Hindawi
Discrete Dynamics in Nature and Society
Volume 2020, Article ID 1632986, 6 pages
https://doi.org/10.1155/2020/1632986

Research Article

Improving Processing Time for the Location Algorithm of Robots

Jing Chen and Liwen Chen

School of Information Science and Engineering, Fujian University of Technology, No. 33 Xueyuan Road University Town, Fuzhou, Fujian, China

Correspondence should be addressed to Liwen Chen; chenlw2002@163.com

Received 10 June 2020; Accepted 17 July 2020; Published 2 September 2020

Guest Editor: Chi-Hua Chen

The paper proposes an algorithm based on the Multi-State Constraint Kalman Filter (MSCKF) algorithm to construct the map for robots special in the poor GPS signal environment. We can calculate the position of the robots with the data collected by inertial measurement unit and the features extracted by the camera with MSCKF algorithm in a tight couple way. The paper focuses on the way of optimizing the position because we adopt it to compute Kalman gain for updating the state of robots. In order to reduce the processing time, we design a novel fast Gauss–Newton MSCKF algorithm to complete the nonlinear optimization. Compared with the performance of conventional MSCKF algorithm, the novel fast-location algorithm can reduce the processing time with the kitti datasets.

1. Introduction

The odometers of robots can divide into two parts: motion propagation and state updating end [1, 2]. The motion propagation is computing the motion matrix with the data of the sensors. Then, the result updates through the updating end. Many robots adopt the cameras to construct visual odometer because the cameras are conveniently mounted. [3] The visual odometers extract the matching features from the video frames to propagate the motion matrix. However, there are many drawbacks at the visual odometer such as the long processing time and the error because of mismatching features. In order to reduce the processing time, some researchers applied two threads to compute the motion equation and update the result, respectively [4, 5]. However, it will increase the hardware cost of the location systems by this way. In order to improve the processing time without additional hardware, Chansoo used the corner features to replace the ordinary features [6]. However, the information of the features sometimes will lose to make it hard to compute the location of robots. Moreover, it is difficult to extract the corner features in the texture flaw environment. In this way, the error of the system becomes big. In order to reduce the processing time and improve accuracy of visual odometer at the same time, more and more researchers began to adopt different sensors to compensate for the drawbacks of the visual odometers [7–12]. GPS can get the accurate location of the robot in the outdoor environment. Inertial measurement unit (IMU) can compensate for the error in the poor GPS signal environment [13, 14]. For the application of the robot in the poor GPS signal environment, many researchers adopted IMU to construct the visual inertial odometer (VIO) [15–19]. In order to compute the motion matrix and construct the map of robots by the VIO, we adopt the MSCKF algorithm. The motion propagation of MSCKF algorithm use the data of IMU to propagate the motion matrix. The update-end of MSCKF algorithm used the observed constraint features to update the motion matrix under the extended Kalman filter (EKF) framework [16, 20, 21]. However, the performance of conventional MSCKF about the processing time and the accuracy is bad. Ramezani et al. expanded the range of the observed features to improve the accuracy of the MSCKF algorithm [22]. Li et al. constructed the model of the error for the gravity accelerator and the bias error of the IMU to improve the accuracy of the MSCKF algorithm [23, 24]. Some researchers add the close-loop detection to update the initial point of the robot [24, 25]. Those methods improved the accuracy of the

system but increased the processing time [25]. In order to improve the processing time, Eckenhoff et al. proposed the online verification to reduce the update time and improve the accuracy of the MSCKF algorithm although it will increase the hardware cost [26]. Sejong Heo and others applied the slide windows to replace the ordinary method of extracting the features of MSCKF algorithm to reduce the computation amount and improve the processing time [17, 27, 28]. However, the processing time did not significantly reduce. Some researchers focused on the improvement of the back end of MSCKF algorithm to improve the processing time because the update end of the MSCKF algorithm based on the EKF filter will cost much process time. Some researchers applied the Cubature Kalman Filters (CKF) and Maximum Likelihood Estimate (MLE) method to replace the EKF although the performance of the processing time still is not ideal [29, 30]. Alibay et al. applied the stage Random Sample Consensus (RANSAC) method to propagate the motion matrix to meet the requirement of real time [31]. Nikolic applied Field Programmable Gate Array (FPGA) to complete the preprocessing the data of IMU to reduce the processing time [32]. However, those methods will increase the hardware cost.

In order to reduce the processing time, we design the novel fast Gauss–Newton MSCKF algorithm without additional hardware cost. The key point of our work is calculating the position and the attitude optimized by the novel nonlinear method. Then, we use the result to compute the Kalman gain in EKF framework.

2. Related Work

MSCKF algorithm is based on the EKF architecture. The algorithm is composed of three parts: propagation, augmentation, and update [16]. We use the state of IMU to propagate the state of robots at the state propagation. The propagation equation is as follows [16]:

$$\dot{\widetilde{X}}_{\text{imu}} = F\widetilde{X}_{\text{imu}} + \text{Gn}_{\text{imu}}, \tag{1}$$

where F means the coefficient matrix of the IMU state matrix that calculated by the accelerator matrix of IMU and the gyroscope data matrix and G means the coefficient matrix of the noise matrix. Then, we compute the Jacobin matrix to complete state augment. We define the matrix of the state of the No. j frame as the symbol $P_{j|j}$. The matrix is constructed of the covariance matrix of the state of the IMU ($P_{\text{II}_{j|j}}$), the covariance matrix between IMU and the camera ($P_{\text{IC}_{j|j}}$), and the covariance matrix of the cameras ($P_{\text{CC}_{j|j}}$):

$$P_{j+1|j} = \begin{bmatrix} P_{\text{II}_{j|j}} & P_{\text{IC}_{j|j}}J^T \\ \text{JP}_{\text{IC}_{j|j}}^T & \text{JP}_{\text{CC}_{j|j}}J^T \end{bmatrix}. \tag{2}$$

Considering the association between the features, MSCKF algorithm updates the propagation by the collection of the observed feature. At the same time, the MSCKF algorithm eliminates the features that are unobserved and worthless. Then, we can compute the reprojection errors of the observed features with nonlinear optimization to update the state matrix. Now, the observation equation denotes as follows:

$$r = H\widetilde{X} + \text{noise}, \tag{3}$$

where r means the residual error of the features, H denotes the Jacobin matrix, and $\widetilde{X}$ denotes the state propagated through the above motion equation. Now, we need to compute the matrix H to get the Kalman gain. The residual error can compute by the difference between the real position of the jth feature and the projected value of the position. The process denotes as follows:

$$\text{r}_i^{(j)} = z_i^j - \widehat{z}_i^j. \tag{4}$$

In this way, $\widehat{z}_i^j$ denotes the estimated position of the No. j feature observed by the No. i camera. z_i^j means the position of the No. j feature. In order to reduce the amount of computation, it does QR decomposition of $H_x^{(j)}$ to accelerate the solution of equations. Then, we get the upper triangular matrix T_H that is nonsingular matrix:

$$H_o = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} T_H \\ 0 \end{bmatrix}. \tag{5}$$

Now, we can use T_H to get the Kalman gain:

$$K = \text{PT}_H^T \left(T_H \text{PT}_H^T + R_{\text{n}}\right)^{-1}, \tag{6}$$

where P means the state matrix of the camera and R_n means the noise matrix. The state covariance of No. $j+1$ frame can be updated as follows:

$$P_{j+1|j+1} = \left(I_{6N+15} - \text{KT}_H\right) P_{j+1|j} \left(I_{6N+15} - \text{KT}_H\right)^T + \text{KR}_n K^T. \tag{7}$$

3. Our Method

The value of $\widehat{z}_i^j$ determines the result of the Kalman gain K in equation 6. Calculating the $\widehat{z}_i^j$ costs the most processing time, the conventional method adopts the Gauss–Newton optimization method to compute the value of ${}^G\widehat{p}_{f_j}$ that used to calculate $\widehat{z}_i^j$. In order to reduce the processing time, we employ the fast Gauss–Newton method to make the nonlinear optimization. The whole process of optimization is divided into two stages.

At the first stage, the difference is so big that we use the gradient descent method to optimize. We make the Taylor expansion of the equation as follows:

$$f(x + \Delta x) \approx f(x) + J(x)\Delta x. \tag{8}$$

We can get the gradient descent of Δx is $J(x)$. However, the gradient descent method may cause the diverged result because iterative step size is too big.

At the second stage, we use the Gauss–Newton method to get Δx when the optimized value is very close to the prior value to keep the convergence of algorithm. The objection of Gauss–Newton algorithm is to minimize $\|f(x + \Delta x)\|^2$.

So, we get the solution of the derivation. We get the Gauss Newton equation:

$$J(x)^T J(x)\Delta x = -J(x)^T f(x). \tag{9}$$

Then, we can compute the value of Δx as follows:

$$\Delta x = -\left(J(x)^T J(x)\right)^{-1} J(x)^T f(x). \tag{10}$$

In order to compute the real value of motion matrix, we bring the prior value of the coordinate into the above equations. If we describe the prior-estimated value of position of the No. j frame $({}^{C_i}\widehat{p}_{f_j})$ as follows:

$${}^{C_i}\widehat{p}_{f_j} = C\left({}^{C_i}_{C_n}\widehat{\overline{q}}\right)\begin{bmatrix} {}^{C_n}\widehat{X}_j \\ {}^{C_n}\widehat{Y}_j \\ {}^{C_n}\widehat{Z}_j \end{bmatrix} + {}^{C_i}p_{c_n}, \tag{11}$$

where ${}^{C_i}\widehat{p}_{f_j}$ means the prior-estimated value of the position of the jth frame and $C({}^{C_i}_{C_n}\widehat{\overline{q}})$ means the matrix of rotation quaternion We can change the above equation by normalization as follows:

$${}^{C_i}\widehat{p}_{f_j} = {}^{C_n}\widehat{Z}_j \left(C\left({}^{C_i}_{C_n}\widehat{\overline{q}}\right)\begin{bmatrix} \dfrac{{}^{C_n}\widehat{X}_j}{{}^{C_n}\widehat{Z}_j} \\ \dfrac{{}^{C_n}\widehat{Y}_j}{{}^{C_n}\widehat{Z}_j} \\ 1 \end{bmatrix} + \frac{1}{{}^{C_n}\widehat{Z}_j}\,{}^{C_i}p_{C_n} \right). \tag{12}$$

In order to make clearer of the expression, we describe as follows:

$$\begin{aligned} \widehat{\alpha}_j &= \frac{{}^{C_n}\widehat{X}_j}{{}^{C_n}\widehat{Z}_j}, \\ \beta_j &= \frac{{}^{C_n}\widehat{Y}_j}{{}^{C_n}\widehat{Z}_j}, \\ \widehat{\rho}_j &= \frac{1}{{}^{C_n}\widehat{Z}_j}. \end{aligned} \tag{13}$$

Now, we can update ${}^{C_i}\widehat{p}_{f_j}$ as follows:

$${}^{C_i}\widehat{p}_{f_j} = {}^{C_n}Z_j \begin{bmatrix} f_{i1}\left(\widehat{\alpha}_j, \widehat{\beta}_j, \widehat{\rho}_j\right) \\ f_{i2}\left(\widehat{\alpha}_j, \widehat{\beta}_j, \widehat{\rho}_j\right) \\ f_{i3}\left(\widehat{\alpha}_j, \widehat{\beta}_j, \widehat{\rho}_j\right) \end{bmatrix}. \tag{14}$$

Now, we update the error between the estimated value and the prior value as follows:

$$\begin{aligned} e(\widehat{\sigma}) &= \widehat{z}_i^{(j)} - \frac{1}{f_{i3}(\widehat{\sigma})}\begin{bmatrix} f_{i1}(\widehat{\sigma}) \\ f_{i2}(\widehat{\sigma}) \end{bmatrix}, \\ \widehat{\sigma} &= \begin{bmatrix} \widehat{\alpha}_i \\ \widehat{\beta}_i \\ \widehat{\rho}_j \end{bmatrix}. \end{aligned} \tag{15}$$

The prior value of $\widehat{z}_i^{(j)}$ is normalization value in this. We can use J_k to stand for the effect of the iteration. We describe J_k as follows:

$$J_k = 0.5 * e(\widehat{\sigma})^T R^{-1} e(\widehat{\sigma}), \quad R = \left\{R^1, \ldots, R^n\right\}. \tag{16}$$

In the above equation, R denotes the weight of noise which includes the bias of gyroscope and the bias of accelerator velocity. In order to reflect the effect of the derivation, we describe the difference between the prior value and the optimized value as η:

$$\eta = \frac{\left|J_{k+1} - J_k\right|}{J_k}. \tag{17}$$

At the first stage, when η is bigger than the threshold value μ, we choose gradient descent as the value of Δx. By the way, the value of μ is the empirical value as 0.5. Now, we describe the step of the derivate δy as follows:

$$\delta y = -J(x)^T e(\widehat{y}), \quad \eta > \mu, \tag{18}$$

where $J(x)$ means the Jacobian matrix of the error $e(\widehat{y})$. At the second stage, when the optimized value is close to the real ones that means the value of η is less than μ. In order to guaranteed convergence of matrix, we choose Gauss–Newton algorithm to complete nonlinear optimization. The step of Δx denotes as follows:

$$\delta y = \left(J^T R^{-1} J\right)^{-1}\left(-J^T R^{-1}\right) e(\widehat{y}), \quad 0.01 < \eta <= \mu. \tag{19}$$

If η is smaller than 0.01, the algorithm converges, and we can stop iteration.

4. Experiment

The experiment of the paper based on kitti datasets. The preprocess of the datasets is extracting the features and checking the external parameters of the cameras [33]. In the processing of the computation, we choose the value of the bias of acceleration and the gyroscope as 10^{-6}. We also choose the same value as the noise of the accelerator and the gyroscope. The result are shown in Table 1.

There are 98 frames of No. 1 datasets at Table 1. Compare the processing time of the conventional MSCKF algorithm, and we found that the processing time decreased by 20.7% when we adopt the novel fast Gauss–Newton MSCKF algorithm. Then, we calculate the final errors by the difference between the calculated result by different algorithms and the real value. Comparing the final errors of the MSCKF algorithm from the IMU data alone, the final error of conventional MSCKF decreases by 56.5%. The final error of the novel fast Gauss–Newton MSCKF decreases by 55.1%. The

Table 1: The performance of the novel MSCKF algorithm with No.1 datasets.

	Process time (s)	Final error (m)
Conventional MSCKF with the data of IMU alone	19.02	2.32
Conventional MSCKF with the data of IMU and the camera	19.02	1.01
Fast Gauss–Newton MSCKF with the data of IMU and the camera	15.09	1.04

Table 2: The performance of improved MSCKF algorithm with No. 2 datasets.

	Process time (s)	Final error (m)
Conventional MSCKF with the data of IMU alone	58.80	2.29
Conventional MSCKF with the data of IMU and the camera	58.80	3.32
Fast Gauss–Newton MSCKF with the data of IMU and the camera	55.30	3.52

final error calculated by the novel MSCKF algorithm increases by 3.0% than the one calculated by the conventional MSCKF algorithm.

There are 239 frames of No. 2 datasets at Table 2. Comparing the processing time of conventional MSCKF algorithm, the processing time of the novel fast Gauss–Newton MSCKF decreased by 6.0%. Comparing the final error of conventional MSCKF with the data of IMU alone, the final error of conventional MSCKF increased by 55.0%. The final error of the novel Gauss–Newton MSCKF algorithm increased by 6.0% relative to the final error of the conventional MSCKF algorithm.

Then, we can compare the trajectory computed by the conventional MSCKF algorithm with the one computed by the novel MSCKF algorithm. In order to make clear the result, we also compare the trajectory computed by the conventional MSCKF using the data of IMU alone with the global truth trajectory.

There are the trajectories calculated by different algorithms. The blue trajectory calculated by the conventional MSCKF algorithm is shown in Figure 1. The blue one of Figure 2 means the trajectory calculated by the novel Gauss–Newton MSCKF algorithm. There are red trajectories calculated by the conventional MSCKF with the data IMU alone at both figures. There are the global truth trajectories in green color at both figures. We can get the conclusion that the blue trajectory is closer to the red trajectory than the green trajectory in Figure 1. It means the conventional MSCKF algorithm can fuse the data from IMU and the stereo camera to reduce the error. The blue trajectory is also closer to the green trajectory than the red trajectory in Figure 2. It means the novel MSCKF algorithm is also able to fuse the data of IMU and the features of the cameras to reduce the error calculated by the MSCKF with the data of the IMU alone. When we compared the blue trajectories at both figures, the blue trajectory of Figure 2 is closer to the red one than the blue trajectory of Figure 1 at the beginning. However, the error of the novel algorithm becomes bigger when the distance becomes longer at Figure 2. We also notice that there is a sudden change in the middle of the trajectory in Figure 2 because there is the switching between two optimization methods.

We also calculate the trajectories by the above algorithms with No. 2 datasets to test the performance of the novel MSCKF algorithm when the VIO runs longer. The red

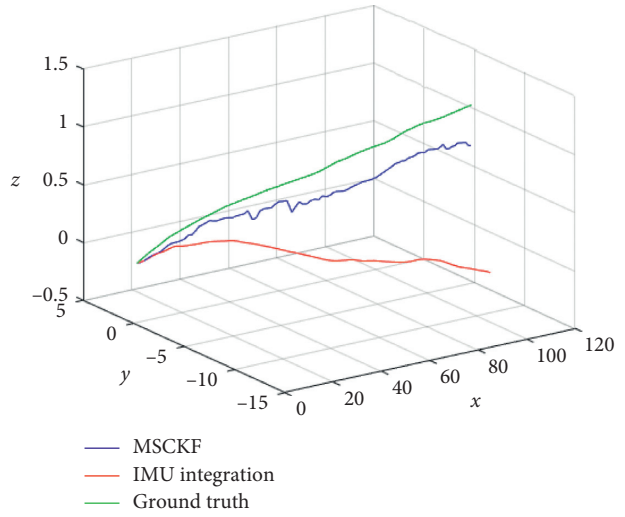

Figure 1: The trajectories by conventional MSCKF algorithm with No. 1 datasets.

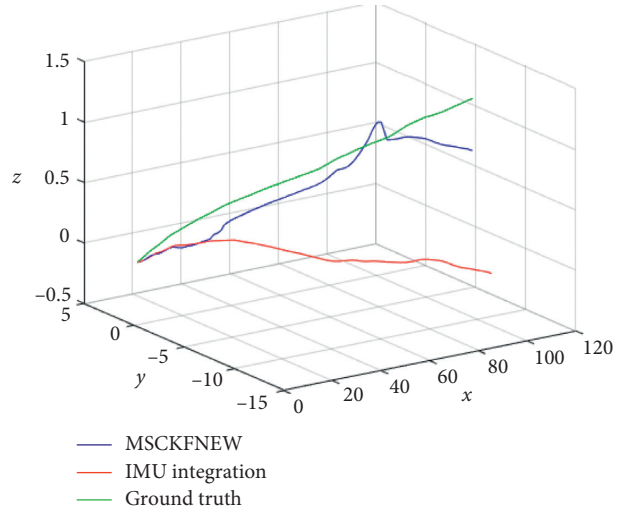

Figure 2: The trajectories by novel MSCKF algorithm with No.1 datasets.

trajectories means the ones calculated by the conventional MSCKF algorithm with the IMU data alone under No. 2 datasets in Figure 3 and 4. The green trajectories means the

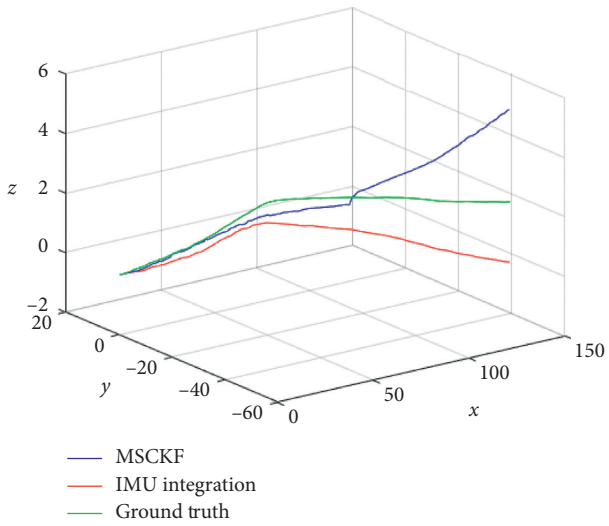

Figure 3: The trajectories of MSCKF with No. 2 datasets.

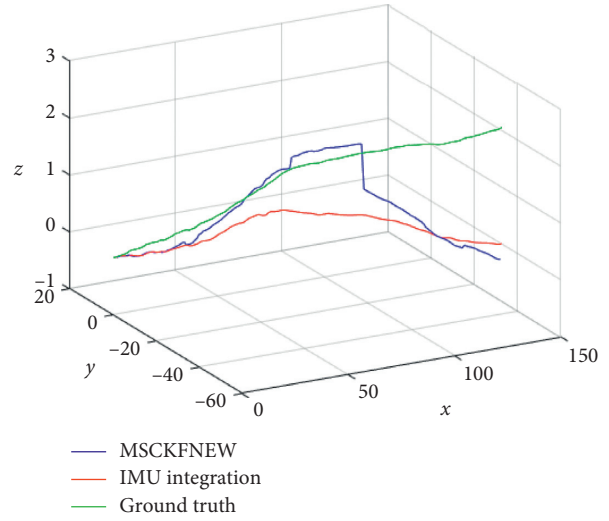

Figure 4: The trajectories of improved MSCKF with No. 2 datasets.

ground truth trajectory in Figures 3 and 4. The blue trajectory of Figure 3 means the trajectory calculated by the conventional MSCKF algorithm with No. 2 datasets. The blue one of Figure 4 means the trajectory calculated by the novel Gauss–Newton MSCKF algorithm with No. 2 datasets. We notice that the blue trajectory at Figure 3 is closer to the green one at first. However, the blue trajectory becomes deviated from the red trajectory when the distance becomes longer at Figure 3. It means the error of novel MSCKF algorithm becomes bigger when the VO runs longer. The blue trajectory becomes closer to the red trajectory at Figure 4 at first. However, when the distance x is between 50 m and 100 m, the blue trajectory is closer to the green one. It means the error computed by the novel MSCKF decrease when the computation continues. When the distance x is bigger than 100 m the error of blue trajectory became bigger. Nevertheless, there is more obvious hip change than the blue trajectory in Figure 2.

5. Conclusion

Comparing the performances of the novel MSCKF algorithm with the conventional MSCKF algorithm, the processing time of the novel MSCKF algorithm decreases by 20.7%~6.0%. The final error calculated by the novel MSCKF algorithm increases by 2.0%~6.0% than the one calculated by the conventional MSCKF algorithm. We can get the conclusion that the novel MSCKF can reduce the processing time. However, the final error computed by the fast Gauss–Newton MSCKF becomes bigger when the running time is longer. It means the novel MSCKF will increase the final error. We can employ it at the low precision and fast speed occasion. Then, there is obvious hip change at the trajectories calculated by the novel MSCKF algorithm because of the switching of the two optimization methods. We also find that the error calculated by the novel MSCKF becomes bigger when the VIO runs longer. In order to reduce the accumulated error, we can add the close-loop detection.

Data Availability

The data of the manuscript is from the kitti datasets which can be accessed through MATLAB. The data can be obtained from https://figshare.com/s/395bc65e815a4e4b3f2f.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Department of the Fujian Science and Technology (Grant 2018H0003), Fujian University of Technology (Grant GY-Z17011), Quanzhou Science and Technology Bureau (Grant 2017G012), Department of Education of Fujian Province (Grant JT180339), and Fujian Universtiy of Technology (JGBK201911).

References

[1] x. Gao, *14 Lessons of Visual Slam: From Theory to Practice*, Beijing, China, 2007.

[2] Y. Sun, "Overview of visual inertial SLAM," *The Application Research of Computers*, vol. 36, no. 12, pp. 3530–3533, 2019.

[3] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[4] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the 2017th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, Nara, Japan, October 2007.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] P. Chansoo, "Illumination change compensation and extraction of corner feature orientation for upward-looking camera-based SLAM," in *Proceeedings of the 12th*

International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 224–227, Goyang, Korea, May 2015.

[7] S. Kohlbrecher and O. von Stryk, "A flexible and scalable SLAM system with full 3D motion estimation," in *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 155–160, Kyoto, Japan, September 2011.

[8] X. Zuo, P. Geneva, Y. Yang, W. Ye, Y. Liu, and G. Huang, "Visual-inertial localization with prior LiDAR map constraints," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3394–3401, 2019.

[9] S.-c. Chu and X. xue, "Optimizing ontology alignment in vector space," *Journal of Internet Technology*, vol. 21, no. 1, pp. 15–23, 2020.

[10] X. Xue and J. Chen, "Efficient user involvement in semi-automatic ontology matching," in *Proceedings of the IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–11, Piscataway, NJ, USA, December 2018.

[11] X. Xue and J. Chen, "Using compact evolutionary tabu search algorithm for matching sensor ontologies," *Swarm and Evolutionary Computation*, vol. 48, pp. 25–30, 2019.

[12] C.-H. Chen, F.-J. Hwang, and H.-Y. Kung, "Travel time prediction system based on data clustering for waste collection vehicles," *IEICE Transactions on Information and Systems*, vol. E102.DD, no. 7, pp. 1374–1383, 2019.

[13] K. Sun, K. Mohta, B. Pfrommer et al., "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[14] A. Yilmaz and A. Gupta, "Indoor positioning using visual and inertial sensors," in *Proceeding of the 2016 IEEE Sensors*, pp. 1–3, Orlando, FL, USA, November 2016.

[15] N. N. Win, "A novel particle filter based SLAM algorithm for lunar navigation and exploration," in *Proceedings of the International Conference on Robotics and Automation Engineering (ICRAE)*, pp. 74–78, Montreal, Canada, May 2019.

[16] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3565–3572, Roma, May 2007.

[17] L. E. Clement, V. Peretroukhin, J. Lambert, and J. Kelly, "The battle for filter supremacy: a comparative study of the multi-state constraint kalman filter and the sliding window filter," in *Proceedings of the 12th Conference on Computer and Robot Vision*, pp. 23–30, Halifax, NS, USA, June 2015.

[18] N. Qi, "An improved MSCKF algorithm based onmulti-mode augmentation method for the camera state equation," *Chinese Journal of Scientific Instrument*, vol. 40, no. 05, pp. 89–98, 2019.

[19] Y. Sun, "Method of indoor robot positioning based on improved," *MSCKF Algorithm", Computer Systems Applications*, vol. 29, no. 2, pp. 238–243, 2020.

[20] G. Jianjun and G. Dongbing, "A direct visual-inertial sensor fusion approach in multi-state constraint Kalman filter," in *Proceedings of the 34th Chinese Control Conference (CCC)*, pp. 6105–6110, Hangzhou, China, July 2015.

[21] T. Nguyen, G. K. I. Mann, A. Vardy, and R. G. Gosine, "Likelihood-based iterated cubature multi-state-constraint Kalman filter for visual inertial navigation system," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4410–4415, Vancouver, BC, USA, November 2017.

[22] M. Ramezani, K. Khoshelham, and L. Kneip, "Omnidirectional visual-inertial odometry using multi-state constraint Kalman filter," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1317–1323, Vancouver, BC, USA, November 2017.

[23] J. Li, H. Bao and G. Zhang, Rapid and robust monocular visual-inertial initialization with gravity estimation via vertical edges," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6230–6236, Macau, China, November 2019.

[24] P. Geneva, K. Eckenhoff, and G. Huang, "A linear-complexity EKF for visual-inertial navigation with loop closures," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 3535–3541, Montreal, QC, Canada, March 2019.

[25] M. Li and A. I. Mourikis, "Improving the accuracy of EKF-based visual-inertial odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 828–835, Saint Paul, MN, Canada, May 2012.

[26] K. Eckenhoff, P. Geneva, J. Bloecker, and G. Huang, "Multi-camera visual-inertial navigation with online intrinsic and extrinsic calibration," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 3158–3164, Montreal, QC, Canada, March 2019.

[27] S. Heo, J. Cha, and C. G. Park, "EKF-based visual inertial navigation using sliding window nonlinear optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 7, pp. 2470–2479, 2019.

[28] J. Hu and M. Chen, "A sliding-window visual-IMU odometer based on tri-focal tensor geometry," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3963–3968, Hong Kong. China, March 2014.

[29] K. Wen, W. Wu, X. Kong, and K. Liu, "A comparative study of the multi-state constraint and the multi-view geometry constraint kalman filter for robot ego-motion estimation," in *Proceedings of the eighth International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Hangzhou, China, September 2016.

[30] J. Xu, H. Yu, and R. Teng, "Visual-inertial odometry using iterated cubature Kalman filter," in *Proceedings of the Chinese Control And Decision Conference (CCDC)*, pp. 3837–3841, Shenyang, China, June 2018.

[31] M. Alibay, S. Auberger, B. Stanciulescu, and P. Fuchs, "Hybrid visual and inertial RANSAC for real-time motion estimation," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pp. 179–183, Paris, France, July 2014.

[32] J. Nikolic, "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 431–437, Hong Kong, China, June 2014.

[33] L. E. Clement, V. Peretroukhin, J. Lambert, and J. Kelly, "The battle for filter supremacy: a comparative study of the multi-state constraint kalman filter and the sliding window filter," in *Proceedings of the 12th Conference on Computer and Robot Vision*, pp. 23–30, Halifax, Canada, June 2015.