Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Multi-UAV collaborative system with a feature fast matching algorithm

Tian-miao WANG[1], Yi-cheng ZHANG[†‡1], Jian-hong LIANG[1], Yang CHEN[2], Chao-lei WANG[3]

[1]*School of Mechanical Engineering & Automation, Beihang University, Beijing 100191, China*

[2]*School of Physics and Mechatronics Engineering, Longyan University, Longyan 364000, China*

[3]*State Key Laboratory of Intelligent Manufacturing System Technology, Beijing Institute of Electronic System Engineering, Beijing 100040, China*

[†]E-mail: zycet@126.com

Received Jan. 27, 2020; Revision accepted Apr. 7, 2020; Crosschecked Apr. 22, 2020; Published online June 4, 2020

Abstract: We present a real-time monocular simultaneous localization and mapping (SLAM) system with a new distributed structure for multi-UAV collaboration tasks. The system is different from other general SLAM systems in two aspects: First, it does not aim to build a global map, but to estimate the latest relative position between nearby vehicles; Second, there is no centralized structure in the proposed system, and each vehicle owns an individual metric map and an ego-motion estimator to obtain the relative position between its own map and the neighboring vehicles'. To realize the above characteristics in real time, we demonstrate an innovative feature description and matching algorithm to avoid catastrophic expansion of feature point matching workload due to the increased number of UAVs. Based on the hash and principal component analysis, the matching time complexity of this algorithm can be reduced from $O(\log N)$ to $O(1)$. To evaluate the performance, the algorithm is verified on the acknowledged multi-view stereo benchmark dataset, and excellent results are obtained. Finally, through the simulation and real flight experiments, this improved SLAM system with the proposed algorithm is validated.

Key words: Multiple UAVs; Collaboration; Simultaneous localization and mapping (SLAM); Feature description and matching
https://doi.org/10.1631/FITEE.2000047 **CLC number:** TP242.6; V279

1 Introduction

1.1 Motivation

Unmanned aerial vehicles (UAVs) play an increasingly important role in different fields, such as environment inspection, security surveillance, rescue, and even military investigation. Due to the size and weight restrictions, a single UAV faces great limitations when performing practical missions. Thus, the collaboration of multiple UAVs is increasingly demanded. If multiple UAVs can collaborate well with each other, then the carrying capacity, detection range, and reliability can be improved significantly. The collaboration of multiple UAVs brings challenges from the perspectives of coordination and configuration. Under most circumstances, UAV collaboration relies on the Global Navigation Satellite System (GNSS). Nevertheless, for UAVs and the application environment, there are several obvious limitations and disadvantages. For example, the accuracy is low if there is no additional GNSS base station, and GNSS signals are external inputs (which can be easily interfered) and are not available indoors. Alternative methods include the infrared motion track system (e.g., the Vicon motion capture system), the ultra-wide band localization system, and lidar with simultaneous localization and mapping (SLAM). The former two methods are applicable only in indoor/semi-indoor environments equipped with facilities for

‡ Corresponding author
ORCID: Yi-cheng ZHANG, https://orcid.org/0000-0003-3643-9384

target localization. These facilities are expensive and require accurate calibration and synchronization. Specifically, lidars are quite expensive and heavy, and are not UAV-friendly. Hence, suitable methods that can remove the aforementioned restrictions are needed.

1.2 Related works

Instead of using the complex and expensive sensors mentioned above, researchers started to use cameras in UAV visual navigation studies. In 1999, Carnegie Mellon University (CMU) began to study micro-vehicle vision and proposed the concept of visual odometry (VO) (Amidi et al., 1999). Brigham Young University (BYU) fused an inertial navigation system (INS) and visual information, modified the INS with visual observation, improved the accuracy of location estimation for UAV (Andersen and Taylor, 2007), and tried to establish VO assisted by inertial navigation (Ready and Taylor, 2007, 2009). Different configurations were investigated by researchers at Stanford University, who adopted an eye-to-hand multi-camera system instead of the eye-in-hand configuration (Masayoshi et al., 2017). An array of extrinsically calibrated cameras was used to realize non-Global Positioning System (non-GPS) localization navigation for UAVs (Eberli et al., 2011). Technical University of Madrid (UPM) estimated the flight altitude through stereo vision, using the fuzzy-control method to realize autonomous flight of vehicles (Meingast et al., 2004; Templeton et al., 2007). In addition, through the micro-helicopter three-dimensional (3D) SLAM technique, Mondragón et al. (2010a, 2010b) collected information in autonomous flight, and extracted and located feature points in the environment to establish the environment map. University of Seville exploited the monocular vision and image montage technique and combined the images collected from visual navigation to form a big map (Caballero et al., 2007; Campoy et al., 2009). When the vehicle flew over the map again, Caballero et al. (2006, 2009) used the map through a Kalman filter (KF) to eliminate the navigation error and updated the map at the same time. There is a lot of data to be stored in this algorithm, and the computing workload is huge. The General Robotics Automation Sensing and Perception (GRASP) Laboratory started to study a multi-rotor flight in 1998. They applied the Vicon infrared visual motion caption system for UAV navigation. Their high dynamics control and indoor formation flight took a leading position in the world. Distance measurement sensors (e.g., ultrasonic rangefinder, laser rangefinder, and binocular vision) were used in many SLAM studies (Fox et al., 2000; Rocha et al., 2005; Howard et al., 2006; Mourikis et al., 2009). Vidal-Calleja et al. (2011) described a collaboration system with ground mobile robots and vehicles. McDonald et al. (2011) proposed a visual SLAM system that performs consecutive localization based on the map built earlier. ORB-SLAM is a real-time monocular SLAM system (Mur-Artal et al., 2015). It uses the ORB feature as its core, supports monocular, stereo, and RGB-D cameras, and has an integrated VO, tracking, and loop closure detection function. With its complete structure and high complexity, it is currently the most popular SLAM framework. However, it does not support inertial data fusion, which imposes great limitations on the dynamic performance of the system. Moreover, it does not support multiple UAVs. Although there is no multi-robot system application, it still provides a sound basis for the multi-robot system. Cunningham et al. (2013) constructed a multi-robot distribution SLAM system. Forster et al. (2013) built a complete multi-UAV SLAM system based on the binary feature descriptor binary robust invariant scalable keypoints (BRISKs). They executed real flight experiments and realized real-time performance. However, this system employs a global map, which results in significant coupling.

The extraction of image feature information is a key step in SLAM. Scale-invariant feature transform (SIFT) proposed by Lowe (2004) is an excellent algorithm for extraction and description of the image feature. This algorithm is far better than the Harris corner detection algorithm in terms of robustness in feature point exaction and feature description. The speeded up robust feature (SURF) algorithm (Bay et al., 2006) is an improvement on the SIFT algorithm. Using the integral image can reduce the workloads in feature point extraction. In addition, SURF resizes the filter instead of the image. Compared with the SIFT algorithm, the SURF algorithm reduces the workloads greatly and retains its major advantages, including invariable translation, rich description information, and insensitivity to illumination. A variant of the SIFT algorithm based on principal component analysis (PCA) was proposed by Ke and Sukthankar

(2004). It runs large-scale data compression on the SIFT feature vector, which decreases data redundancy and ensures consistency and reliability of the feature vector.

1.3 Contributions and outlines

Based on the above analysis and comparison with traditional methods such as GNSS-based solutions, vision-based navigation systems solve the problems related to cost, weight, and uncertainties of the environments. However, building a complete real-time monocular SLAM system is still quite challenging, and is even more difficult in the case of multiple UAVs. A multi-UAV monocular SLAM system generally imposes more constraints on computing efficiency, real-time performance, robustness, and cost.

Note that many matching operations exist in the SLAM system, especially the one with multiple UAVs. To avoid catastrophic expansion of computing workload due to the increased numbers of UAVs and map feature points, in this study, we propose an innovative feature point description and matching algorithm named "fast matching feature" (FMF). Based on the SURF feature vectors and the principal ideas of hash and PCA, FMF reduces the matching time complexity for large-scale feature points from $O(\log N)$ to $O(1)$, which is suitable for the field containing massive matching operations. In addition, it has been verified on the acknowledged multi-view stereo (MVS) benchmark dataset (Simo-Serra et al., 2015; Yi et al., 2016). Although the matching accuracy is reduced compared with SURF, the matching efficiency has been improved, allowing FMF to sufficiently meet the requirements and achieve the expectations.

Based on FMF, we describe a real-time monocular SLAM system with a new distributed structure for a multi-UAV collaboration task. The system aims to estimate the latest relative position between nearby vehicles without using the traditional global map, bundle adjustment, or loop closure detection. Each vehicle has its individual map and filter instead of merging the maps. Using the relationship between maps enables us to estimate the relative position of nearby vehicles. With this design, even if several vehicles fail, the functions of other vehicles are still available. Finally, through simulation and real flight experiments, this improved SLAM framework with the proposed algorithm is validated, making it a highly efficient and real-time multi-UAV relative-position estimation SLAM system.

2 Fast matching feature

The application of the SLAM technique to UAV collaboration presents challenges that rarely happen in normal application scenarios (e.g., a monocular case). The challenges consist of two aspects:

1. The data scale is proportional to the number of vehicles. As a result, the data scale is larger than that of typical SLAM applications.

2. Real-time computation is of extreme importance. The latency of calculation will influence the collaboration function, and thus a highly efficient data process is required.

Feature vector extraction and matching is the key computing procedure in SLAM. It is used in loop closure detection and ego-motion estimation. In math, the matching can be described as follows: In a high-dimensional space Ω, we search for the closest point to a given point $q\in\Omega$ in the point set $S\in\Omega$. The error can be represented by the Euclidean distance. To solve the matching problem caused by a large number of points, exhaustion whose computing complexity is $O(N)$ can be used. Apart from that, there are several common optimization methods such as K-dimensional (K-D) tree, random K-D forest, and local sensitive hashing (LSH). The tree-like methods can efficiently process the low-dimensional space, but they cannot deal with the high-dimensional space. The hash-like method is superior in terms of computing efficiency, but the establishment of hash mapping functions and the mismatch created by the lost information remain great challenges.

Therefore, we propose a hash-like feature extraction and matching algorithm. In this algorithm, the feature vector is produced by the SURF algorithm and compressed by the PCA algorithm. The information redundancy is reduced and the dimension disaster is avoided. Through generating multiple hash values, the mismatch probability can be lowered.

2.1 Evaluation standard and dataset

One of the core advantages of the feature point description algorithm is consistency; i.e., images of the same point taken at different angles and directions

in diverse illuminations with various distances can generate the same or similar feature description vectors. Another advantage is discrimination; i.e., images of different points can generate vectors with a distance as far as possible. In addition, we hope that the information provided by the descriptors is as concise as possible on the premise of being sufficient for comparison.

To analyze and compare different feature description algorithms, researchers generally use the MVS dataset as the evaluation benchmark. The MVS dataset consists of three subsets, i.e., Liberty (LY), NotreDame (ND), and Yosemite (YO), containing about 380 000 image patches with different angles, illumination conditions, and scales. Each image patch is 64×64 pixels, labeled with 3DPatchID. Image patches originating from the same point are labeled with the same 3DPatchID. There are multiple pre-generated match tables, with the known numbers of matching and non-matching image pairs attached to the dataset. To evaluate the feature description algorithm proposed in this study, this MVS dataset and matching table are used as the evaluation benchmarks. Unless otherwise stated, the matching table "m50_10000_10000_0.txt" in the LY subset is employed in the text all along. This matching table contains 10 000 image pairs. Half of the pairs are corresponding image patches and the other half pairs are non-corresponding ones.

2.2 Algorithm description

Considering the computing efficiency, the FMF algorithm adopts the SURF algorithm as the basis. To analyze the information expression and redundancy of SURF features, the SURF algorithm is used to calculate the feature vector for each image patch in the MVS dataset. The overall mean value of SURF feature vectors in the MVS dataset is 0.0546, and the standard deviation is 0.1126.

Fig. 1 describes the element-by-element mean value and the distribution of SURF feature vectors. Fig. 2 depicts the element-by-element standard deviation of SURF feature vectors. It can be seen from Figs. 1 and 2 that the distribution characteristics of the vector elements are inconsistent. Hence, the SURF feature description algorithm does not make full use of the overall vector space.

To further analyze information redundancy, the PCA algorithm is applied. Because the SURF feature vectors are 64-dimensional, they are mapped to a 64-dimensional vector space by the PCA algorithm. Fig. 3 shows the variance ratio for each vector element after PCA linear mapping. It can be concluded that the amount of information contributed by the last 40 dimensions is extremely low, less than 1×10^{-14}%. The total amount of information contributed by the former 20 dimensions exceeds 99.999 653%. Therefore, we can conclude that the SURF feature description algorithm has a lot of information redundancy.

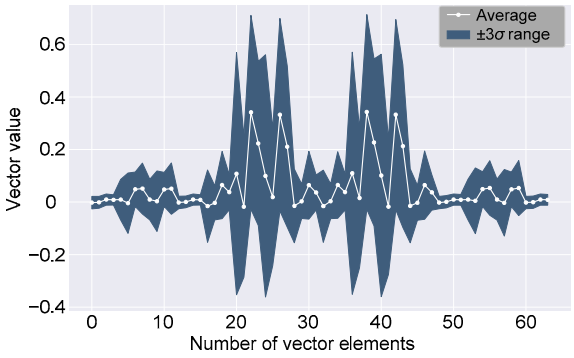


Fig. 1 Element average and 3σ range of SURF vectors

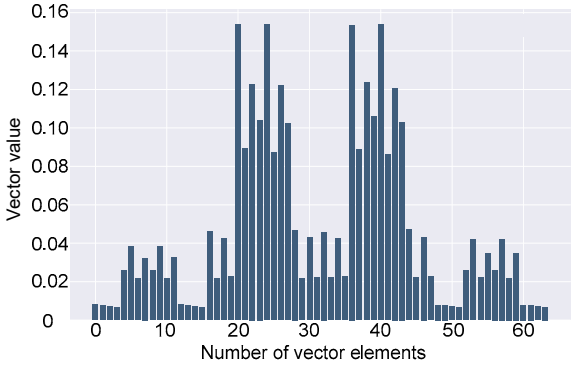


Fig. 2 Element standard deviation of the SURF vectors

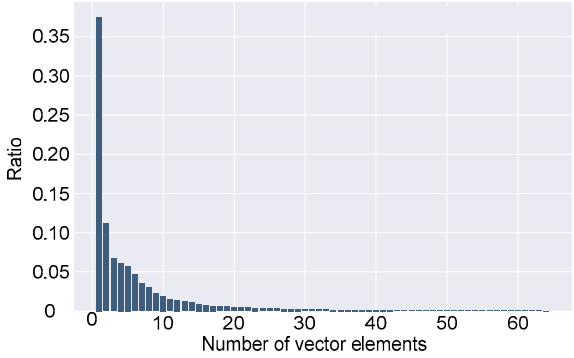


Fig. 3 Element explained variance after PCA mapping

The FMF algorithm is designed based on the above data analysis; the basic structure and process flow are shown in Fig. 4. In the FMF algorithm, to realize the consistency, discrimination, and orderliness of the local image description, the feature vector $\boldsymbol{V}_{\text{surf}}$ acquired by the SURF algorithm is substituted into the PCA algorithm and compressed to 20 dimensions; i.e., set $\boldsymbol{V}_{\text{fmf}}=\boldsymbol{M}_{\text{C}}\boldsymbol{V}_{\text{surf}}$, where $\boldsymbol{M}_{\text{C}}$ is a 20×64 transformation matrix obtained by the PCA algorithm.

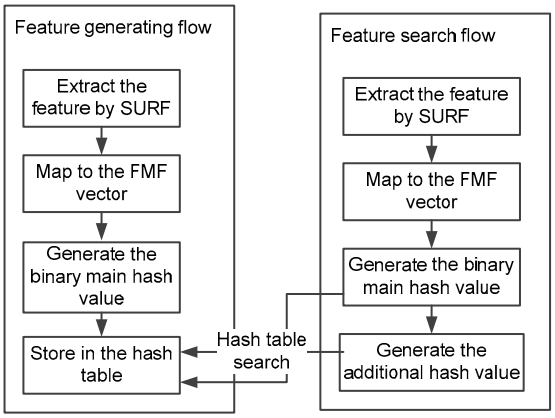


Fig. 4 FMF algorithm process flow

To analyze the performance of $\boldsymbol{V}_{\text{fmf}}$, the Euclidean distance of the point pairs in the matching table is computed. Fig. 5 shows the Euclidean distance distribution of the SURF feature vector pairs of the MVS dataset matched and unmatched points. Fig. 6 illustrates the Euclidean distance distribution after the SURF feature vectors are mapped into the FMF feature vectors. It can be seen from Figs. 5 and 6 that the Euclidean distance distribution of the FMF vectors is basically the same as that of the SURF vector. $\boldsymbol{V}_{\text{fmf}}$ inherits the advantages of the SURF feature vectors, maintaining good consistency and discrimination.

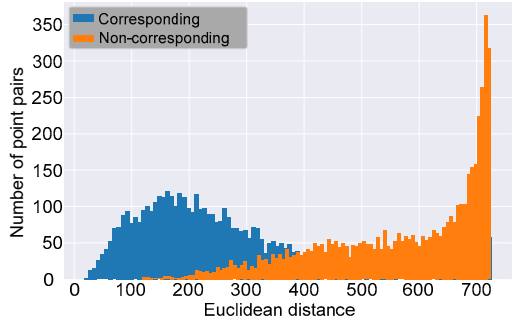


Fig. 5 Euclidean distance distribution of SURF vectors of the corresponding and non-corresponding point pairs

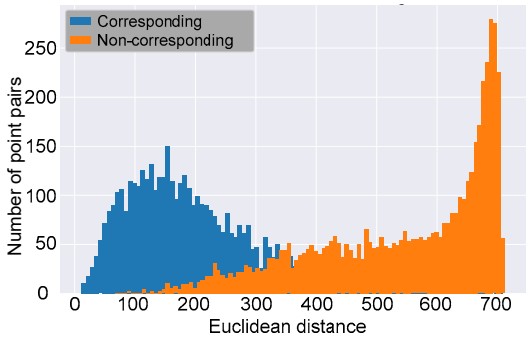


Fig. 6 Euclidean distance distribution of FMF vectors of the corresponding and non-corresponding point pairs

Then, binarization is performed on $\boldsymbol{V}_{\text{fmf}}$. Set elements of $\boldsymbol{V}_{\text{fmf}}\geq 0$ to 1 and elements of $\boldsymbol{V}_{\text{fmf}}<0$ to 0, and we obtain a 20-bit binary feature $\boldsymbol{B}_{\text{fmf}}$, which is further called the "main hash value." Referring to hash search, the hash table established is implemented with a linked list at a length of 2^{20} (i.e., the size of 1 million nodes). In this table, $\boldsymbol{B}_{\text{fmf}}$ is employed as the address to access data; i.e., if other nodes are already stored in the current position, then the new incoming node is linked to the existing node tail. Its storage structure is shown in Fig. 7.

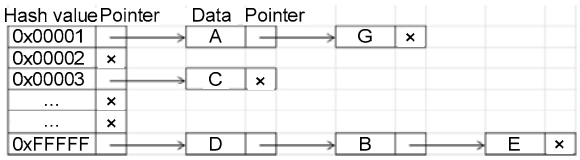


Fig. 7 Linked list storage structure of the hash table

When a new local image patch needs to be matched, repeat the above steps, i.e., calculate the SURF feature vector $\boldsymbol{V}_{\text{surf}}$, map it to $\boldsymbol{V}_{\text{fmf}}$, and convert it into a binary feature $\boldsymbol{B}_{\text{fmf}}$. Then use $\boldsymbol{B}_{\text{fmf}}$ to directly access the value stored at the corresponding position in the hash table, and calculate the Euclidean distance between the stored $\boldsymbol{V}_{\text{fmf}}$ and the current one. If the distance is less than the specified value, it is considered as matched; otherwise, no match is returned. Due to the inherent characteristics of hash table search, the time complexity of this operation is $O(1)$, which means the matching performance is greatly improved, especially for a multi-UAV collaboration task with a large number of feature points. Furthermore, the matching table of the MVS dataset is used to evaluate the mentioned algorithm, in which the Euclidean

distance threshold is determined by the median value of the double peaks in Figs. 5 and 6. The correct matching rate of the original SURF algorithm is 82.21%, whereas that of the current algorithm is 55.67%. From the above matching rates, a decrease of 26.54% in the matching rate is generated in the current algorithm, which is a great loss.

The above method is to be improved to reduce the probability of missing match. From the method of generating $\boldsymbol{B}_{\text{fmf}}$, we know that if the corresponding element of $\boldsymbol{V}_{\text{fmf}}>0$ then the element is 1. By analyzing the distribution of a large number of feature point vectors, it is found that when the absolute value of the element in $\boldsymbol{V}_{\text{fmf}}$ is small, its value is unstable. Different image patches taken at the same point generate different $\boldsymbol{B}_{\text{fmf}}$, which leads to a failure of the hash table search. Therefore, we sort the absolute values of the elements in $\boldsymbol{V}_{\text{fmf}}$ and select the smallest N (number of elements). For example, for the case of N=4, all possible combinations of these four elements are exhausted, from the smallest to the largest (referring to the absolute value of the elements in $\boldsymbol{V}_{\text{fmf}}$). As a result, the corresponding 2^4=16 new hash values are generated based on $\boldsymbol{B}_{\text{fmf}}$, allowing the newly produced hash values to be called additional hash values. Use these 16 hash values in turn to search the hash table, and calculate the Euclidean distance between the stored data and the current $\boldsymbol{V}_{\text{fmf}}$. When the distance is less than the set value, the search ends; otherwise, the search continues. With N=4, the search time of the improved algorithm is increased by up to 2^4 times compared to that of the unimproved algorithm.

2.3 Algorithm verification

To analyze the performance of the FMF algorithm, we compare the original SURF algorithm and the FMF algorithm with different values of *N*. N=0 means no optimization, hence searching for only one node of the hash table, while N=8 means searching for 2^8 nodes at most. Based on the MVS dataset, we execute a test on the matching table. In the test, the same Euclidean distance threshold mentioned in Section 2.2 is used to calculate the correct and incorrect matching rates of the algorithm with different parameters.

The original SURF algorithm obtains the highest probability of correct matching (Table 1). The probability of correct matching of the FMF algorithm without improvement is reduced by 26.54% compared with the original SURF algorithm, showing a large number of missing matches. However, compared with the original SURF algorithm, for cases N=4, 6, and 8 in the improved algorithm, the correct rate has decreased by 18.89%, 14.32%, and 9.42%, respectively, and the probability of missing match has significantly declined, achieving obvious progress compared with the FMF algorithm before improvement.

Table 1 Correct rates of different algorithms with different parameters

Algorithm		Correct rate
SURF		82.21%
FMF	N=0	55.67%
	N=4	63.32%
	N=6	67.89%
	N=8	72.79%

To analyze and compare the performances of the algorithm comprehensively, the precision-recall curves—the standard evaluation method in the field of data classification—with different parameters are further calculated (Fig. 8). The area under the curve (AUC) values are demonstrated in Table 2. It can be seen that although the performance of the improved FMF algorithm is poorer compared with the SURF algorithm, the AUC value reduces only by 0.0458 compared with the SURF algorithm when N=8.

The time of matching 100 feature points in a feature point dataset with different scales is calculated to evaluate the computing efficiency of the FMF algorithm (Fig. 9). As shown in Fig. 9, the matching time of the FMF algorithm is irrelevant to the scale of the feature point dataset, and the time complexity is constantly $O(1)$. When a traditional matching method is employed in the SURF algorithm, the time complexity is proportional to the scale of the feature point dataset, with the time complexity being $O(n)$. However, different values of N (N=0 refers to the unimproved FMF algorithm) have no significant impact on the performance of the FMF algorithm. The time is far less than that of the SURF matching algorithm, especially in the case of a large-scale feature point dataset. Detailed calculation of time of the FMF algorithm with different values of N is shown in Fig. 10.

Table 2 Area under the curve (AUC) of different algorithms with different values of *N*

Algorithm		AUC
SURF		0.9009
FMF	*N*=0	0.7752
	N=4	0.8122
	N=6	0.8320
	N=8	0.8551

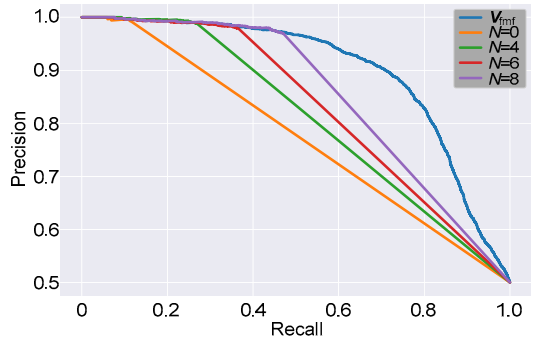


Fig. 8 Precision-recall curves of different algorithms with different values of *N*

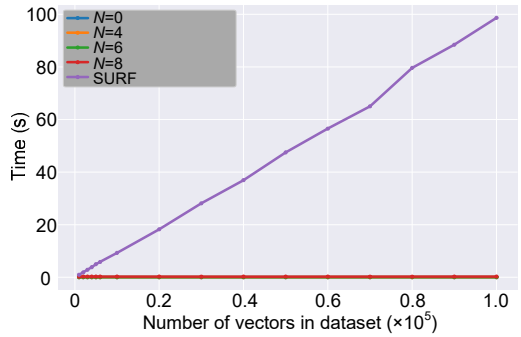


Fig. 9 Matching time of datasets with different scales
Curves of the FMF algorithm are overlapped

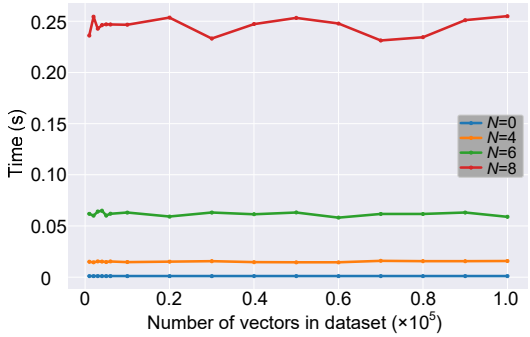


Fig. 10 Matching time of the FMF algorithm with different values of *N* on datasets with different scales

Therefore, we can conclude that the FMF algorithm greatly improves the speed of matching feature points in the large-scale dataset, with the matching speed irrelevant to the scale of the dataset and the theoretical time complexity being $O(1)$. Like other similar algorithms, the performance of the FMF algorithm is verified based on the MVS dataset. The correct matching rate is reduced compared with the SURF algorithm; however, the accuracy rate has dropped by only 9.42% when N=8. When the scale of the dataset is 100 000, the time is reduced by 99.7%. Considering that the application environment of this algorithm, i.e., multiple collaborative monocular SLAMs, is sensitive to computing efficiency because of the large amount of data, certain data errors can be tolerated.

3 Single-UAV navigation

To achieve multi-UAV navigation, it is necessary to realize single-UAV navigation, i.e., ego-motion estimation. Because the IMU sensors equipped with low-cost UAVs are cheap micro-electro-mechanical-system (MEMS) devices, their bias and noise are quite significant. Therefore, the attitude and heading reference system (AHRS) algorithm is exploited to estimate the attitude. What is more, the downward monocular camera and the ultrasonic sensors are used to estimate the vehicle velocity. The algorithm framework is shown in Fig. 11.

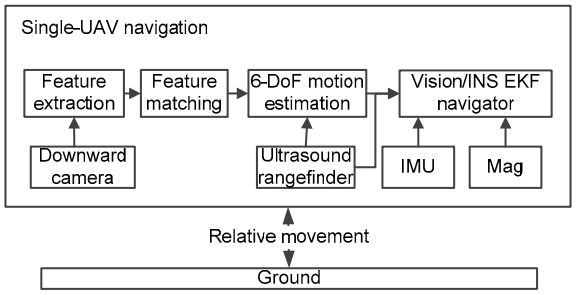


Fig. 11 Single-UAV navigation algorithm framework (DoF: degree-of-freedom)

3.1 6-DoF ego-motion estimation

The FMF feature description algorithm is used to extract feature points from continuous images. As long as a set of matching point pairs are obtained, they can be used to determine the transformation relationship of the images, which can be used to infer the motion matrix of the camera. Because the camera is

connected to the vehicle, the motion matrix of the camera is considered as the motion matrix of the vehicle. Assuming that the homogeneous coordinates of the matching point pairs in two images are $[u_p, v_p, 1]$ and $[u_q, v_q, 1]$, the image homography transformation matrix caused by the motion can be expressed as

$$\boldsymbol{H}=\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}. \tag{1}$$

The coordinate relationship of the matching point pairs by image homography transformation can be expressed as

$$\lambda\begin{bmatrix} u_q \\ v_q \\ 1 \end{bmatrix}=\boldsymbol{H}\begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}=\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}, \tag{2}$$

where λ represents the proportionality factor. In general, there are significantly more matching point pairs than parameters to be solved in $\boldsymbol{H}$. Moreover, it is not guaranteed that all matching point pairs are correct. To ensure the accuracy and robustness of the solution, random sample consensus (RANSAC) is applied.

As for point pairs $[u_p, v_p, 1]$ and $[u_q, v_q, 1]$, Eq. (2) can be transformed into the following non-homogeneous equation set:

$$\begin{cases} u_q = h_{11}u_p + h_{12}v_p + h_{13}, \\ v_q = h_{21}u_p + h_{22}v_p + h_{23}, \\ 1 = h_{31}u_p + h_{32}v_p + h_{33}. \end{cases} \tag{3}$$

Substituting the third equation into the first and second equations in Eq. (3), we obtain

$$\begin{cases} u_q(h_{31}u_p + h_{32}v_p + h_{33}) = h_{11}u_p + h_{12}v_p + h_{13}, \\ v_q(h_{31}u_p + h_{32}v_p + h_{33}) = h_{21}u_p + h_{22}v_p + h_{23}. \end{cases} \tag{4}$$

Set the following intermediate variables:

$$\boldsymbol{a}_m = [u_p, v_p, 1, 0, 0, 0, -u_q u_p, -u_q v_p, -u_q]^{\mathrm{T}}, \tag{5}$$

$$\boldsymbol{a}_n = [0, 0, 0, u_p, v_p, 1, -v_q u_p, -v_q v_p, -v_q]^{\mathrm{T}}, \tag{6}$$

$$\boldsymbol{h} = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^{\mathrm{T}}. \tag{7}$$

Then Eq. (4) can be converted into

$$\begin{cases} \boldsymbol{a}_m^{\mathrm{T}}\boldsymbol{h} = 0, \\ \boldsymbol{a}_n^{\mathrm{T}}\boldsymbol{h} = 0. \end{cases} \tag{8}$$

Therefore, Eq. (3) has been transformed into a linear homogeneous equation. For all feature point pairs that pass the RANSAC test, the following equation holds:

$$\boldsymbol{A}\boldsymbol{h}=\boldsymbol{0}, \tag{9}$$

where $\boldsymbol{A}=[\boldsymbol{a}_{m1}^{\mathrm{T}}, \boldsymbol{a}_{n1}^{\mathrm{T}}, \ldots, \boldsymbol{a}_{mn}^{\mathrm{T}}, \boldsymbol{a}_{nn}^{\mathrm{T}}]^{\mathrm{T}}$ stands for the point-pair dataset matrix. Eq. (9) can be regarded as a least-squares problem of the linear homogeneous equation. We use the method of singular value decomposition (SVD) combined with the M estimation method (Zhang, 1997) to find the corresponding homography matrix $\boldsymbol{H}$. The feature point matching relationship has been converted into an image transformation relationship. To solve the camera motion matrix from image transformation, assume that the intrinsic camera parameter is $\boldsymbol{A}_{\mathrm{c}}$. Setting $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ as the projection coordinates of the fixed ground point $\boldsymbol{p}$ in two images before and after motioning $\tilde{\boldsymbol{m}}_1$ and $\tilde{\boldsymbol{m}}_2$ as the corresponding pixel coordinates, the following relationships can be obtained:

$$\begin{cases} Z_1^{\mathrm{c}}\tilde{\boldsymbol{m}}_1 = \boldsymbol{A}_{\mathrm{c}}\boldsymbol{m}_1, \\ Z_2^{\mathrm{c}}\tilde{\boldsymbol{m}}_2 = \boldsymbol{A}_{\mathrm{c}}\boldsymbol{m}_2, \end{cases} \tag{10}$$

$$\lambda\tilde{\boldsymbol{m}}_2 = \boldsymbol{H}\tilde{\boldsymbol{m}}_1, \tag{11}$$

where Z_1^{c} and Z_2^{c} represent Z coordinates of the projection points in the camera coordinate system and $\lambda = Z_1^{\mathrm{c}} / Z_2^{\mathrm{c}}$ represents the scale factor. Because $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ are the projected coordinates of the shooting point $\boldsymbol{p}$ at the two moments before and after motions, it can be assumed that camera motion is a combination of translation movement $\boldsymbol{t}$ and rotation movement $\boldsymbol{R}$. The relationship between the movements is shown in Fig. 12, whose mathematical relationship can be expressed as

$$\boldsymbol{m}_2 = \boldsymbol{R}(\boldsymbol{m}_1 + \boldsymbol{t}). \tag{12}$$

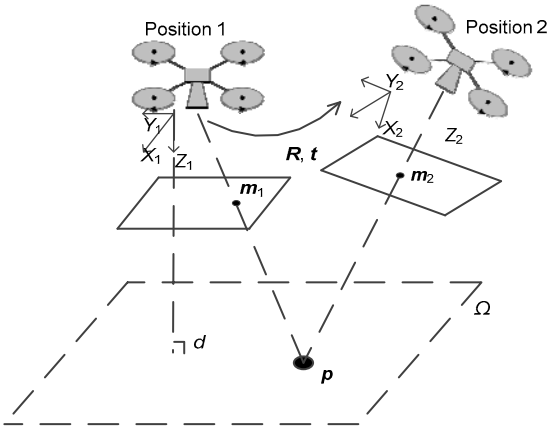


Fig. 12 Position relationship of the same point observed by the vehicle before and after motions

Setting $\boldsymbol{n}$ as the vertical normal vector of plane Ω and d the distance between position 1 and plane Ω, Eq. (12) can be transformed into

$$\boldsymbol{m}_2 = \boldsymbol{R}\left(\boldsymbol{I} + \frac{\boldsymbol{t}\boldsymbol{n}^{\mathrm{T}}}{d}\right)\boldsymbol{m}_1. \tag{13}$$

Substituting Eq. (10) into Eq. (13), we can obtain

$$\boldsymbol{H} = \boldsymbol{A}_{\mathrm{c}}\boldsymbol{R}\left(\boldsymbol{I} + \frac{\boldsymbol{t}\boldsymbol{n}^{\mathrm{T}}}{d}\right)\boldsymbol{A}_{\mathrm{c}}^{-1}, \tag{14}$$

$$\boldsymbol{H}_{\mathrm{c}} = \boldsymbol{A}_{\mathrm{c}}^{-1}\boldsymbol{H}\boldsymbol{A}_{\mathrm{c}} = \boldsymbol{R}\left(\boldsymbol{I} + \frac{\boldsymbol{t}\boldsymbol{n}^{\mathrm{T}}}{d}\right), \tag{15}$$

where $\boldsymbol{H}_{\mathrm{c}}$ stands for the calibration homography matrix. According to Tsai et al. (1982), we perform SVD on $\boldsymbol{H}_{\mathrm{c}}$, thus resulting in $\boldsymbol{H}_{\mathrm{c}}=\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\mathrm{T}}$, where $\boldsymbol{U}=[\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3]$, $\boldsymbol{V}=[\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3]$, and $\boldsymbol{D}=\mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$. Referring to Caballero et al. (2006), the translation movement $\boldsymbol{t}$, rotation movement $\boldsymbol{R}$, and unit normal vector $\boldsymbol{n}$ can be expressed as

$$\begin{cases} \boldsymbol{t} = \mu d\left[\beta\boldsymbol{u}_1 - \left(\dfrac{\lambda_3}{\lambda_2} - s\alpha\right)\boldsymbol{u}_3\right], \\ \boldsymbol{R} = \boldsymbol{U}\begin{bmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -s\beta & 0 & s\alpha \end{bmatrix}\boldsymbol{V}^{\mathrm{T}}, \\ \boldsymbol{n} = \dfrac{\delta\boldsymbol{v}_1 + \boldsymbol{v}_3}{\mu}, \end{cases} \tag{16}$$

where

$$\begin{cases} \alpha = \dfrac{\lambda_1 + s\lambda_3\delta^2}{\lambda_2(1+\delta^2)}, \\ \beta = \pm\sqrt{1-\alpha^2}, \\ s = \det(\boldsymbol{U})\det(\boldsymbol{V}), \\ \delta = \pm\sqrt{\dfrac{\lambda_1^2 - \lambda_2^2}{\lambda_2^2 - \lambda_3^2}}, \end{cases} \tag{17}$$

λ_1, λ_2, and λ_3 ($\lambda_1\geq\lambda_2\geq\lambda_3$) represent three eigenvalues of SVD, and μ is the scale factor that guarantees $\|\boldsymbol{n}\|=1$.

The translation and rotation movements of the vehicle at a time interval can be obtained. If the movement is divided by the interval time, we can obtain the corresponding velocity and angular velocity.

3.2 Fusion of vision and inertial data

As described in Li et al. (2017), there are several methods for state estimation of multi-input and multi-output nonlinear systems; however, KF remains the most common one because the extended Kalman filter (EKF) can achieve an effective estimate of its system state by linearizing the nonlinear system. To obtain a vehicle state with high precision and dynamics, EKF is used to fuse the velocity obtained in visual measurement and the data obtained in MEMS measurement. In single-UAV navigation, the vehicle can be regarded as a 6-degree-of-freedom (6-DoF) rigid body moving in a 3D space. The filter regards the 6-DoF state and the MEMS sensor error as the model state, and the vehicle state can be defined as

$$\boldsymbol{X} = [v, q, b_{\mathrm{a}}, b_{\omega}]^{\mathrm{T}}, \tag{18}$$

where v represents the velocity of the vehicle in the navigation coordinates of xyz, q the quaternion of the vehicle attitude, b_{a} the accelerometer bias, and b_{ω} the gyroscope bias. The continuous state space equation of the system is demonstrated as

$$\dot{\boldsymbol{X}}(t) = \boldsymbol{f}[\boldsymbol{X}(t), \boldsymbol{U}(t), t] + \boldsymbol{G}[\boldsymbol{X}(t), t]\boldsymbol{w}(t). \tag{19}$$

Assuming that the IMU measurement axis and the vehicle body coordinate system coincide

according to the principle of inertial navigation, we can obtain

$$\boldsymbol{G}=\begin{bmatrix}-\boldsymbol{C}_b^n(q) & \boldsymbol{O}_{3\times3} & \boldsymbol{O}_{3\times3} & \boldsymbol{O}_{3\times3}\\ \boldsymbol{O}_{4\times3} & -\frac{1}{2}\boldsymbol{Z}(q) & \boldsymbol{O}_{4\times3} & \boldsymbol{O}_{4\times3}\\ \boldsymbol{O}_{3\times3} & \boldsymbol{O}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{O}_{3\times3}\\ \boldsymbol{O}_{3\times3} & \boldsymbol{O}_{3\times3} & \boldsymbol{O}_{3\times3} & \boldsymbol{I}_{3\times3}\end{bmatrix}, \tag{20}$$

$$\boldsymbol{f}=\begin{bmatrix}\boldsymbol{C}_b^n(q)\bar{\boldsymbol{a}}^b+\boldsymbol{g}\\ \frac{1}{2}\boldsymbol{\Omega}(\bar{\omega}^b)\boldsymbol{q}\\ \boldsymbol{O}_{3\times1}\\ \boldsymbol{O}_{3\times1}\end{bmatrix}, \tag{21}$$

$$\boldsymbol{w}=[n_a,\, n_\omega,\, n_{ba},\, n_{b\omega}]^{\mathrm{T}}, \tag{22}$$

$$\boldsymbol{U}=[\boldsymbol{a}_m^b,\ \boldsymbol{\omega}_m^b], \tag{23}$$

where $\boldsymbol{g}$ stands for the gravity acceleration vector and $\boldsymbol{U}$ the input control of the inertial sensor. The acceleration vector after calibration is $\bar{\boldsymbol{a}}^b=\boldsymbol{a}_m^b-\boldsymbol{b}_{\mathrm{a}}$ with the angular velocity vector $\bar{\boldsymbol{\omega}}^b=\boldsymbol{\omega}_m^b-\boldsymbol{b}_\omega$ after calibration. $\boldsymbol{C}_b^n$ represents the Euler direction cosine conversion matrix from the vehicle body system to the navigation system, i.e., $\boldsymbol{C}_b^n=(\boldsymbol{C}_n^b)^{\mathrm{T}}$.

The quaternion attitude differential equation is

$$\dot{\boldsymbol{q}}=\frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}^b)\boldsymbol{q}=\frac{1}{2}\boldsymbol{Z}(q)\boldsymbol{\omega}^b, \tag{24}$$

where

$$\boldsymbol{\Omega}(\omega)=\begin{bmatrix}0 & -\omega_1 & -\omega_2 & -\omega_3\\ \omega_1 & 0 & \omega_3 & -\omega_2\\ \omega_2 & -\omega_3 & 0 & \omega_1\\ \omega_3 & \omega_2 & -\omega_1 & 0\end{bmatrix}, \tag{25}$$

$$\boldsymbol{Z}(q)=\begin{bmatrix}-q_1 & -q_2 & -q_3\\ q_0 & -q_3 & q_2\\ q_3 & q_0 & -q_1\\ -q_2 & q_1 & q_0\end{bmatrix}. \tag{26}$$

The state equation of continuous system (19) can be discretized into

$$\delta\boldsymbol{X}(t)=\boldsymbol{X}(t)-\hat{\boldsymbol{X}}(t)=[\delta v,\ \delta q,\ \delta b_a,\ \delta b_\omega]^{\mathrm{T}}, \tag{27}$$

$$\delta\dot{\boldsymbol{X}}(t)=\boldsymbol{F}(t)\delta\boldsymbol{X}(t)+\boldsymbol{G}(t)\boldsymbol{w}(t), \tag{28}$$

$$\delta\boldsymbol{X}_k=\boldsymbol{\Phi}_{k,k-1}\delta\boldsymbol{X}_{k-1}+\boldsymbol{W}_{k-1}, \tag{29}$$

where

$$\boldsymbol{F}(t)=\left.\frac{\partial\boldsymbol{f}[\boldsymbol{X}(t)]}{\partial\boldsymbol{X}(t)}\right|_{\boldsymbol{X}(t)=\hat{\boldsymbol{X}}(t)}. \tag{30}$$

Setting Δt as the sampling period of the inertial device, Eq. (29) can be expressed in a different form as

$$\boldsymbol{\Phi}_{k,k-1}\approx\boldsymbol{I}+\boldsymbol{F}(t_{k-1})\Delta t=\boldsymbol{I}+\boldsymbol{F}_{k-1}\Delta t. \tag{31}$$

According to the EKF filtering principle, the attitude observation update equation based on inertial sensors is established along with the velocity observation update equation based on vision. The attitude observation update equation of the model is defined as

$$\boldsymbol{Z}_{1,k}=\boldsymbol{H}_1\boldsymbol{X}_k+\boldsymbol{V}_{1,k}, \tag{32}$$

where

$$\boldsymbol{Z}_{1,k}=\boldsymbol{q}, \tag{33}$$

$$\boldsymbol{V}_{1,k}\sim N[0,\boldsymbol{R}_1,k], \tag{34}$$

$$\boldsymbol{H}_1=[\boldsymbol{O}_{4\times3},\boldsymbol{I}_{4\times4},\boldsymbol{O}_{4\times3},\boldsymbol{O}_{4\times3}]. \tag{35}$$

The velocity observation update equation is defined as

$$\boldsymbol{Z}_{2,k}=\boldsymbol{H}_2\boldsymbol{X}_k+\boldsymbol{V}_{2,k}, \tag{36}$$

where

$$\boldsymbol{Z}_{2,k}=\boldsymbol{v}, \tag{37}$$

$$\boldsymbol{V}_{2,k}\sim N[0,\boldsymbol{R}_2,k], \tag{38}$$

$$\boldsymbol{H}_2=[\boldsymbol{I}_{3\times3},\boldsymbol{O}_{3\times4},\boldsymbol{O}_{3\times3},\boldsymbol{O}_{3\times3}]. \tag{39}$$

The time update equations are

$$\hat{\boldsymbol{X}}_{k,k-1}=\hat{\boldsymbol{X}}_{k-1}+\boldsymbol{f}(\hat{\boldsymbol{X}}_{k-1})\Delta t, \tag{40}$$

$$\boldsymbol{P}_{k,k-1}=\boldsymbol{\Phi}_{k,k-1}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}_{k,k-1}^{\mathrm{T}}+\boldsymbol{Q}_{k-1}. \tag{41}$$

The measurement update equations when obtaining attitude data are depicted as

$$\boldsymbol{K}_{1,k}=\boldsymbol{P}_{k,k-1}\boldsymbol{H}_1^{\mathrm{T}}(\boldsymbol{H}_1\boldsymbol{P}_{k,k-1}\boldsymbol{H}_1^{\mathrm{T}}+\boldsymbol{R}_1)^{-1}, \tag{42}$$

$$\hat{\boldsymbol{X}}_k = \hat{\boldsymbol{X}}_{k,k-1} + \boldsymbol{K}_{1,k}(\boldsymbol{Z}_{1,k} - \boldsymbol{H}_1\hat{\boldsymbol{X}}_{k,k-1}), \quad (43)$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_{1,k}\boldsymbol{H}_1)\boldsymbol{P}_{k,k-1}. \quad (44)$$

The measurement update equations when obtaining velocity data are

$$\boldsymbol{K}_{2,k} = \boldsymbol{P}_{k,k-1}\boldsymbol{H}_2^{\mathrm{T}}(\boldsymbol{H}_2\boldsymbol{P}_{k,k-1}\boldsymbol{H}_2^{\mathrm{T}} + \boldsymbol{R}_2)^{-1}, \quad (45)$$

$$\hat{\boldsymbol{X}}_k = \hat{\boldsymbol{X}}_{k,k-1} + \boldsymbol{K}_{2,k}(\boldsymbol{Z}_{2,k} - \boldsymbol{H}_2\hat{\boldsymbol{X}}_{k-m/2}), \quad (46)$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_{2,k}\boldsymbol{H}_2)\boldsymbol{P}_{k,k-1}. \quad (47)$$

Thus, the fusion of vision and inertial data has been achieved through EKF, and available attitude and velocity estimates have been obtained.

4 Multi-UAV navigation

In a classic SLAM system, the framework can be divided into front- and back-end modules. The back-end module includes map building and optimization and loop closure detection. The focus of the structure is to build a complete and accurate map, and further to estimate the current position of the vehicle from the map. In the map building and optimization step, the latest feature points received need to be matched with the existing feature points in the map. When using the most basic exhaustive search method, $K\times N\times M$ times Euclidean distance calculations are required for the latest feature points received (where K represents the number of the latest feature points received, N the number of vehicles, and M the number of feature points stored in the map). Because this process involves massive floating-point addition and multiplication operations with the time complexity of $O(K\times N\times M)$, when dealing with such a large number of vehicles, the long running time and great number of feature points collected in the map can impose serious negative effects on the real-time system. Currently, only local search is executed to improve the system performance, which will eventually lead to a loop closure detection problem. Considering the above problems and task requirements, a multi-UAV navigation framework is designed (Fig. 13). This framework is not aimed to build a complete and accurate map, but to estimate the relative position between adjacent vehicles.

In this framework, the method mentioned in Section 3 is employed to achieve single-UAV navigation. Each vehicle transmits the feature vector $\boldsymbol{V}_{\mathrm{fmf}}$ together with its related information (i.e., feature point pixel position and vehicle status) extracted from the image to the independent back-end module of each vehicle (Fig. 13). In the back-end module, each vehicle completes the building and update of its own map, and realizes the localization in its own map. This position relationship is referred to as vehicle-to-map (V2M) for simplification. At the same time, the position relationship between maps is obtained as an intermediary by the input feature vector. For simplification, this position relationship is called map-to-map (M2M). Relying on V2M and M2M, we can solve the problem of relative position between vehicles.

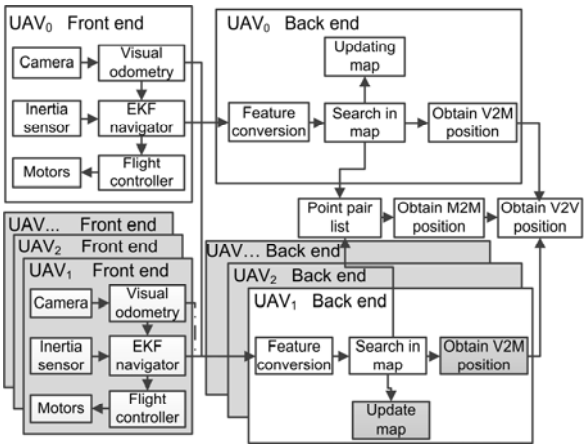


Fig. 13 Front- and back-end algorithm framework of multi-UAV SLAM

4.1 Self-position estimation

In the back-end module, the map is a sparse landmark metric one, in which each landmark contains only feature point vectors, spatial positions, and position error information without terrain or image information. Maps are stored in a list-type data structure. During the ego-motion estimation process of the front-end module, $\boldsymbol{V}_{\mathrm{fmf}}$ is extracted from the image and sent to the back-end module. In the back-end module, according to the FMF algorithm, $\boldsymbol{V}_{\mathrm{fmf}}$ is converted to $\boldsymbol{B}_{\mathrm{fmf}}$ and multiple additional hash values are generated. We use the hash table method to search for the feature points stored in the maps.

If matching feature points do not exist, a new landmark point is created using the inverse-depth

non-delayed landmark initialization method proposed by Montiel et al. (2006). Information, such as feature vectors, angles, positions, and errors, is stored in the map data list. The landmark location parameters are defined as follows:

$$\boldsymbol{Y}_i=[x_0, y_0, z_0, \theta, \phi, \rho]^{\mathrm{T}}, \tag{48}$$

where $[x_0, y_0, z_0]$ is the position coordinate of the vehicle when the landmark is observed, and θ, ϕ, and ρ are the azimuth, elevation, and inverse of the length of the line segment from $\boldsymbol{p}_0$ to the landmark point, respectively. Fig. 14 is the schematic of the position relationship of relevant points.

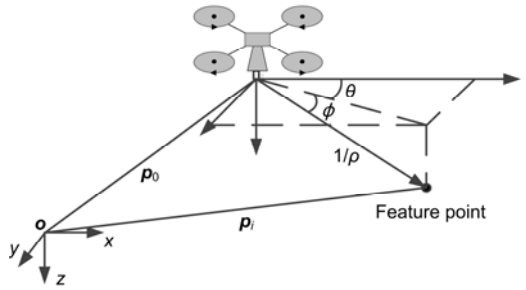


Fig. 14 Schematic of vehicle observing feature points on the map

After the information is stored in the map data list, the offset is stored in the hash table (the storage location is determined by the main hash value $\boldsymbol{B}_{\mathrm{fmf}}$ of the FMF algorithm).

If the valid data is matched by the FMF algorithm, the relevant position and error information of landmarks are accessed from the map data list using the offset stored in the hash table. The landmark position in the navigation coordinate system can be expressed as

$$\boldsymbol{p}_i=\boldsymbol{q}[\boldsymbol{p}, \boldsymbol{Y}_i]=\boldsymbol{p}_0+\frac{\boldsymbol{m}[\theta, \phi]}{\rho}, \tag{49}$$

$$\boldsymbol{m}[\theta, \phi]=\begin{bmatrix}\cos\theta\cos\phi\\ \sin\theta\cos\phi\\ \sin\phi\end{bmatrix}. \tag{50}$$

With the method proposed by Montiel et al. (2006), the position and error of the landmarks are updated. Landmarks can be accurately located after several matching and updating operations. After map update is completed, we obtain the vehicle position relative to the map using the Rao-Blackwellized particle filter (RBPF) method, referring to the carrier localization algorithm (Montemerlo et al., 2002, 2003) in FastSLAM. So far, the position relationship between the vehicle and its own map has been obtained.

4.2 Relative-position estimation

In the back-end module, because each vehicle uses independent maps, after determining the V2M position relationship, the position relationship between different vehicle maps needs to be determined.

Taking UAV_0 and UAV_x as examples, to estimate the relative position between these two vehicles, the feature points that UAV_0 accepts from the front-end module are used as intermediaries to perform matching in the maps of UAV_0 and UAV_x, so that the matching relationship between the points in the two maps is obtained. After matching, the feature points in the maps are organized into point pairs and stored in an independent point-pair list. The point-pair list is employed to determine the position relationship between maps.

Note that, in practice, the point-pair list between any two vehicles uses feature points provided by the front-end modules of both vehicles as a matching intermediary. Any vehicle that needs to obtain the relative position has an independent point-pair list. In this process, a large number of feature point matching operations are involved, because the framework uses the FMF algorithm and the complexity of matching operations for a large-scale feature point dataset is extremely low. As a result, the efficiency of the system can be guaranteed.

In this framework, we pay attention to only the latest position relationship, so each point-pair list is a fixed-length circular list, which saves only a certain number of the latest point pairs. Moreover, as the relative position between the maps changes slowly, the process of calculating the M2M relationship by the point-pair list is performed at a low frequency and is performed asynchronously with the image acquisition process.

The detailed mathematical calculation process of M2M is as follows: Let the feature point set collected by UAV_0 be $O=\{O_0, O_1, \ldots, O_i\}$, the point set matched with the UAV_0 map be $P=\{P_0, P_1, \ldots, P_j\}$, and the

point set matched with the UAV_x map be $Q=\{Q_0, Q_1, \ldots, Q_k\}$. Set S as the common point-pair set of P and Q, that is, $S=\{<P_0, Q_0>, <P_1, Q_1>, \ldots, <P_l, Q_l>\}$. Because P and Q are the matching results with point set O, the corresponding points in P and Q are matching point pairs (due to the overlap and matching problems, there are several feature points that do not match, so S is a subset of P and Q). After point-pair set S is obtained, it is stored in an independent point-pair list.

Supposing that there is a point $\boldsymbol{o}$ and that the corresponding point pair in the point-pair list is $<\boldsymbol{p}, \boldsymbol{q}>$, the coordinates of $\boldsymbol{p}$ in the UAV_0 map can be expressed as homogeneous coordinates $[X_p, Y_p, Z_p, 1]^{\mathrm{T}}$, while those of $\boldsymbol{q}$ in the UAV_x map can be expressed as homogeneous coordinates $[X_q, Y_q, Z_q, 1]^{\mathrm{T}}$. Because the point pair $<\boldsymbol{p}, \boldsymbol{q}>$ corresponds to the same point $\boldsymbol{o}$ in the 3D space, the position relationship of the point pair can be expressed as a non-homogeneous linear equation (51):

$$\begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix} \begin{bmatrix} X_q \\ Y_q \\ Z_q \\ 1 \end{bmatrix}. \tag{51}$$

Expand Eq. (51) as an equation set as follows:

$$\begin{cases} X_p = M_{11}X_q + M_{12}Y_q + M_{13}Z_q + M_{14}, \\ Y_p = M_{21}X_q + M_{22}Y_q + M_{23}Z_q + M_{24}, \\ Z_p = M_{31}X_q + M_{32}Y_q + M_{33}Z_q + M_{34}, \\ 1 = M_{41}X_q + M_{42}Y_q + M_{43}Z_q + M_{44}. \end{cases} \tag{52}$$

Using a method similar to the one described in Section 3, substitute the fourth equation of Eq. (52) into the first three to obtain the equation set (53):

$$\begin{cases} X_p(M_{41}X_q + M_{42}Y_q + M_{43}Z_q + M_{44}) \\ \quad = M_{11}X_q + M_{12}Y_q + M_{13}Z_q + M_{14}, \\ Y_p(M_{41}X_q + M_{42}Y_q + M_{43}Z_q + M_{44}) \\ \quad = M_{21}X_q + M_{22}Y_q + M_{23}Z_q + M_{24}, \\ Z_p(M_{41}X_q + M_{42}Y_q + M_{43}Z_q + M_{44}) \\ \quad = M_{31}X_q + M_{32}Y_q + M_{33}Z_q + M_{34}. \end{cases} \tag{53}$$

Considering

$$\begin{aligned} \boldsymbol{M} = [&M_{11}, M_{12}, M_{13}, M_{14}, M_{21}, M_{22}, M_{23}, M_{24}, \\ &M_{31}, M_{32}, M_{33}, M_{34}, M_{41}, M_{42}, M_{43}, M_{44}]^{\mathrm{T}} \end{aligned} \tag{54}$$

as the variable to be solved and setting $\boldsymbol{A}$ as

$$\begin{cases} \boldsymbol{A}_1 = [X_q, Y_q, Z_q, 1, 0, 0, 0, 0, 0, 0, 0, 0, \\ \qquad -X_pX_q, -X_pY_q, -X_pZ_q, -X_p], \\ \boldsymbol{A}_2 = [0, 0, 0, 0, X_q, Y_q, Z_q, 1, 0, 0, 0, 0, \\ \qquad -X_pX_q, -X_pY_q, -X_pZ_q, -Y_p], \\ \boldsymbol{A}_3 = [0, 0, 0, 0, 0, 0, 0, 0, X_q, Y_q, Z_q, 1, \\ \qquad -X_pX_q, -X_pY_q, -X_pZ_q, -Z_p], \end{cases} \tag{55}$$

then Eq. (51) can be simplified as

$$\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix} \boldsymbol{M} = 0. \tag{56}$$

The non-homogeneous equation (51) has been transformed into a homogeneous equation (56), where $\boldsymbol{M}$ is a vector to be solved and $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \boldsymbol{A}_3 \end{bmatrix}$ is a known matrix. Considering that the number of point pairs in the point-pair list is significantly larger than that of the coefficients to be determined, the RANSAC method is used to remove outliers. Using the point-pair set verified by RANSAC, the equation set can be established, which, in turn, can be regarded as the least-squares problem. The SVD method can be used to obtain the optimal solution to vector $\boldsymbol{M}$. The optimal solution is the column vector corresponding to the smallest singular value of diagonal matrix $\boldsymbol{V}$. So far, the transformation relationship of point pair $<\boldsymbol{p}, \boldsymbol{q}>$ can be expressed as

$$\boldsymbol{p} = \boldsymbol{M} \times \boldsymbol{q}. \tag{57}$$

Therefore, based on the obtained matrix $\boldsymbol{M}$, the position of any point in the UAV_x map can be projected to the UAV_0 map. Based on V_02M_0 (position relationship of UAV_0 in the UAV_0 map), V_x2M_x (position relationship of UAV_x in the UAV_x map), and M_02M_x (position relationship between the maps just

obtained), the position relationship between two UAVs, called $V_0 2V_x$, can be obtained.

5 Simulation and experiments

5.1 Simulation

In the simulation, a set of closed-loop simulation programs are designed using MATLAB as the development environment. To simulate the real flight conditions of the vehicle, the aerial image (Fig. 15) is used as the ground image. The program can generate inertial sensor data with reasonable noise and downward camera images according to the input flight trajectory. MATLAB is used to realize all the algorithms and data structures. To verify the single- and multi-UAV navigation frameworks and the FMF algorithm, two vehicles successively take off from the starting point and fly along a preset circular trajectory. The theoretical distance between these two vehicles remains constant during the flight. Other parameters are as follows: downward camera resolution 300×300, field-of-view (FoV) 45°, aircraft initial height 20 m, image update frequency 5 Hz, and inertial sensor update frequency 50 Hz.

Fig. 15 Ground picture used in the simulation

The simulated flight state is demonstrated in Fig. 16, where the black circle marks the truth value of the theoretical trajectory. The blue vehicle takes off first and the red one takes off second. The triangle symbols stand for the vehicles, and the following curves represent the historical trajectories. The bottom wireframe of the vehicle is the estimated camera FoV at the current moment, and the black points and green curves on the ground in the 3D figure are the estimates of positions and position variances of the feature points, respectively. Pictures on the right are the current camera images of the two vehicles.

The position curves of the flight are shown in Fig. 17. It can be seen that using the single-UAV navigation algorithm, the vehicle position is well estimated and each vehicle accurately follows the truth curve independently. Compared with the inertial navigation method, the proposed single-UAV navigation algorithm greatly improves the accuracy of position estimation. Fig. 18 shows the position error curves of the flight. It can be seen that the error of position estimation is maintained at a low level but expands with time.

In terms of multi-UAV navigation, the distance between two vehicles is calculated using the relative position estimation algorithm. Results are shown in Fig. 19. The estimate varies around the truth value, and there is no error expansion over time. The average error is 0.3 m and the standard deviation is 0.32 m, both at a relatively low level. Thus, the design goal of the algorithm has been achieved.

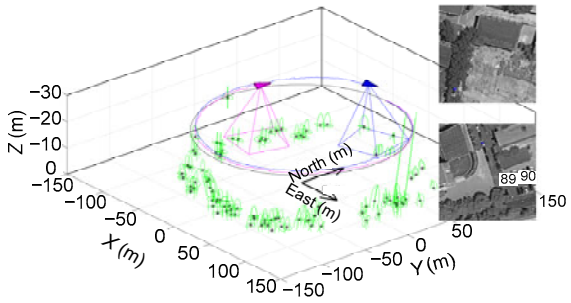


Fig. 16 Three-dimensional flight state of multi-UAV SLAM simulation (References to color refer to the online version of this figure)

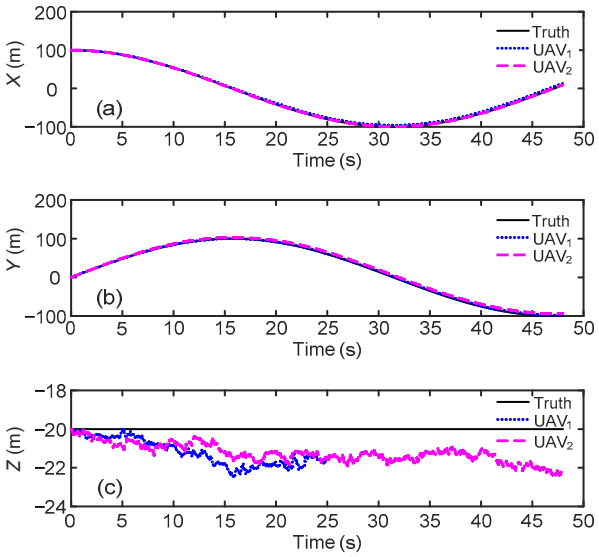


Fig. 17 Position curves of the vehicles: (a) *X*; (b) *Y*; (c) *Z*

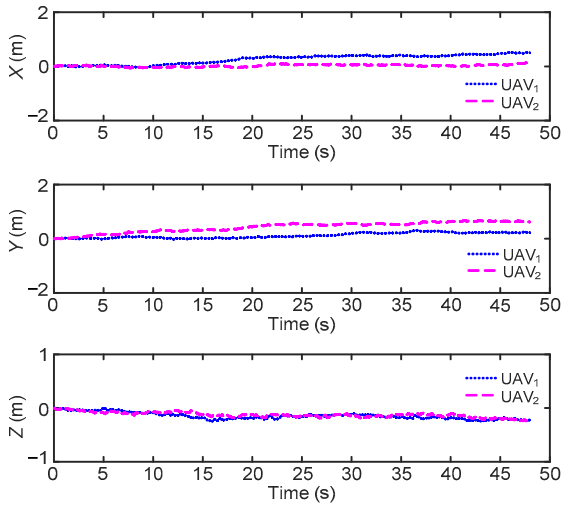


Fig. 18 Position error curves of the vehicles: (a) *X*; (b) *Y*; (c) *Z*

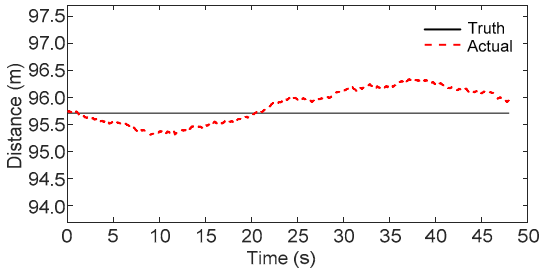


Fig. 19 Curve of the relative distance between vehicles

5.2 Real flight experiments

In real flight experiments, there are two quadrotor UAVs. Each vehicle has an F450 frame quadrotor X standard layout and is equipped with PX4. The gyroscope, accelerometer, magnetometer, and barometric altimeter are all built into the vehicle. The vehicle attitude solving and controlling algorithm runs onboard. The vehicle can output the sensor state and accept user control commands. The wireless module is used for communication with the ground monitor software on a personal computer. A radio control (RC) receiver is used to manually control the vehicle. A DC-DC module is used to supply power for the equipment. Fig. 20 shows the UAV and the equipment.

The bottom parts of the vehicle house the main experimental equipment, including an ultrasonic range sensor to measure the distance off the ground and a global shutter camera implemented in a vertically downward position to avoid anamorphosis in high-velocity motion. The model is AR0144, which

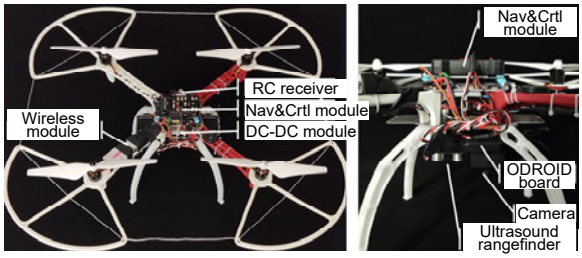


Fig. 20 UAV and equipment used in the experiments

can realize 640×480@50 Hz image collection. The ODROID process board is a 4-core high-performance system-on-a-chip (SoC) based on ARM, and a Linux system runs on it. It collects image data through the USB and receives the PX4 state data from the serial ports. The image process and navigation algorithm run on the board. Finally, control commands are sent to PX4.

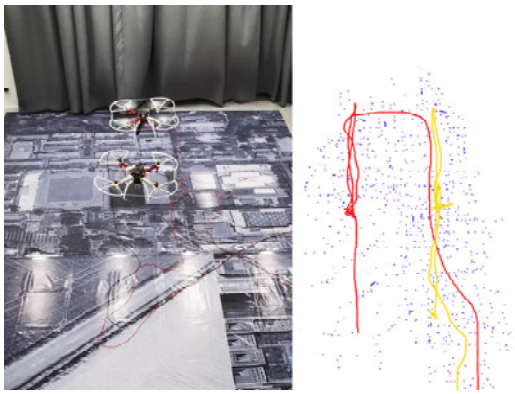
The experiment environment is about 4 m (width)×6 m (length)×4 m (height). To simulate the real application environment, the district (structural image) and grass (nonstructural image) aerial photos are stuck on the ground.

To comprehensively verify the FMF algorithm and multi-UAV SLAM collaboration framework, a two-vehicle formation flight test is designed in the real flight experiments. In this test, each of the two vehicles executes the proposed algorithm. After completing the estimation of their own positions, the feature point information is exchanged via the data link to estimate the relative positions of the two vehicles.

Detailed experiment steps are as follows: The master UAV flies first to the center of the field, and then the second UAV flies to the field. After relative position resolution, the second vehicle flies to the side of the master UAV. Next, the master UAV performs linear motion on both the left and right sides, and then the second UAV follows the master UAV.

Fig. 21a shows the flight environment and the state of the vehicles. Fig. 21b demonstrates the flight trajectories and the feature points of the two vehicles. It can be seen that the estimated trajectory is consistent with that of the actual one in the flight experiments. The second vehicle achieves good follow-up with the master UAV, proving the effectiveness of relative position estimation and the real-time performance of the system. In addition, as for the feature point map established, there are more feature

points in the structural ground image area and fewer points in the nonstructural area. Even in the nonstructural area, the vehicle still achieves effective position estimation.



(a) (b)

Fig. 21 Flight environment and the state of the two vehicles (a) and flight trajectories and the feature points of the two vehicles (b)

6 Conclusions and future work

In this study, we have presented an efficient FMF algorithm using the key concepts of PCA and hash. The aim is to realize real-time coordination and application of multiple UAVs with low cost and limited dependence. According to the results of the experiments, the conclusions are as follows:

1. The calculation costs of the feature vector transformation and hash value generation are low, and the time complexity of matching is $O(1)$ (almost irrelevant to the data scale). Thus, the FMF algorithm is suitable for the large-data environment.

2. Even though the correct matching rate is lower than that of the SURF algorithm, considering that the RANSAC algorithm is applied to motion estimation, a lot of mismatches do not have marked influence on the overall function.

3. We have constructed an innovative multi-UAV SLAM framework based on FMF, and emphasized optimization of real-time multi-UAV collaboration without establishing a complete global map. Specifically, we have solved the problem of low computing efficiency as the data scale grows in the traditional SLAM algorithm. In addition, it is superior in terms of both low sensor cost and limited dependence on the environment. Finally, the simulation and real flight experiments have verified the feasibility of the FMF algorithm and this framework.

However, this framework has weaknesses with respect to the keypoints in the current test. The number of keypoints cannot meet the requirements occasionally. In the future, we will use the adaptive particle swarm optimization proposed by Moallem and Razmjooy (2012) to optimize the threshold value, by ensuring the stability of the number of keypoints to guarantee the reliability of position estimation.

Contributors

Tian-miao WANG proposed methodology. Yi-cheng ZHANG formulated the overarching research goals and aims. Chao-lei WANG executed theoretical reasoning. Yi-cheng ZHANG designed the research and software. Jian-hong LIANG administrated the project. Yi-cheng ZHANG and Chao-lei WANG processed the data. Yang CHEN implemented the investigation and validation. Yi-cheng ZHANG drafted the manuscript. Tian-miao WANG helped organize the manuscript. Yi-cheng ZHANG and Tian-miao WANG revised and finalized the paper.

Compliance with ethics guidelines

Tian-miao WANG, Yi-cheng ZHANG, Jian-hong LIANG, Yang CHEN, and Chao-lei WANG declare that they have no conflict of interest.

References

Amidi O, Kanade T, Fujita K, 1999. A visual odometer for autonomous helicopter flight. *Robot Auton Syst*, 28(2):185-193. https://doi.org/10.1016/S0921-8890(99)00016-0

Andersen ED, Taylor CN, 2007. Improving MAV pose estimation using visual information. IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.3745-3750. https://doi.org/10.1109/IROS.2007.4399563

Bay H, Tuytelaars T, van Gool L, 2006. SURF: speeded up robust features. Proc 9th European Conf on Computer Vision, p.404-417. https://doi.org/10.1007/11744023_32

Caballero F, Merino L, Ferruz J, et al., 2006. Improving vision-based planar motion estimation for unmanned aerial vehicles through online mosaicing. Proc IEEE Int Conf on Robotics and Automation, p.2860-2865. https://doi.org/10.1109/ROBOT.2006.1642135

Caballero F, Merino L, Ferruz J, et al., 2007. Homography based Kalman filter for mosaic building. Applications to UAV position estimation. Proc IEEE Int Conf on Robotics and Automation, p.2004-2009. https://doi.org/10.1109/ROBOT.2007.363616

Caballero F, Merino L, Ferruz J, et al., 2009. Unmanned aerial vehicle localization based on monocular vision and online mosaicking. *J Intell Robot Syst*, 55(4-5):323-343. https://doi.org/10.1007/s10846-008-9305-7

Campoy P, Correa JF, Mondragón I, et al., 2009. Computer vision onboard UAVs for civilian tasks. *J Intell Robot Syst*, 54(1-3):105-135. https://doi.org/10.1007/s10846-008-9256-z

Cunningham A, Indelman V, Dellaert F, 2013. DDF-SAM 2.0: consistent distributed smoothing and mapping. Proc IEEE Int Conf on Robotics and Automation, p.5220-5227. https://doi.org/10.1109/ICRA.2013.6631323

Eberli D, Scaramuzza D, Weiss S, et al., 2011. Vision based position control for MAVs using one single circular landmark. *J Intell Robot Syst*, 61(1-4):495-512. https://doi.org/10.1007/s10846-010-9494-8

Forster C, Lynen S, Kneip L, et al., 2013. Collaborative monocular SLAM with multiple micro aerial vehicles. IEEE/RSJ Int Conf on Intelligent Robots and Systems, p.3962-3970.

Fox D, Burgard W, Kruppa H, et al., 2000. A probabilistic approach to collaborative multi-robot localization. *Auton Robot*, 8(3):325-344. https://doi.org/10.1023/A:1008937911390

Howard A, Sukhatme GS, Matarić MJ, 2006. Multirobot simultaneous localization and mapping using manifold representations. *Proc IEEE*, 94(7):1360-1369. https://doi.org/10.1109/JPROC.2006.876922

Ke Y, Sukthankar R, 2004. PCA-SIFT: a more distinctive representation for local image descriptors. Proc IEEE Computer Society Conf on Computer Vision and Pattern Recognition, p.506-513. https://doi.org/10.1109/CVPR.2004.1315206

Li TC, Su JY, Liu W, et al., 2017. Approximate Gaussian conjugacy: parametric recursive filtering under nonlinearity, multimodality, uncertainty, and constraint, and beyond. *Front Inform Technol Electron Eng*, 18(12): 1913-1939. https://doi.org/10.1631/FITEE.1700379

Lowe DG, 2004. Distinctive image features from scale-invariant keypoints. *Int J Comput Vis*, 60(2):91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

Masayoshi M, Chen A, Singh SPN, et al., 2017. Autonomous helicopter tracking and localization using a self-surveying camera array. In: Corke P, Sukkariah S (Eds.). Field and Service Robotics. Springer Tracts in Advanced Robotics. Springer, Berlin, Heidelberg, Germany. https://doi.org/10.1007/978-3-540-33453-8_3

McDonald J, Kaess M, Cadena C, et al., 2011. 6-DOF multi-session visual SLAM using anchor nodes. European Conf on Mobile Robotics, p.69-76.

Meingast M, Geyer C, Sastry S, 2004. Vision based terrain recovery for landing unmanned aerial vehicles. Proc IEEE Conf on Decision and Control, p.1670-1675. https://doi.org/10.1109/cdc.2004.1430284

Moallem P, Razmjooy N, 2012. Optimal threshold computing in automatic image thresholding using adaptive particle swarm optimization. *J Appl Res Technol*, 10(5):703-712. https://doi.org/10.22201/icat.16656423.2012.10.5.361

Mondragón IF, Campoy P, Martinez C, et al., 2010a. Omnidirectional vision applied to unmanned aerial vehicles (UAVs) attitude and heading estimation. *Robot Auton Syst*, 58(6):809-819. https://doi.org/10.1016/j.robot.2010.02.012

Mondragón IF, Olivares-Méndez MA, Campoy P, et al., 2010b. Unmanned aerial vehicles UAVs attitude, height, motion estimation and control using visual systems. *Auton Robot*, 29(1):17-34. https://doi.org/10.1007/s10514-010-9183-2

Montemerlo M, Thrun S, Koller D, et al., 2002. FastSLAM: a factored solution to the simultaneous localization and mapping problem. Proc 8th National Conf on Artificial Intelligence, p.593-598.

Montemerlo M, Thrun S, Roller D, et al., 2003. FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. Proc 18th Int Joint Conf on Artificial Intelligence, p.1151-1156.

Montiel JMM, Civera J, Davison AJ, 2006. Unified inverse depth parametrization for monocular SLAM. Robotics: Science and Systems II, p.81-88. https://doi.org/10.15607/RSS.2006.II.011

Mourikis AI, Trawny N, Roumeliotis SI, et al., 2009. Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Trans Robot*, 25(2):264-280. https://doi.org/10.1109/TRO.2009.2012342

Mur-Artal R, Montiel JMM, Tardós JD, 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans Robot*, 31(5):1147-1163. https://doi.org/10.1109/TRO.2015.2463671

Ready BB, Taylor CN, 2007. Improving accuracy of MAV pose estimation using visual odometry. Proc American Control Conf, p.3721-3726. https://doi.org/10.1109/ACC.2007.4283137

Ready BB, Taylor CN, 2009. Inertially aided visual odometry for miniature air vehicles in GPS-denied environments. *J Intell Robot Syst*, 55(2-3):203-221. https://doi.org/10.1007/s10846-008-9294-6

Rocha R, Dias J, Carvalho A, 2005. Cooperative multi-robot systems: a study of vision-based 3-D mapping using information theory. Proc IEEE Int Conf on Robotics and Automation, p.384-389. https://doi.org/10.1109/ROBOT.2005.1570149

Simo-Serra E, Trulls E, Ferraz L, et al., 2015. Discriminative learning of deep convolutional feature point descriptors. IEEE Int Conf on Computer Vision, p.118-126. https://doi.org/10.1109/ICCV.2015.22

Templeton T, Shim DH, Geyer C, et al., 2007. Autonomous vision-based landing and terrain mapping using an MPC-controlled unmanned rotorcraft. IEEE Int Conf on Robotics and Automation, p.1349-1356. https://doi.org/10.1109/ROBOT.2007.363172

Tsai R, Huang T, Zhu WL, 1982. Estimating three-dimensional motion parameters of a rigid planar patch, II: singular value decomposition. *IEEE Trans Acoust Speech Signal Process*, 30(4):525-534. https://doi.org/10.1109/TASSP.1982.1163931

Vidal-Calleja TA, Berger C, Solà J, et al., 2011. Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. *Robot Auton Syst*, 59(9):654-674. https://doi.org/10.1016/j.robot.2011.05.008

Yi KM, Trulls E, Lepetit V, et al., 2016. LIFT: learned invariant feature transform. Proc 14th European Conf on Computer Vision, p.467-483. https://doi.org/10.1007/978-3-319-46466-4_28

Zhang ZY, 1997. Parameter estimation techniques: a tutorial with application to conic fitting. *Image Vis Comput*, 15(1): 59-76. https://doi.org/10.1016/S0262-8856(96)01112-2