SupSLAM: A Robust Visual Inertial SLAM System Using SuperPoint for Unmanned Aerial Vehicles

Cong-Hoang Quach
VNU University of Engineering and Technology
Hanoi, Vietnam
hoangqc@vnu.edu.vn

Manh Duong Phung
VNU University of Engineering and Technology
Hanoi, Vietnam
duongpm@vnu.edu.vn

Vu-Ha Le
VNU University of Engineering and Technology
Hanoi, Vietnam
halv@vnu.edu.vn

Stuart Perry
University of Technology Sydney
New South Wale, Australia
Stuart.Perry@uts.edu.au

***Abstract* — Simultaneous localization and mapping (SLAM) is essential for unmanned aerial vehicle (UAV) applications since it allows the UAV to estimate not only its position and orientation but also the map of its working environment. We propose in this study a new SLAM system for UAVs named SupSLAM that works with a stereo camera and an inertial measurement unit (IMU). The system includes a front-end that provides an initial estimate of the UAV position and working environment and a back-end that compensates the drift caused by the initial estimate. To improve the accuracy and robustness of the system, we use a new feature extraction method named SuperPoint which includes a pretrained deep neural network to detect key points for estimation. This method is not only accurate in feature extraction but also efficient in computation so that it is relevant to implement on UAVs. We have conducted a number of experiments and comparisons to evaluate the performance of the proposed system. The results show that the system is feasible for UAV SLAM with the performance comparable to state-of-art methods in most scenarios and better in some challenging scenarios.**

Keywords— visual-inertial SLAM, superpoint, unmanned aerial vehicle, smart agriculture

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are expecting to provide efficient solutions for many applications from precision agriculture to smart construction due to their flexibility in operating environments and capability of carrying various types of sensors [1][2][3]. To carry out a task, UAVs typically rely on GPS to locate their position and navigate themselves in the environment. GPS signals however are not always stable due to the large distance from satellites to UAVs and the possibility of being blocked by structures such as plant canopies or buildings. In those scenarios, simultaneous localization and mapping (SLAM) provides an alternative solution by allowing the UAV to not only determine its location but also construct a map of the environment, which is beneficial for many tasks such as crop monitoring in agriculture. Modern visual SLAM algorithms can exploit advancements in computer vision and deep learning to deal with harsh flying conditions. By using only cameras and inertial measurement units (IMU), these approaches provide UAVs with better localization, safer operation, and higher quality data collection at an acceptable cost [4], [5].

Methods of utilizing information from input images in visual SLAM can be classified into direct methods and feature-based methods [12]. The direct methods directly use a whole picture for tracking and mapping. Pixel intensities or depth values can be used as measurements [13], [14]. In contrast, feature-based methods extract geometric features such as interest points and edges to estimate the motion of the camera. Feature-based SLAM thus compresses an image to a set of geometric features that enables the construction of sparse maps. This approach is therefore more suitable for embedded computers on UAVs [15].

Traditional feature-based SLAM operates based on the detection and tracking of interest points and landmarks. A popular technique is the features from accelerated segment test (FAST) which is computationally efficient and suitable for camera localization in fast-moving conditions [16]. This technique is enhanced in [17] with a modified feature descriptor named binary robust independent elementary features (BRIEF) that can improve the overall performance.

In recent years, modern feature-based SLAM that uses machine learning techniques for feature extraction is receiving increasing interest and expecting to outperform the traditional approach in almost all robot working environments [18]–[20]. This approach takes advantage of trained networks to learn key points so that it enhances robustness in data association for visual SLAM. Modern feature-based SLAM is also expected to generate reliable perception graphs for real-time mapping since learning-based feature points are more repeatable and evenly distributed. Besides, these local features can be used as inputs of neural correspondence networks to remove outliers so that the pose estimation becomes more accurate [14], [15].

On another note, SLAM on UAVs often poses constraints on computation due to their limited resource. In this case, feature-based SLAM is more relevant since the extracted interest points can be later used for tasks related to segmentation and object detection. Methods for extracting feature points thus play a key role in SLAM.

In this study, we introduce a visual inertial SLAM method named SupSLAM based on the use of a new feature point called SuperPoint. These feature points are extracted from input images of the SLAM front end through a deep neural network. Our SLAM front end requires only a stereo camera with an IMU as inputs. Pose estimation is undertaken by a multi-state constraint Kalman filter (MSCKF) [22] whereas the trajectory reliability is maintained by a graph optimization process running at the back end. Our contributions are

twofold: (i) leveraging MSCKF by using SuperPoint, and (ii) analyzing the influence of SuperPoint in SLAM by using not only standard datasets but also additional synthetic datasets generated by our toolset.

II. Proposed Method

This section describes our overall SLAM system and the rationale of using SuperPoint to enhance SLAM performance.

A. System overview

SLAM involves the estimate of a UAV position via maintaining a map of the environment. Mathematically, a complete SLAM system can be formulated as a maximum of posterior (MAP), considering the trajectory **X**, landmarks **L** and their data association **D** given measurements **Z** [23]:

$$\hat{X}, \hat{\mathcal{L}}, \hat{D} = arg \max_{X, \mathcal{L}, D} log\, p(Z|X, \mathcal{L}, D) \qquad (1)$$

Data association establishing the relationship between information collected at different time instances is one of the most critical tasks in SLAM. Our approach uses two-step maximization described by equations (2a) and (2b) to solve equation (1).

$$D^{i+1} = arg \max_{D} p(D|X^i, \mathcal{L}^i, Z) \qquad (2a)$$

$$X^{i+1}, \mathcal{L}^{i+1} = arg \max_{X, \mathcal{L}} log\, p(Z|X, \mathcal{L}, D^{i+1}) \qquad (2b)$$

The diagram representing our SLAM system is presented in Fig.1. The system uses a stereo camera as the input to extract features of the environment. This camera is accompanied by an IMU to measure linear acceleration and angular velocity. Data from input devices is then processed via front-end and back-end modules.

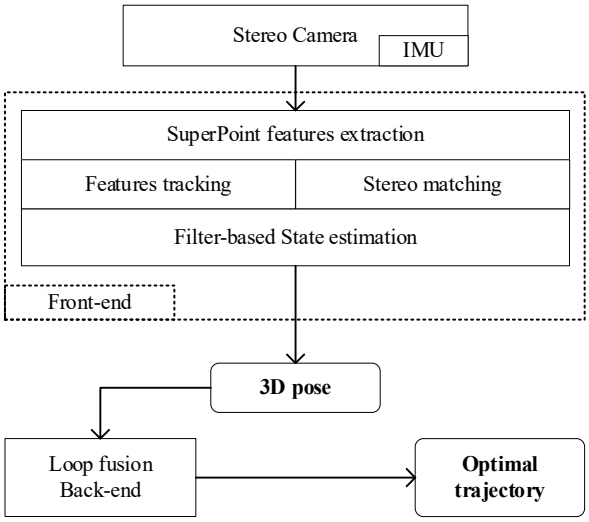


Fig. 1. SLAM algorithms overview

The front-end extracts feature points from the input data and then matches them between the left and right cameras to obtain depth information. This information is then fed to a multi-state constraint Kalman filter (MSCKF) [22] to estimate the 3D pose of the UAV. The back-end on the other hand tracks features in keyframes to carry out a loop closure procedure for map consistency and trajectory estimation. The front-end thus behaves as a visual inertial odometry (VIO) to provide a real-time estimate of the UAV pose whereas the back-end tracks and adjusts the pose over time. In fact, the key for accurate SLAM lies in feature extraction since it provides measurements for other stages. In this work, we use SuperPoint to enhance the detection and distribution of visual features with details being described below.

B. SuperPoint for data association in SLAM

SuperPoint is a fully convolutional neural network that computes 2D feature point locations together with descriptors in a single forward pass and run [18]. In our work, we only consider feature points to reduce computation cost and maintain consistency in the matching results. The architecture of the SuperPoint used is shown in Fig.2. It includes an encoder that maps an input image $I \in R^{W \times H}$ to a tensor $T \in R^{W_c \times H_c \times 65}$ with smaller width $W_c = W/8$ and height $H_c = H/8$, but greater channel depth. The tensor is then fed to a decoder to detect feature points X. The decoder uses convolution layers to extract the response $P \in R^{W_c \times H_c \times 65}$ for feature points which also includes a "no feature point" dustbin. The channel-wise *softmax* is then used to remove the dustbin dimension and the *reshape* function is applied to convert P to the input dimension $W \times H$.

The loss function L for the feature point detector is a convolutional cross-entropy loss computed over the elements $x \in X$. Let $y \in Y$ be the ground-truth feature point, the loss function is computed as:

$$L(X, Y) = \frac{1}{W_c H_c} \sum_{\substack{w=1 \\ h=1}}^{W_c, H_c} l(x_{w,h}, y_{w,h}), \qquad (3)$$

where

$$l(x, y) = -\log\left(\frac{\exp(y)}{\sum_{k=1}^{65} \exp(x_k)}\right).$$

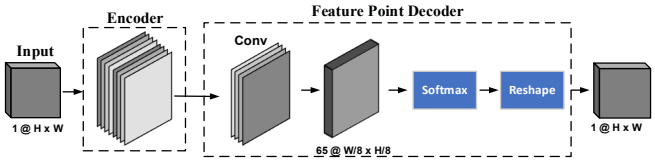


Fig. 2. The model architecture of SuperPoint

C. SuperPoint-based SLAM implementation

In our system, at time step ***t***, a stereo camera with a built-in IMU is used to collect data of the environment through measurement $\mathbf{Z}^t = \{\mathbf{Z_L}^t, \mathbf{Z_R}^t, \mathbf{Z_I}^t\}$, where $\mathbf{Z_I}^t$ is the inertial measurement and $\mathbf{Z_L}^t$ and $\mathbf{Z_R}^t$ are the stereo pair images captured by the camera with resolution $W \times H$. The SuperPoint model is then used to calculate a map of feature points of the left image $\mathbf{Z_L}^t$. After that, we use the non-maximum suppression (NMS) algorithm to select best feature points $\boldsymbol{p_L}^t$ from $\mathbf{Z_L}^t$. The Kanade-Lucas-Tomasi (KLT) feature tracker [24] and random sample and consensus (RANSAC) [25] algorithms are then applied to find correspondent points $\boldsymbol{p_R}^t$ on the right image.

In the tracking stage, the KLT and RANSAC matching methods are also used to track the feature points ($\boldsymbol{p_L}^t$, $\boldsymbol{p_R}^t$), which are being stored in the database, to update their latest positions. The tracking database saves locations of feature points in different viewpoints over time. It is the basis for getting high-quality depth using the 3D re-triangulation calculation, which is important for state estimation. In our implementation, the tracking stage is executed at every frame

whereas the initialization stage is only executed when the number of tracking points goes below 80% of its maximum.

Since the stereo camera model including the baseline, intrinsic, and extrinsic parameters is known, 3D poses can be estimated directly from the given stereo tracking data. However, measurement errors from the stereo camera in different poses may cause mismatches in the 3D transform. To overcome that problem, we use an extended version of MSCKF [26] as the 3D pose estimator. The filter propagates all sensor data and incorporates all tracking features in a sliding window time to improve the estimation.

In addition, we adopt a loop fusion technique as in [27] to refine the MSCKF estimator so that it can compensate for pose drifts caused by the estimation errors accumulated over time. Initially, the loop fusion constructs a pose graph in which each node is current left image with the present 3D pose given by the MSCKF. After that, the loop fusion will add a new node if there has a significant change of the current pose (in translation and rotation) to the latest keyframe. When the UAV revisits a previously known location, the loop-closure detector will recognize similarities in its features to activate the pose-graph optimization procedure. The loop closure thus works in a loosely coupled manner with the front-end in the sense that it improves the final estimation result, but that result is not fed back to the MSCKF of the front-end.

III. RESULTS

To evaluate the performance of the proposed SLAM system, we have conducted experiments with details as follows.

A. Experimental setup

The data for experiments is collected from a quadcopter with the frame size of 40 cm x 40 cm. The IMU has an update rate of 200 Hz. The stereo camera has the baseline of 7 cm, resolution of 752x480, and speed of 20 frames per second.

The SLAM system is implemented in C++ using the OpenVINS framework [26]. OpenCV and LibTorch libraries are used for image processing. The number of visual features being tracked is limited to 400 for real-time performance, and new feature points are added when the number of active points drops to under 300. A pretrained deep neural network is used to extract features from input images at the resolution of 752x480. The time window in MSCKF is set to 3 seconds. In the back-end, a new keyframe is added to the pose graph every 1.2 m.

B. Data preparation

Both real and synthetic data is used in experiments. The real data includes three most complex sequences named MH3, MH4, and MH5 of the EuRoC dataset in which the drone flies along a machine hall [29]. The synthetic data includes two scenarios representing a rural farm and an urban area generated by our toolset developed from AirSim [28] as shown in Fig. 3. The rural farm simulates a rural environment with uneven terrain, a variety of plants, and agricultural equipment. The urban area presents a flat area with different features from cars, trees, and living premises. The drone flies at the speed of 5 m/s and altitude of 5 m along a rectangular trajectory to form a dataset named LoopF. Similarly, it flies at the same speed and altitude along other trajectories above the urban area to form two datasets named Loop1 and Loop2. In these scenarios, we use the drone configuration similar to the system used for the EuRoC dataset, i.e., a 20 Hz stereo camera and a 200 Hz IMU. However, our datasets are more challenging since the drone travels longer distances at a higher speed in environments with less trackable landmarks.

Fig. 3. The synthetic scenarios of a rural farm and an urban area used for experiments

C. Evaluation metrics

We use two metrics for performance evaluation including the absolute trajectory error (ATE) and relative trajectory error (RTE) [30]. The ATE is computed by first aligning the estimated trajectory to the ground truth and then measuring the difference between them as illustrated in Fig.4a. The RTE is computed by dividing the estimated trajectory into segments $\mathbf{d}_k$ and then aligning each segment to the ground truth trajectory to calculate the error as illustrated in Fig.4b.

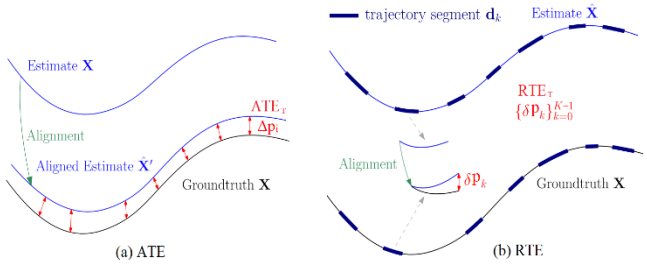


Fig. 4. Illustration of evaluation metrics. (a) Absolute trajectory error and (b) Relative trajectory error.

If only the translational errors are considered, the calculation of RTE and ATE is given by:

$$ATE_T = \left(\frac{1}{N}\sum_{i=0}^{N-1}\|\Delta p_i\|^2\right)^{\frac{1}{2}} \quad (3a)$$

$$RTE_T = \{\delta p_k\}_k^{K-1} \quad (3b)$$

where

- N is the number of estimated poses in the trajectory.
- Δp is distance between full-aligned estimated pose and its correspondence ground truth.
- δp is distance between segment-aligned estimated pose and its correspondence ground truth.

Since the ATE is sensitive to the time the error occurs, it is suitable for evaluating the performance of full SLAM systems whereas the RTE is better in measuring the drift of VO systems.

D. SLAM results

Figure 5 shows the feature points detected by SuperPoint in three scenarios including the urban area, rural farm, and machine hall with two different setups, 100 and 400 feature points. It can be seen that features such as corners, edges, color changes, etc., are well detected. In addition, the detected features are spread across the images so that the algorithm is less dependent on certain objects.

(a) Urban scenario with 100 feature points

(b) Urban scenario with 400 feature points

(c) Rural scenario with 100 feature points

(d) Rural scenario with 400 feature points

(e) Machine hall scenario with 100 feature points

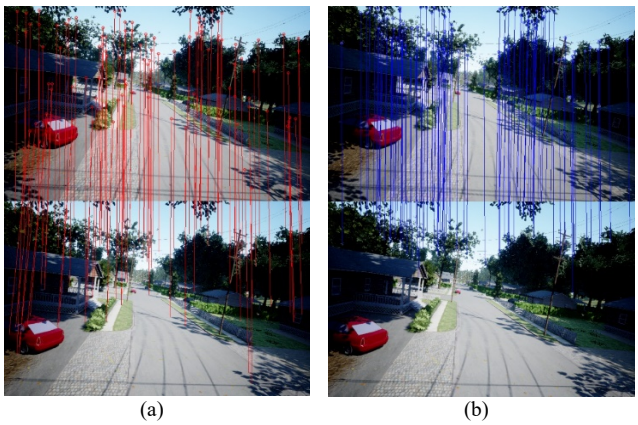
(f) Machine hall scenario with 400 feature points

Fig. 5. Feature points detected by Superpoint.

Figure 6 shows the correspondence of the detected feature points between the left and right images. It can be seen that most features points are properly matched implying the extracted depth information is reliable.

Fig. 6. Feature points detected by Superpoint.

The tracking of feature points between two image frames captured at different time $\Delta t = 0.3$ s is shown in Fig.7a. Since the detected feature points are spread across the image, the number of common feature points between the frames are well maintained, which is important for stable SLAM.



(a) (b)

Fig. 7. The tracking results with (a) FAST and (b) SuperPoint in two frames captured with the time difference $\Delta t = 0.3$ s.

Figure 8 shows the SLAM result for the LoopF dataset in which the UAV flights two rounds above the rural farm. It can be seen that the final estimated trajectory (red line) tracks well the ground truth trajectory (yellow line) even over a long distance of nearly 500 m. In fact, the estimation errors decrease over time since more feature points are detected and the environment map is built as shown in Fig.9.

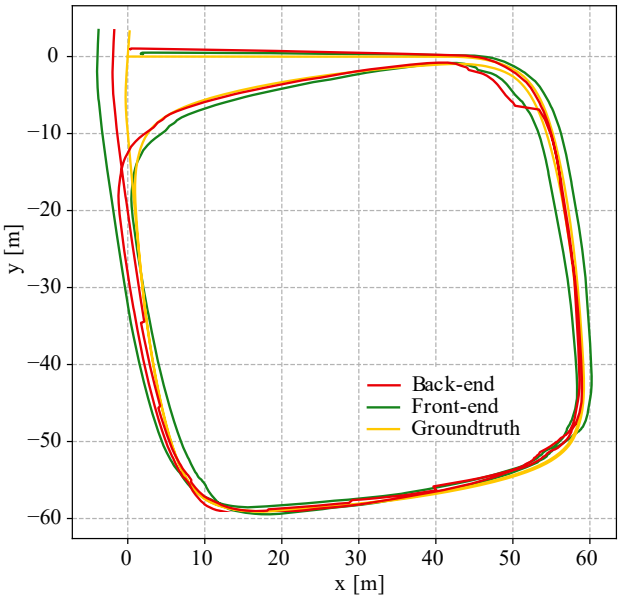


Fig. 8. Trajectories of the UAV for the LoopF dataset including the ground truth trajectory (yellow line), front-end estimation (green line), and back-end estimation (red line).

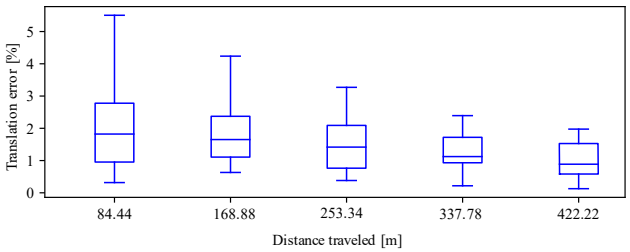


Fig. 9. Estimation errors with respect to the traveling distance for the LoopF dataset.

E. Comparison results

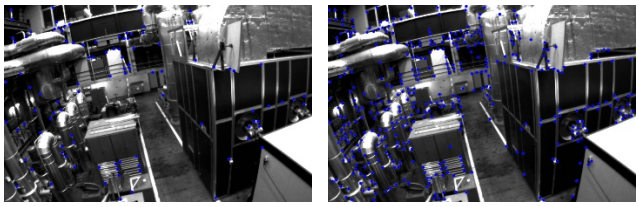
To further evaluate the performance of our method, we have conducted comparisons with OpenVINS [26], a state-of-art SLAM system that uses FAST for feature extraction. Figure 10 shows the feature points detected by FAST. Unlike SuperPoint, these feature points are concentrated around certain objects such as plants, houses, or machines. Consequently, the number of common feature points is significantly decreased between the image frames taken at different time when objects are moving out of the scene as can be seen in Fig.7b. That problem in turn would affect the SLAM results.

(a) Urban scenario with 100 feature points

(b) Urban scenario with 400 feature points

(c) Rural scenario with 100 feature points

(d) Rural scenario with 400 feature points



(e) Machine hall scenario with 100 feature points

(f) Machine hall scenario with 400 feature points

Fig. 10. Feature points detected by FAST.

Table I shows the SLAM performance of the front-end on EuRoC and the synthetic datasets. It shows that our method is better in EuRoC whereas OpenVINS is better in other datasets. However, if we look closer to the difference between the two methods, their performance is in fact similar since the differences are relatively small. Those differences are only significant for challenging datasets such as Loop1 and Loop2 as shown in Table II where SupSLAM clearly outperforms OpenVINS. This result is further confirmed in Fig.11 that compares the SLAM performance in the front-end and back-end of the two methods.

TABLE I. PERFORMANCE COMPARISON IN SLAM FRONT-END

DATA	OpenVINS (FAST)		SupSLAM (SuperPoint)	
	RTE_T (%)	**ATE_T (m)**	**RTE_T (%)**	**ATE_T (m)**
MH3	0.32	0.1	**0.28**	**0.09**
MH4	0.96	0.25	**0.68**	**0.23**
MH5	0.8	0.18	**0.57**	**0.18**
LoopF	**1.49**	1.77	1.74	**1.23**
Loop1	**7.6**	**20.3**	7.7	20.8
Loop2	**8.01**	39.4	8.03	**26.8**

TABLE II. RTE_T PERFORMANCE COMPARISON IN SLAM BACK-END

	OpenVINS (FAST)	SupSLAM (SuperPoint)
Loop1	4.50%	**2.45%**
Loop2	2.78%	**1.54%**

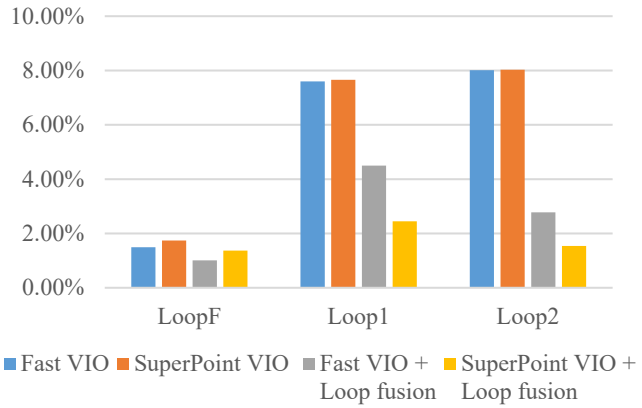


Fig. 11. Comparison of RTE_T without and with SLAM back-end on the synthetic datasets

On another note, Table III shows the standard deviation of RTE_T during UAV movement in the rural farm. It shows that SupSLAM is more stable in most parts of the trajectories due to the even distribution of the feature points detected by SuperPoint.

TABLE III. STANDARD DEVIATION OF RTE_T ON THE SYNTHETIC DATA

Trajectory in percent		**20%**	**40%**	**60%**	**80%**	**100%**
Loop1	OpenVINS	3.36	2.28	3.01	1.37	0.81
	SupSLAM	**2.39**	**2.18**	**2.84**	1.63	**0.87**
Loop2	OpenVINS	3.36	2.14	3.1	1.53	1.78
	SupSLAM	**3.25**	2.19	**1.95**	**1.08**	**1.27**

IV. CONCLUSION

In this work, we have proposed a visual inertial SLAM system named SupSLAM for UAVs. The system features SuperPoint as the method to extract features from the images captured by a stereo camera. Compared to other feature extraction methods, SuperPoint provides a better distribution of feature points and thus enhances the tracking performance in fast-moving scenarios. Since SuperPoint is a pretrained network, it also requires less computational resources and thus is suitable for UAVs which are limited in their payload and battery capacity. To evaluate the performance of SupSLAM, we have conducted a number of experiments and comparisons with both real and synthetic datasets. The results confirm the effectiveness and validity of our proposed system for UAVs. In future work, we aim to leverage SuperPoint to define more efficient data association methods to further improve the accuracy and robustness of the current system.

ACKNOWLEDGMENT

Quach Cong Hoang was funded by Vingroup Joint Stock Company and supported by the Domestic Ph.D. Scholarship Programme of Vingroup Innovation Foundation (VINIF), Vingroup Big Data Institute (VINBIGDATA), code VinIF 2020. TS.23

REFERENCES

[1] D. C. Tsouros, A. Triantafyllou, S. Bibi, and P. G. Sarigannidis, "Data acquisition and analysis methods in UAV- based applications for precision agriculture," *Proc. - 15th Annu. Int. Conf. Distrib. Comput. Sens. Syst. DCOSS 2019*, pp. 377–384, 2019, doi: 10.1109/DCOSS.2019.00080.

[2] M. D. Phung, C. H. Quach, T. H. Dinh, Q. Ha, "Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection," Automation in Construction, vol. 81, pp. 25-33, 2017, doi: 10.1016/j.autcon.2017.04.013.

[3] V. T. Hoang, M. D. Phung, T. H. Dinh and Q. P. Ha, "System Architecture for Real-Time Surface Inspection Using Multiple UAVs," IEEE Systems Journal, vol. 14, no. 2, pp. 2925-2936, 2020, doi: 10.1109/JSYST.2019.2922290..

[4] X. Liu *et al.*, "Robust Fruit Counting: Combining Deep Learning, Tracking, and Structure from Motion," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1045–1052, doi: 10.1109/IROS.2018.8594239.

[5] I. Sa *et al.*, "Build your own visual-inertial drone: A cost-effective and open-source autonomous drone," *IEEE Robot. Autom. Mag.*, vol. 25, no. 1, pp. 89–103, 2018, doi: 10.1109/MRA.2017.2771326.

[6] C. Cadena *et al.*, "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Jun. 2016, doi: 10.1109/TRO.2016.2624754.

[7] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006, doi: 10.1109/MRA.2006.1638022.

[8] B. Huang, J. Zhao, and J. Liu, "A Survey of Simultaneous Localization and Mapping," pp. 1–15, 2019, [Online]. Available: http://arxiv.org/abs/1909.05214.

[9] D. Scaramuzza and F. Fraundorfer, "Tutorial: Visual odometry," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, 2011, doi: 10.1109/MRA.2011.943233.

[10] D. Scaramuzza and Z. Zhang, "Visual-Inertial Odometry of Aerial Robots," in *Encyclopedia of Robotics, Springer, 2019*, 2019, pp. 1–13.

[11] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual--Inertial Odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017, doi: 10.1109/TRO.2016.2597321.

[12] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, no. 1, 2017, doi: 10.1186/s41074-017-0027-2.

[13] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, 2018, doi: 10.1109/TPAMI.2017.2658577.

[14] H. Matsuki, L. von Stumberg, V. Usenko, J. Stuckler, and D. Cremers, "Omnidirectional DSO: Direct Sparse Odometry With Fisheye Cameras," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3693–3700, Oct. 2018, doi: 10.1109/LRA.2018.2855443.

[15] J. Delmerico and D. Scaramuzza, "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2502–2509, doi: 10.1109/ICRA.2018.8460664.

[16] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3951 LNCS, pp. 430–443, 2006, doi: 10.1007/11744023_34.

[17] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, 2017, doi: 10.1109/TRO.2017.2705103.

[18] D. Detone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 2018-June, pp. 337–349, 2018, doi: 10.1109/CVPRW.2018.00060.

[19] J. Tang, L. Ericson, J. Folkesson, and P. Jensfelt, "GCNv2: Efficient Correspondence Prediction for Real-Time SLAM," *IEEE Robot. Autom. Lett.*, pp. 1–1, Feb. 2019, doi: 10.1109/LRA.2019.2927954.

[20] P. E. Sarlin, D. Detone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning Feature Matching with Graph Neural Networks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4937–4946, 2020, doi: 10.1109/CVPR42600.2020.00499.

[21] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 61–76, 2012, doi: 10.1109/TRO.2011.2170332.

[22] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 3565–3572, 2007, doi: 10.1109/ROBOT.2007.364024.

[23] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic SLAM," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1722–1729, 2017, doi: 10.1109/ICRA.2017.7989203.

[24] B. D. Lucas and T. Kanade, "Iterative Image Registration Technique With an Application To Stereo Vision.," *Proc. Imaging Underst. Work. pp. 121-130*, vol. 2, pp. 674–679, 1981.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981, doi: 10.1145/358669.358692.

[26] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A Research Platform for Visual-Inertial Estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 4666–4672, doi: 10.1109/ICRA40945.2020.9196524.

[27] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, 2018, doi: 10.1109/TRO.2018.2853729.

[28] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," pp. 1–14, 2017, [Online]. Available: http://arxiv.org/abs/1705.05065.

[29] M. Burri *et al.*, "The EuRoC micro aerial vehicle datasets," *Int. J. Rob. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016, doi: 10.1177/0278364915620033.

[30] Z. Zhang and D. Scaramuzza, "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 7244–7251, 2018, doi: 10.1109/IROS.2018.8593941.