. From this perspective, the scheme above may be computationally expensive due to the large number of measurements. To address this issue, we introduce a technique, fault grouping, which can greatly reduce the number of monitored subsets. The basic idea of this technique is to divide the image into several non-overlapping zones, called superpixels, each of which includes a group of features. Figure 2 illustrates the generation of superpixels by using a 2D example.

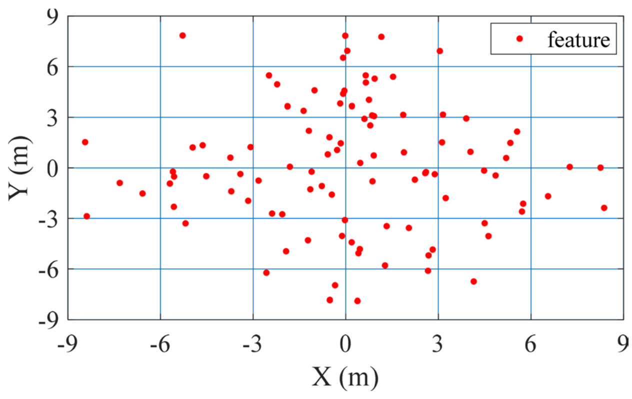

Fig. 2 Illustration of image segmentation in a 2D case (each rectangle is an individual superpixel)

After introducing the fault grouping technique, we can determine the monitored subsets following the same procedure as that in the traditional MHSS method. But we need to regard each superpixel as an individual "feature" in this case. The prior fault probability of the superpixel, which represents the probability of any feature in it being faulted, is given by the following:

$$p_{sp} = 1 - \left(1 - p_f\right)^{n_p}, \tag{36}$$

where p_f and p_{sp} denote the prior fault probability of each feature and each superpixel, respectively; and n_p is the number of features in this superpixel.

4.3 Protection level calculation

Protection level (PL) is a probabilistic error bound computed so as to guarantee that the probability of the pose error exceeding the said number is smaller than or equal to the target integrity risk [15]. In this paper, the PL is defined separately for each pose component. The protection level $\mathrm{PL_q}$ for the q-th component can be determined by solving the following equation:

$$P_{HMI,q}\left(1 - \frac{p_{nm}}{P_{HMI}}\right) = 2Q\left(\frac{PL_q}{\sigma_q^{(0)}}\right) + \sum_{j=1}^{N_s} p_{fs}^{(j)} Q\left(\frac{PL_q - T_q^{(j)}}{\sigma_q^{(j)}}\right), \tag{37}$$

where,

$$P_{HMI,q} = \begin{cases} P_{HMI,R}, & q = 1,2,3 \\ P_{HMI,T}, & q = 4,5,6 \end{cases} \tag{38}$$

The proof of safety associated with this protection level and the method to solve Eq. (37) are shown in ARAIM baseline algorithm [4]. In this equation, each term of the right-hand side is an upper bound of the contribution of each fault mode to the integrity risk and the left-hand term is the target integrity risk allocated to the monitored subsets.

5 Experiments and results

In this section, several simulations are carried out to validate the proposed algorithms. To represent the feature geometry in true urban environments, we extract the ORB features from an image provided by KITTI dataset [6]. The depth of each feature is also calculated by comparing the outputs of the left and right cameras. Figure 3 shows the geometry of

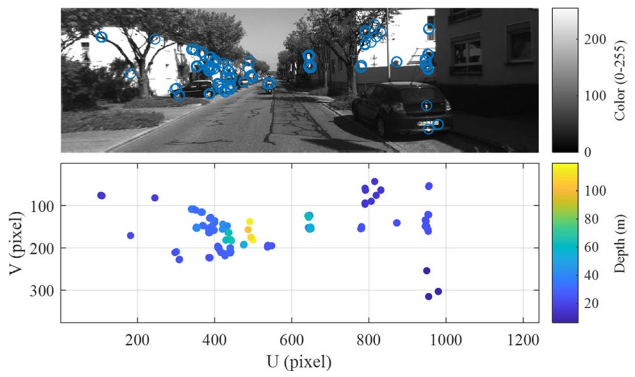

Fig. 3 Feature points in an image and their spatial distribution

the extracted features in the camera frame. We assume that this image comes from a vehicle running in a well-mapped environment and the feature correspondences between this image and the map have been determined. To simulate noisy measurements, we add white noises to the feature points according to the measurement error covariances shown in Table 2. Because the proposed pose estimation and integrity monitoring schemes are both "snapshot" methods, i.e., the results only rely on the current observations, the static scenario shown in Fig. 3 is sufficient for the performance evaluation.

Based on the simulated feature correspondences, we firstly validate the proposed pose determination algorithm and the associated covariance estimation methodology. To capture the covariance variation over vehicle's pose, the ten scenarios shown in Table 4 are used to perform this analysis. The parameters in this table represent the poses of the vehicle relative to the local map. For each scenario, we conduct Monte Carlo simulations to generate 5000 random scenarios which are used to statistically determine the error standard deviations. Figure 4 presents the statistical and estimated error standard deviations. The results suggest that the navigation state parameters can be estimated with satisfactory accuracy by using the proposed algorithm, and they also prove the feasibility of the covariance estimation methodology. This figure shows that the pose uncertainty is obviously sensitive to the vehicle's attitude while keeps invariant over translational parameters.

Then we conduct another simulation to reveal the impacts of fault grouping on both the computational complexity and the navigation performance. In this simulation, we assume that the features shown in Fig. 3 constitute the inlier set generated by RANSAC. The fault probability of each feature is set to different values for sensitivity analysis. In addition, fault grouping is performed through dividing the 3D image into several cuboids whose size greatly impacts the grouping results. Table 5 shows the comparison of CPU time

Table 4 List of the cases used in the simulations

Case no	1	2	3	4	5	6	7	8	9	10
Roll (α)/rad	0	0	$\pi/6$	$\pi/3$	0	0	$\pi/3$	$\pi/6$	$\pi/6$	$\pi/6$
Pitch (β)/rad	0	0	0	0	$\pi/6$	$\pi/3$	$\pi/3$	$\pi/6$	$\pi/6$	$\pi/6$
Yaw (γ)/rad	$\pi/6$	$\pi/3$	0	0	0	0	$\pi/3$	$\pi/6$	$\pi/6$	$\pi/6$
X (t_x)/m	0	0	0	0	0	0	0	0	0	5
Y (t_y)/m	0	0	0	0	0	0	0	0	0	5
Z (t_z)/m	5	5	5	5	5	5	5	5	10	10

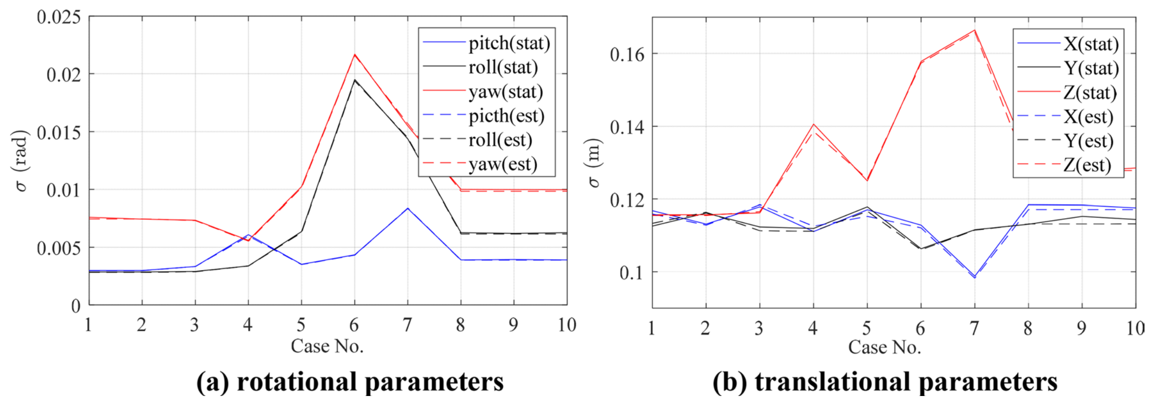

Fig. 4 The statistical (solid lines) and estimated (dashed lines) error standard deviations

Table 5 Computational complexity comparison among different grouping results

Case no	p_{fi}	Number of superpixels	Number of subsets	Time consuming
1	10^{-3}	152 (no grouping)	> 20,000,000	Very long
2	10^{-3}	40 (small cuboid)	561,001	777.10 s
3	10^{-3}	17 (big cuboid)	4352	6.18 s
4	10^{-4}	152	577,654	784.39 s
5	10^{-4}	40	8336	11.20 s
6	10^{-4}	17	514	0.88 s
7	10^{-5}	152	11,535	16.60 s
8	10^{-5}	40	735	1.23 s
9	10^{-5}	17	127	0.34 s

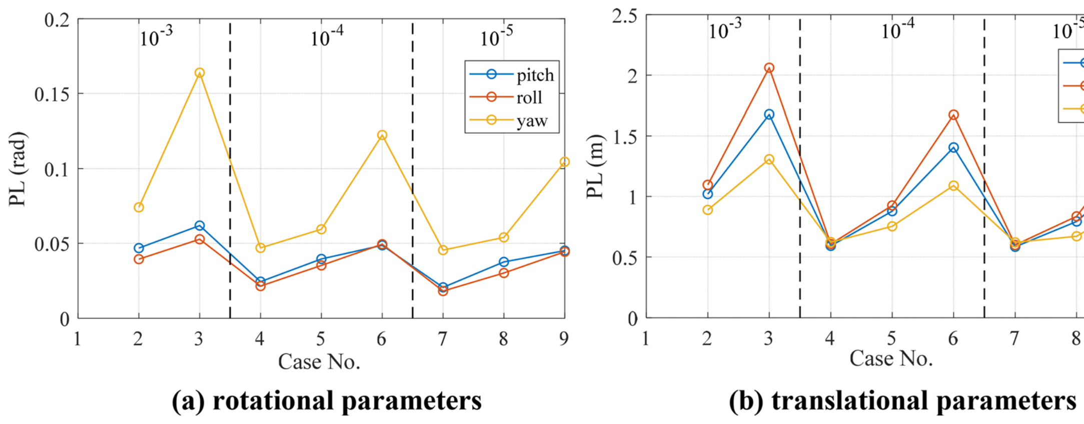

Fig. 5 The variation in protection levels over fault grouping results and prior fault probabilities

among different grouping results, and Fig. 5 presents the associated protection levels. The simulation is run on the MATLAB 2018b software installed on a laptop with Intel Core i5-8300H and 8 GB memory. Note that we do not calculate the protection levels for case 1 because this process consumes a lot of time.

According to the results shown in Table 5 and Fig. 5, we can draw the following conclusions. First, the integrity monitoring framework cannot be applied to real-time applications without employing fault grouping. Second, fault grouping can greatly decrease the time consumption through reducing the number of subsets while at an expense of the increase in protection levels. Thirdly, the sensitivity analysis over fault probability suggests that high fault probability will lead to great number of subsets and large protection levels. Therefore, it is imperative to perform the first layer fault detection because it can effectively reduce the fault probabilities of the features used in the second layer detection.

Finally, we validate the proposed fault detection scheme through simulating two fault scenarios. As shown in Fig. 3, some features (labeled as superpixel A) are extracted from the motorcycle in the center of the image, and some other features (labeled as superpixel B) locate at the tree in the left of the image. In the first scenario, we manually inject a 10-m fault to the depth component of superpixel A. This represents a case where there is a non-static landmark in the map. The second one is a multi-fault scenario, which is simulated by adding a 5-m fault to the depth component of superpixel B in addition to the fault in superpixel A. The fault in superpixel B might be caused by the incorrect feature matching. The parameters adopted in Monte Carlo simulations are shown in Table 6.

Table 6 The parameters used in the simulations

Parameters	Values
(α, β, γ)	[0, 0, 0] rad
(t_x, t_y, t_z)	[0, 0, 6] m
p_{fi}	10^{-5}
Number of superpixels	17
Measurement errors	See Table 2
Simulation times	1000

Figure 6 shows the position errors in Z direction (usually the direction along the street) under nominal and

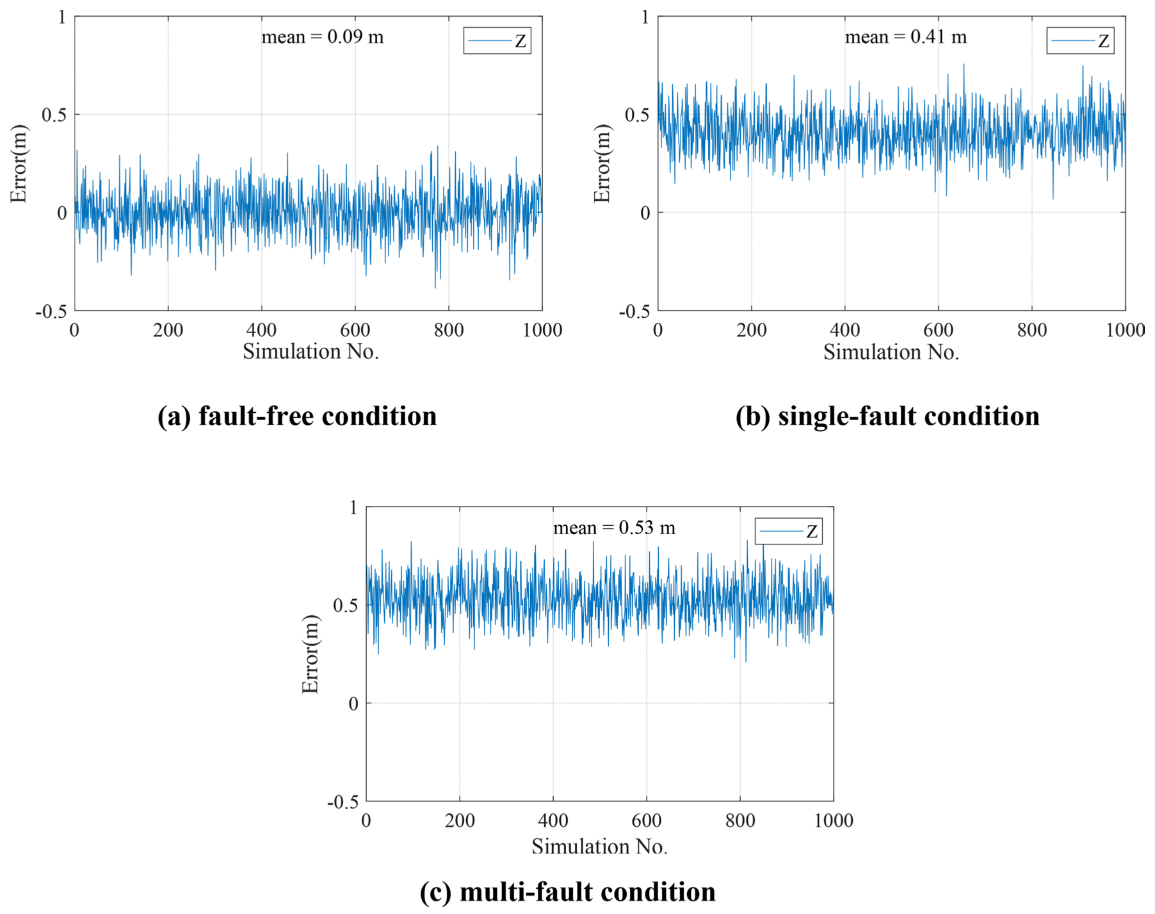

Fig. 6 Position errors in Z direction under fault-free and faulted conditions

faulted conditions. It is obvious that compared to fault-free case, the position errors are significantly larger in the presence of visual faults. Because large navigation errors may lead to serious accidents, it is necessary to perform fault detection in safety–critical applications. Figure 7 presents the results of fault detection under the three scenarios. The fault detector will issue an alarm when the statistics exceed the thresholds. As shown in this figure, there is no missed detection or false alarm occurring in this experiment. Therefore, the results prove the promising performance of the fault detection scheme under both single-fault and multi-fault conditions.

6 Conclusions

This paper develops an integrity monitoring framework to ensure the safety of feature-based visual navigation. In this framework, we design a two-layer fault detection scheme to cope with misleading features and rigorously derive the protection levels to quantify navigation safety. To support real-time applications, we also leverage the fault grouping technique to reduce the computational complexity of this framework. Several Monte Carlo simulations are carried out to validate the proposed algorithms. Firstly, the results prove the feasibility of the least squares (LS)-based pose determination algorithm and the associated covariance estimation method. In addition, a key trade-off between computational complexity and navigation integrity is pointed out: fault grouping can greatly reduce time consumption of the algorithm while at the expense of the increase in protection levels. We also reveal that high prior fault probability can simultaneously lead to large computational burden and poor navigation performance. Finally, simulation results suggest the promising performance of the proposed fault detection algorithm. Our future work will focus on validating the proposed framework with open-source datasets, investigating an efficient fault exclusion algorithm, and optimizing the two-layer fault detection scheme.

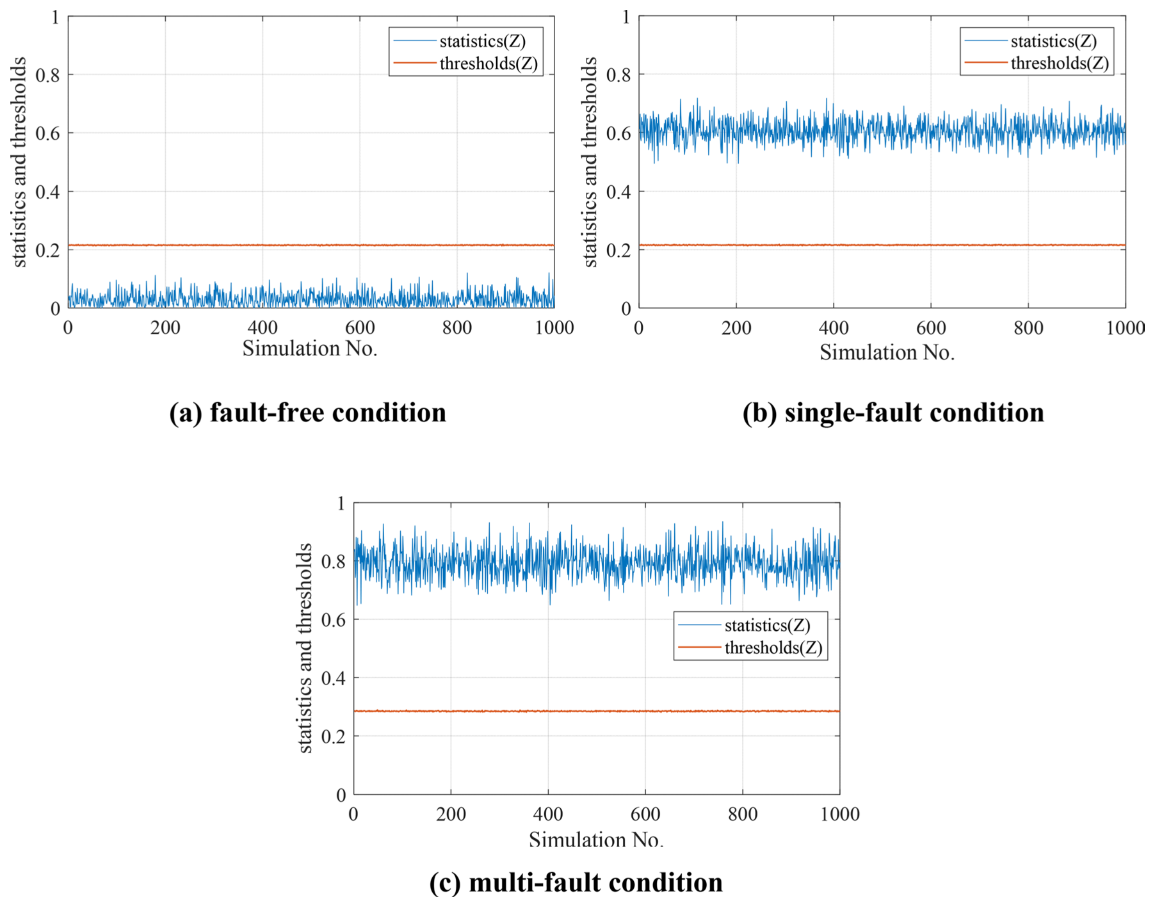

Fig. 7 Test statistics and the associated thresholds under fault-free and faulted conditions

Acknowledgements This study was supported by Shanghai Jiao Tong University (SJTU) Global Strategic Partnership Fund (2019 SJTU–UoT).

Appendix: Proof of Eq. (24)

$\delta\boldsymbol{\varphi}$ is usually called misalignment angles, while $\boldsymbol{\varepsilon}_\varphi$ denotes the errors in attitude angles. The relationship between them is given by the following:

$$\delta\boldsymbol{\varphi} = \boldsymbol{C}_b^n \cdot \boldsymbol{C}_A^\omega \cdot \boldsymbol{\varepsilon}_\varphi, \tag{39}$$

where $\boldsymbol{C}_b^n = \boldsymbol{R}^{\mathrm{T}}$ and $\boldsymbol{C}_A^\omega$ is shown as:

$$\boldsymbol{C}_A^\omega = \begin{bmatrix} 1 & 0 & -s\beta \\ 0 & c\alpha & s\alpha c\beta \\ 0 & -s\alpha & c\alpha c\beta \end{bmatrix}. \tag{40}$$

This equation is obtained from the attitude kinematic equation. Substituting (40) to (39),

$$\delta\boldsymbol{\varphi} = \begin{bmatrix} c\beta c\gamma & -s\gamma & 0 \\ c\beta s\gamma & c\gamma & 0 \\ -s\beta & 0 & 1 \end{bmatrix} \cdot \boldsymbol{\varepsilon}_\varphi. \tag{41}$$

Therefore, we have:

$$\boldsymbol{\varepsilon}_\varphi = \begin{bmatrix} c\gamma/c\beta & s\gamma/c\beta & 0 \\ -s\gamma & c\gamma & 0 \\ c\gamma s\beta/c\beta & s\gamma s\beta/c\beta & 1 \end{bmatrix} \cdot \delta\boldsymbol{\varphi} \tag{42}$$

References

1. Aqel MOA, Marhaban MH, Saripan MI, Ismail NB (2016) Review of visual odometry: types, approaches, challenges, and applications. SpringerPlus 5(1897):1–26
2. Arana GD, Joerger M, Spenko M (2019) Efficient Integrity Monitoring for KF-based Localization. In: International Conference on Robotics and Automation (ICRA), vol 2019, pp 6374–6380
3. Bhamidipati, S., & Gao, G. X. (2019). SLAM-based integrity monitoring using GPS and fish-eye camera, pp 1–14. arXiv:1910.02165

4. Blanch J, Walker T, Enge P et al (2015) Baseline advanced RAIM user algorithm and possible improvements. IEEE Trans Aerosp Electron Syst 51(1):713–732
5. Brown R (1992) A baseline GPS RAIM scheme and a note on the equivalence of three RAIM methods. Navigation 39(3):301–316
6. Geiger A, Lenz P, Stiller C, Urtasun R (2013) Vision meets robotics: the KITTI dataset. Int J Robot Res 32(11):1231–1237
7. Huang, B., Zhao, J., & Liu, J. (2019). A survey of simultaneous localization and mapping with an envision in 6G wireless networks, pp 1–17. arXiv:1909.05214
8. Joerger M, Pervan B (2019) Quantifying safety of laser-based navigation. IEEE Trans Aerosp Electron Syst 55(1):273–288
9. Kitt B, Geiger A, Lategahn H (2010) Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. In: Proceedings of IEEE Intelligent Vehicles Symposium, pp 486–492
10. Li C, Waslander SL (2019) Visual measurement integrity monitoring for UAV localization. In: Proceedings of 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp 22–29
11. Li Z, Wang J (2015) Comparison of multiple fault detection methods for monocular visual navigation with 3D maps. In: 2014 Ubiquitous positioning indoor navigation and location based service, UPINLBS 2014—Conference Proceedings, pp 228–237
12. Mostafa M, Zahran S, Moussa A, El-Sheimy N, Sesay A (2018) Radar and visual odometry integrated system aided navigation for UAVS in GNSS denied environment. Sensors (Switzerland) 18(9):1–29
13. Mur-Artal R, Tardos JD (2017) ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. IEEE Trans Robot 33(5):1255–1262
14. Tanıl Ç, Khanafseh S, Joerger M, Kujur B, Kruger B, De Groot L, Pervan B (2019) Optimal INS/GNSS coupling for autonomous car positioning integrity. In: Proceedings of the 32nd International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS+ vol 2019, pp 3123–3140
15. Zhu N, Marais J, Betaille D, Berbineau M (2018) GNSS position integrity in urban environments: a review of literature. IEEE Trans Intell Transp Syst 19(9):2762–2778