IJLAI Transactions on Science and Engineering (ISSN: 3006-3418) 2024, vol. 2, no. 1, pp: 8–16

KLT-VIO: Real-time Monocular Visual-Inertial Odometry

Yuhao Jin[1], Hang Li[1], and Shoulin Yin[1]

Software College, Shenyang Normal University
No.253, Huanghe North Street, Huanggu District, Shenyang, 110034 China
ewan.kim.std@gmail.com;lihangsoft@163.com;yslinhit@163.com
Corresponding author: Hang Li and Shoulin Yin

Abstract. This paper proposes a Visual-Inertial Odometry (VIO) algorithm that relies solely on monocular cameras and Inertial Measurement Units (IMU), capable of real-time self-position estimation for robots during movement. By integrating the optical flow method, the algorithm tracks both point and line features in images simultaneously, significantly reducing computational complexity and the matching time for line feature descriptors. Additionally, this paper advances the triangulation method for line features, using depth information from line segment endpoints to determine their Plcker coordinates in three-dimensional space. Tests on the EuRoC datasets show that the proposed algorithm outperforms PL-VIO in terms of processing speed per frame, with an approximate 5% to 10% improvement in both relative pose error (RPE) and absolute trajectory error (ATE). These results demonstrate that the proposed VIO algorithm is an efficient solution suitable for low-computing platforms requiring real-time localization and navigation.

Keywords: Visual-inertial odometry, Optical flow, Point features, Line features, Bundle adjustment.

1. Introduction

Visual-Inertial Odometry (VIO) technology stands as a pivotal foundation in contemporary robot navigation and localization, distinguished by its seamless integration of visual data with Inertial Measurement Units (IMU). VIO systems are broadly categorized into feature-based and direct methods, reflecting their principal technological trajectories for feature processing. Among the myriad of algorithms, the VIO module within ORB-SLAM3 is celebrated for its exceptional precision and robustness. However, its demanding hardware requirements impede its deployment across various application scenarios. Concurrently, VINS-Mono, a venerable VIO implementation, although commendable in precision, reveals room for enhancement in robustness when subjected to real-world challenges.

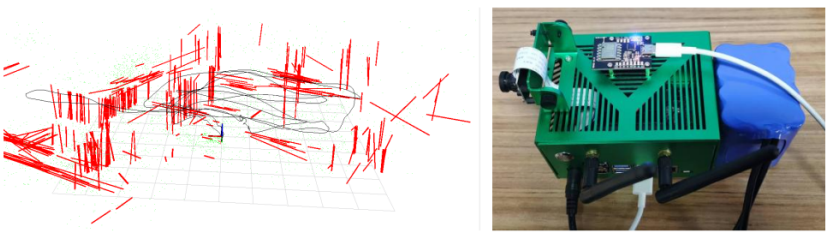

Fig. 1. The left image presents the test results of the KLT-VIO algorithm on the EuRoC datasets, where the red lines represent line features that have moved out of the sliding window, and the black curve denotes the camera's motion trajectory; the right image illustrates the NVIDIA Jetson Xavier

To surmount these challenges, our research introduces an advanced VIO algorithm that bolsters system robustness particularly in environments scarce in features, through the incorporation of line features. In comparison to the PL-VINS algorithm, which also harnesses point and line features, our proposed approach achieves a notable acceleration in processing, drawing the technology based on point and line features closer to the realm of real-time operation. Figure 1. illustrates the performance validation of our algorithm on the EuRoC dataset.

The key contributions of this study are encapsulated in the following facets:

1. The algorithm's front end has been enhanced with the introduction of optical flow tracking for both point and line features already extracted. Traditional VIO methodologies predicated on point and line features allocate a significant portion of processing time to the computation and matching of line features. By employing optical flow for unified tracking, our study substantially curtails the time dedicated to feature matching.

2. Capitalizing on the dualistic nature of points and planes in three-dimensional space, we present an innovative approach to computing the Plcker coordinates for line features. Utilizing the homogeneous coordinates of line segment endpoints to derive the Plcker coordinate matrix, and thereby the Plcker line itself, our method mitigates the issue of inaccurate Plcker line calculations due to acute angles between planes.
3. We have implemented and tested our algorithm on the NVIDIA Jetson Xavier platform, demonstrating enhanced robustness through the addition of line features. This improvement ensures effective pose estimation even under conditions of limited feature visibility, significantly curbing the occurrence of pose drift. Figure 1. showcases some of the experimental apparatus employed in our tests.

2. Related Works

In feature-based VSLAM or VO methods, ORB-SLAM3 [1] utilizes ORB feature points and computes descriptors for feature matching. Compared to its predecessor ORB-SLAM2 [2], its innovation lies in the introduction of Maximum A Posteriori (MAP) estimation during IMU initialization and the multi-submap system, enabling ORB-SLAM3 to effectively reuse information from all historical algorithm modules. OKVIS [3] is the first to propose a visual-inertial tightly-coupled VO system based on keyframes and BA optimization. ORBSLAM-VI [4] was the first to introduce a map reuse-based visual-inertial SLAM system. VINS-Mono [5] and VINS-Fusion are highly accurate and robust Visual-Inertial Odometry (VIO) systems that include loop detection and 4-DoF pose graph optimization, using Shi-Tomasi feature points and KLT optical flow for tracking features between frames.

In direct methods-based VSLAM or VO approaches, LSD-SLAM [6] can establish large-scale semi-dense maps, but it uses pose graph estimation instead of BA optimization, leading to less than ideal accuracy and robustness. SVO [7] employs FAST feature points and matches them using a direct method, with reprojection error as the optimization cost function, performing well in terms of accuracy and robustness but lacking loop detection and relocalization capabilities, meaning it only considers short-term feature correspondences. DSO [8] can calculate camera poses in situations where feature points are not distinct, enhancing robustness in low-texture environments. DSM [9] introduced map reuse techniques in direct methods. ROVIO [10] uses Shi-Tomasi feature points and direct method for feature matching, but its pose estimation is based on an Extended Kalman Filter (EKF). VI-DSO [11] builds upon DSO by incorporating an Inertial Measurement Unit (IMU) and combines photometric errors with inertial measurements, significantly improving precision and robustness compared to DSO.

In visual-inertial odometry based on point and line features: Choi et al. [12] proposed a geometrically constrained Extended Kalman Filter (EKF) framework for a line feature that every line is orthogonal or parallel to each other. PLF-VINS [13] enhanced the algorithm's accuracy, robustness, and real-time performance through the fusion of point and line features, as well as the integration of parallel line features. In PLI-VINS [14], a line segment extraction algorithm with adaptive threshold value was proposed to improve the quality of extracted line segment. Liu et al. [15] proposed an IMU-assisted hierarchical grid optical flow tracking method that could more accurately and quickly track points between frames. Zuo et al. [16] employed the orthonormal representation as the minimal parameterization to model line features along with point features in visual SLAM and analytically derived the Jacobians of the re-projection errors with respect to the line parameters, which significantly improved the SLAM solution. PL-VIO [17] was a tightly-coupled monocular visual-inertial odometry system exploiting both point and line features. Plcker coordinates and orthonormal representation for the line are employed to obtain both computation simplicity and representational compactness of a 3D spatial line.

3. Overview

To capture the camera's motion, this paper presents a real-time monocular Visual-Inertial Odometry named KLT-VIO that employs both point and line features. The paper uses optical flow to track points and lines, accelerating the feature matching process by skipping the computation and matching of descriptors. The architectural diagram of KLT-VIO is shown in Figure 2. The camera's motion is described by three rotational degrees of freedom (DoF), represented by quaternions, and three linear displacement DoFs, denoted by a three-dimensional vector. The algorithm is primarily divided into two modules: the feature perception module and the local visual-inertial odometry module. The algorithm mainly consists of two modules: the Perception of Features module, primarily responsible for feature extraction and matching, and the Local Visual-Inertial Odometry module, which focuses on pose estimation.

3.1. Feature Perception Module

KLT-VIO initially acquires RGB images and two types of datałacceleration and angular velocityłfrom the current frame using a calibrated monocular pinhole camera and an inertial measurement unit, respectively. The algorithm

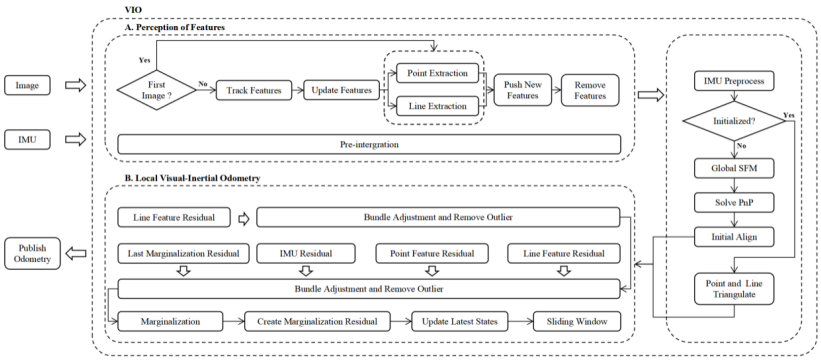

Fig. 2. Algorithm architecture diagram

extracts feature points and lines from each input image and matches features between two consecutive frames using optical flow. We utilize the Shi-Tomasi algorithm for point feature extraction, LSD or EDLine algorithms for line feature extraction, and the KLT algorithm for tracking both types of features. Specific details are described in sections 4.1. Drawing on the work of VINS-Fusion, we provide an initial estimate of motion through IMU pre-integration, which reduces computational complexity, enhances error tolerance, and aligns the image frame rate. Before initialization is complete, we calibrate the gyroscope and calculate velocity, gravity, and scale factors using LDLT decomposition. After preprocessing the input data, the entire system needs to be initialized, where it is necessary to quickly and accurately estimate the relative pose of the current system using feature points.

3.2. Local Visual-Inertial Odometry Module

KLT-VIO is an optimization-based odometry system. The primary issue is to solve the nonlinear least squares problem. In this context, we use the matching results from optical flow and calculate the spatial coordinates of points and lines between two consecutive frames through triangulation; specific details about triangulation are described in section 4.2. Based on the spatial coordinates of points and lines, we minimize re-projection errors to obtain a highly accurate camera pose, employing bundle adjustment to optimize the estimated poses.

4. Method

4.1. Using Optical Flow to Track Line Features

The extended optical flow method has enhanced its functionality to include the tracking of line features, which was initially restricted to point feature tracking. A line feature is essentially a set of points that possess directional and length information, and tracking them can be essentially reduced to tracking individual points. To apply sparse optical flow for line feature tracking, it's crucial to select representative characteristic points along the lines while preserving their directional and length information. Grounded in this theory, the paper introduces a line feature quantification method involving sampling and self-verification. As illustrated in Figure 3, the sampling points at the line feature's ends show significant pixel gradient changes, making them robust for sparse optical flow tracking. Conversely, the middle points may shift along the line due to uniform gradient changes, losing their positional accuracy but retaining directional information.

Based on the analysis, a line feature quantification method using four-point sampling is proposed. Optical flow tracking deems the endpoints reliable, but the middle points' accuracy is questionable. Despite this, the middle points remain confined to the line's path, preserving directional information while losing length details. Consequently, sparse optical flow tracking of line features necessitates a self-validation step. This validation compares the direction vectors from Figures 3 (b) and (c) with those from Figures 3 (a) and (d), determining the line feature's reliability as illustrated by the formula.

$$C = |\frac{(a_u - d_u)/(a_v - d_v)}{(b_u - c_u)/(b_v - c_v)}|. \tag{1}$$

Where C represents confidence, which in the algorithm implementation, needs to be greater than 0.99 to ensure that the line features tracked using sparse optical flow are trustworthy; a and d represent the two endpoints of the

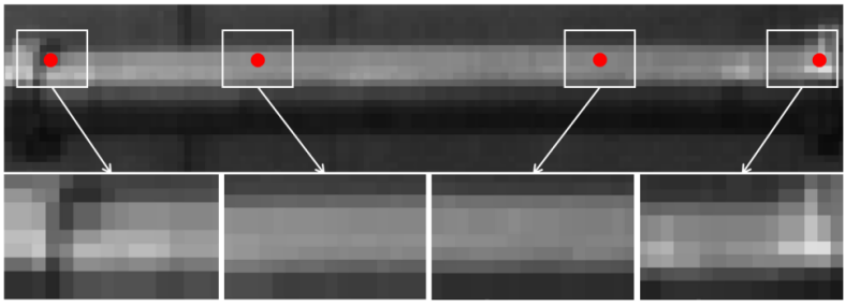

Fig. 3. Quantification of line features. The line feature is quantified into four feature points labeled as a, b, c, and d. When using sparse optical flow for tracking, their correspondences are preserved.

line feature; b and c represent the two middle sample points of the line feature; u denotes the x-coordinate of a point, v denotes the y-coordinate of a point.

An analysis of the PL-VINS [18] algorithm's feature extraction and matching component revealed that line feature descriptor computation and matching are time-consuming, exceeding the processing time for point feature tracking using sparse optical flow. The algorithm's front-end does not parallelize the extraction and assignment of point and line features across different CPU cores, resulting in cumulative processing times. The paper's proposed point and line feature tracking via sparse optical flow improves this, boosting VIO's front-end speed by nearly 40%, as validated by the experiments in Figure 4.

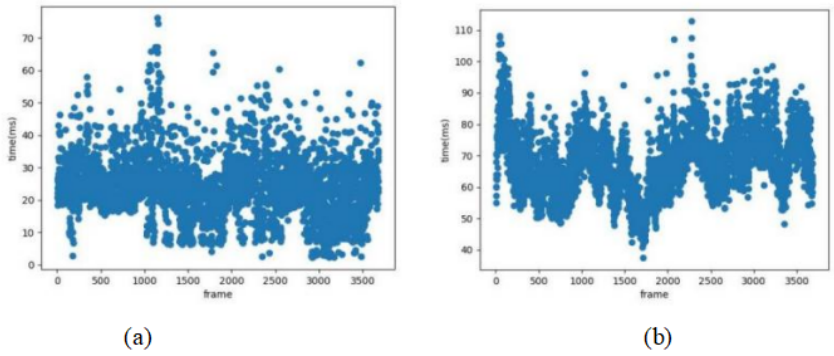

Fig. 4. (a) Time consumption for tracking line features using extended optical flow; (b) Time consumption for tracking line features using descriptor matching.

4.2. Triangulation of Line Features

In the field of three-dimensional computer vision, line feature triangulation is an important technique used to recover the three-dimensional structure of a scene and camera pose from images. The process involves the following steps:

1. Plane Construction: First, the LSD [19] algorithm is used to extract line features, and their endpoints in each view are located. Then, using these endpoints and the camera's optical center (or optic axis), a plane is constructed. This plane encompasses the potential position of the line feature in space.
   Comparison of extended optical flow and descriptor matching for line feature tracking, using a video with a total of 3682 frames. Each point in the graph represents the processing speed of each frame. It is evident that the processing speed using extended optical flow is significantly faster than the latter.
2. Angle Computation: For each line feature, several such planes are computed from different perspectives in the images. The system seeks the two planes with the largest included angle because a larger angle results in more accurate positioning and depth of the line feature where the two planes intersect.
3. Line Feature Positioning: Finally, the intersection of the two planes is computed to determine the position and depth of the line feature in three-dimensional space. This intersection line represents the optimal estimate of the line feature in three-dimensional space.The formula is shown below.

$$M = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \\ m_{13} & m_{14} & m_{15} & m_{16} \end{bmatrix} = P_1 P_2^T - P_2 P_1^T \tag{2}$$

$$L = [m_3 m_7 m_{11} - m_6 m_2 - m_1]^T. \tag{3}$$

Where M represents the Plücker matrix, P_1 and P_2 represent the two intersecting planes, and L represents the Plücker line.

When the camera's motion direction aligns with the direction vector of the line feature, there may be issues with a small angle between two planes, leading to errors. In such cases, the angle between the two planes can be very small, and if it falls below a set threshold, it can result in triangulation failure since the intersecting line cannot be accurately computed.

In the optimized method, we no longer rely on the intersection of two planes to determine the position of the line feature. Instead, based on the duality principle of points and planes in projective space, the formula for computing the Plücker line using the intersection of two planes can be transformed into using the endpoints to compute the Plücker line. Therefore, the depth information of the two endpoints of the line feature can be directly used to calculate its position in three-dimensional space. The advantage of this method is that as long as the depth of the two endpoints of the line feature can be accurately calculated, the position of the line feature in three-dimensional space can be precisely determined, regardless of the camera's motion direction.

The steps of the optimized line feature triangulation are as follows.

1. Compute endpoint depths: By identifying the line features in the image and finding the two endpoints of these line features in each view. Then, perform triangulation on each endpoint to determine its depth.
2. Determine the line feature's position: The Plücker coordinates of the line feature are calculated using the depths of the two endpoints. The formula is shown below.

$$M = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \\ m_{13} & m_{14} & m_{15} & m_{16} \end{bmatrix} = \bar{P_1} \bar{P_2}^T - \bar{P_2} \bar{P_1}^T \tag{4}$$

Where M represents the Plücker matrix, $\bar{P}_1$ and $\bar{P}_2$ represent the two intersecting planes, and L represents the Plücker line.

The two aforementioned methods for finding the Plücker line are shown in Figure 5.

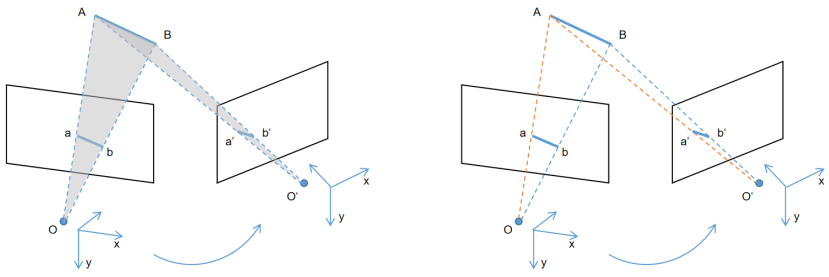

Fig. 5. The left image uses the intersection of two planes a and b to find the Plücker line, while the right image uses the homogeneous coordinates of two points a and b to determine the Plücker line.

5. Experiments and Analysis

The algorithm in this paper is developed based on Robot Operating System(ROS), and all experiments use OpenCV version 4.7.0 and the Ceres-solver library version 1.14.0. During the experiments, the threshold for the number of line features was set to 50, with a line feature mask width of 10 pixels, and the threshold for the number of point features was set to 150, with a point feature mask width of 30 pixels.

5.1. Dataset

In this paper, we utilized the EuRoC dataset, which is a stereo visual-inertial (VI) dataset specifically designed for indoor micro aerial vehicles (MAVs). The significance of this dataset lies in providing a standardized environment for evaluating and comparing different SLAM (Simultaneous Localization and Mapping) algorithms.

The EuRoC [20] dataset comprises two distinct scenarios: Machine Hall and Vicon Room. These scenes are provided by ETH Zurich, with the aim of advancing research and development in SLAM technology [21,22]. They are representative as they simulate indoor environments that MAVs might encounter in practical applications.

Experiments show that LK-VIO outperforms VINS-Fusion in terms of robustness and accuracy, and significantly surpasses PL-VINS in processing speed.

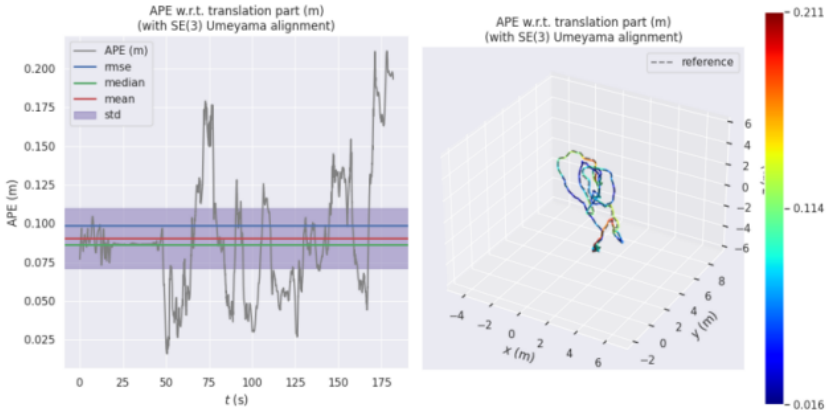

Fig. 6. Absolute pose error of KLT-VIO on MH_01_easy dataset

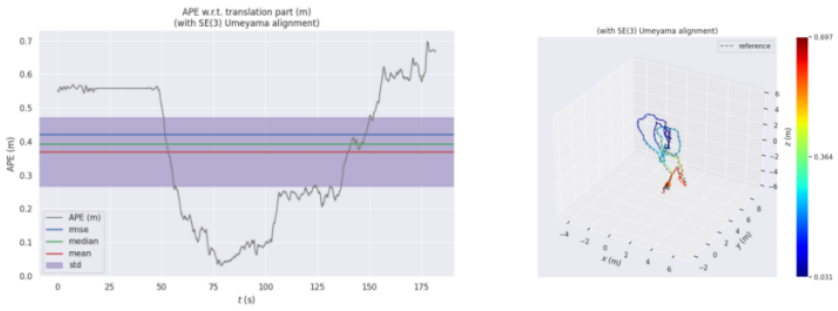

Fig. 7. Absolute pose error of VINS-Fusion on MH_01_easy dataset

Table 1. Performance of KLT-VIO and VINS-Fusion on the MH_01_easy dataset

Method	max	mean	median	min	rmse	sse	std
VINS-Fusion	0.697274	0.369813	0.393521	0.031016	0.422290	648.760747	0.203881
KLT-VIO	0.211310	0.09051	0.086498	0.015847	0.098496	35.293920	0.038843

The remaining test data is shown in the table below.

6. Conclusion

This paper primarily introduces a novel Visual-Inertial Odometry (VIO) algorithm based on monocular cameras and IMU called KLT-VIO. Compared to traditional VIO methods, the approach proposed in this paper employs an

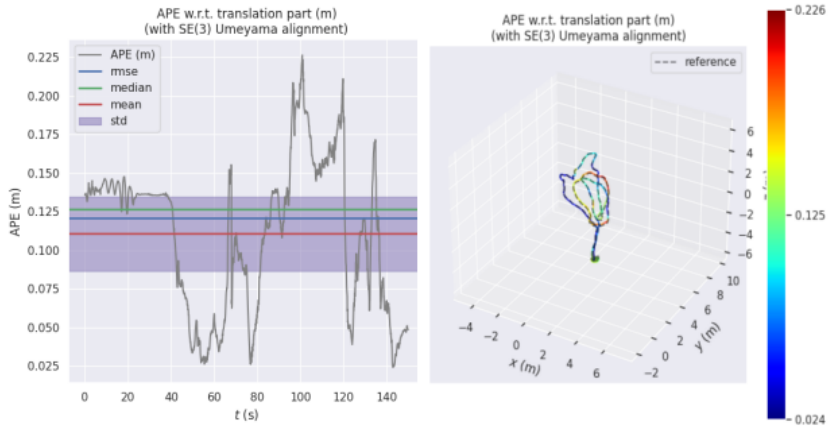

Fig. 8. Absolute pose error of KLT-VIO on MH_02_easy dataset

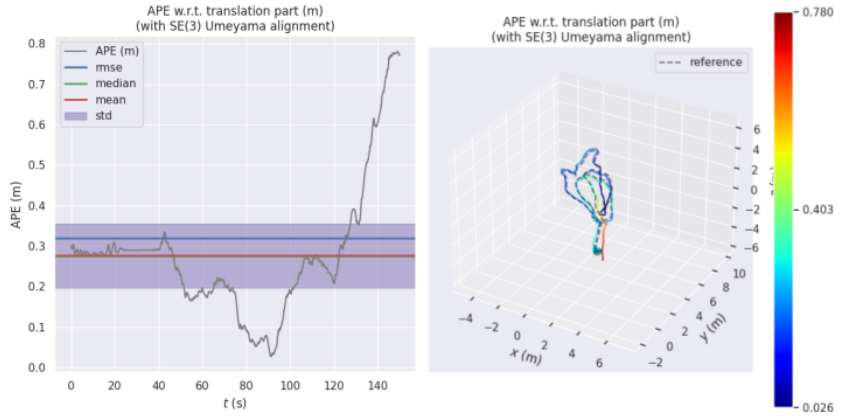

Fig. 9. Absolute pose error of VINS-Fusion on MH_02_easy dataset

Table 2. Performance of KLT-VIO and VINS-Fusion on the MH_02_easy dataset

Method	max	mean	median	min	rmse	sse	std
VINS-Fusion	0.780244	0.276083	0.275313	0.026406	0.318478	304.284477	0.158766
KLT-VIO	0.226376	0.110654	0.126630	0.024108	0.120596	43.630311	0.047950

Table 3. Performance of KLT-VIO and VINS-Fusion on the MH_02_easy dataset

Method	Dataset	max	mean	median	min	rmse	sse	std
VINS-Fusion	MH_03_medium	0.340473	0.182992	0.185718	0.015425	0.205624	110.691878	0.093781
LK-VIO	MH_03_medium	0.391490	0.223175	0.224726	0.071118	0.233272	142.460204	0.067888
VINS-Fusion	MH_04_difficult	0.600204	0.282647	0.24787	0.111838	0.301392	178.858738	0.104632
LK-VIO	MH_04_difficult	0.492885	0.268471	0.290959	0.044961	0.284548	158.534044	0.094290
VINS-Fusion	V1_01_easy	0.248323	0.136686	0.121735	0.019170	0.150714	63.21487	0.063495
LK-VIO	V1_01_easy	0.294921	0.101206	0.088756	0.030284	0.110968	34.269742	0.045511
VINS-Fusion	V1_02_medium	0.366023	0.151447	0.147489	0.050353	0.165002	43.234163	0.065494
LK-VIO	V1_02_medium	0.341891	0.166910	0.146781	0.072832	0.180930	51.983963	0.069831
VINS-Fusion	V1_03_difficult	0.308435	0.157611	0.168716	0.033760	0.172894	59.336542	0.071073
LK-VIO	V1_03_difficult	0.557396	0.183605	0.165870	0.011634	0.011634	86.170362	0.099820

enhanced feature tracking technique that not only tracks point features but also line features in images by integrating optical flow, effectively reducing the computational complexity of the entire VIO system. It also significantly decreases the matching time for line feature descriptors, enhancing the overall efficiency of the algorithm.

Furthermore, to improve the three-dimensional reconstruction accuracy of line features, this paper also proposes an improved triangulation method for line features. Traditional methods often overlook the depth information of line segment endpoints, whereas our algorithm utilizes this information to determine the Plcker coordinates of line segments in three-dimensional space, thus providing an alternative representation of line features. This enhancement is effective for handling dynamic scenes and long straight line structures and can significantly increase the dimensionality of the matrix.

To validate the performance of the proposed algorithm, we conducted a series of experimental tests on the publicly available EuRoC dataset. The test results show that our algorithm significantly outperforms the currently popular PL-VIO algorithm in terms of processing speed per frame, with an improvement of about 5% to 10% in relative pose error and absolute trajectory error. These performance improvements demonstrate the enhancements in real-time performance and accuracy of our VIO algorithm.

7. Conflict of Interest

The authors declare that there are no conflict of interests, we do not have any possible conflicts of interest.

Acknowledgments. None.

References

1. Campos C, Elvira R, Rodrłguez J J G, et al. Orb-slam3: An accurate open-source library for visual, visualCinertial, and multimap slam[J]. IEEE Transactions on Robotics, 2021, 37(6): 1874-1890.
2. Mur-Artal R, Tards J D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras[J]. IEEE transactions on robotics, 2017, 33(5): 1255-1262.
3. Hong E, Lim J. Visual inertial odometry using coupled nonlinear optimization[C]//2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 6879-6885.
4. Mur-Artal R, Tards J D. Visual-inertial monocular SLAM with map reuse[J]. IEEE Robotics and Automation Letters, 2017, 2(2): 796-803.
5. Qin T, Li P, Shen S. Vins-mono: A robust and versatile monocular visual-inertial state estimator[J]. IEEE Transactions on Robotics, 2018, 34(4): 1004-1020.
6. Engel J, Schps T, Cremers D. LSD-SLAM: Large-scale direct monocular SLAM[C]//European conference on computer vision. Cham: Springer International Publishing, 2014: 834-849.
7. Forster C, Pizzoli M, Scaramuzza D. SVO: Fast semi-direct monocular visual odometry[C]//2014 IEEE international conference on robotics and automation (ICRA). IEEE, 2014: 15-22.
8. Matsuki H, Von Stumberg L, Usenko V, et al. Omnidirectional DSO: Direct sparse odometry with fisheye cameras[J]. IEEE Robotics and Automation Letters, 2018, 3(4): 3693-3700.
9. Zubizarreta J, Aguinaga I, Montiel J M M. Direct sparse mapping[J]. IEEE Transactions on Robotics, 2020, 36(4): 1363-1370.
10. Bloesch M, Omari S, Hutter M, et al. Robust visual inertial odometry using a direct EKF-based approach[C]//2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2015: 298-304.
11. Von Stumberg L, Usenko V, Cremers D. Direct sparse visual-inertial odometry using dynamic marginalization[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 2510-2517.
12. Choi Y H, Lee T K, Oh S Y. A line feature based SLAM with low grade range sensors using geometric constraints and active exploration for mobile robot[J]. Autonomous Robots, 2008, 24: 13-27.
13. Lee J, Park S Y. PLF-VINS: Real-time monocular visual-inertial SLAM with point-line fusion and parallel-line fusion[J]. IEEE Robotics and Automation Letters, 2021, 6(4): 7033-7040.
14. Zhao Z, Song T, Xing B, et al. PLI-VINS: Visual-Inertial SLAM Based on Point-Line Feature Fusion in Indoor Environment[J]. Sensors, 2022, 22(14): 5457.
15. Liu X, Wen S, Zhang H. A Real-time Stereo Visual-Inertial SLAM System Based on Point-and-Line Features[J]. IEEE Transactions on Vehicular Technology, 2023.
16. Zuo X, Xie X, Liu Y, et al. Robust visual SLAM with point and line features[C]//2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 1775-1782.
17. He Y, Zhao J, Guo Y, et al. PL-VIO: Tightly-coupled monocular visualCinertial odometry using point and line features[J]. Sensors, 2018, 18(4): 1159.
18. Fu Q, Wang J, Yu H, et al. PL-VINS: Real-time monocular visual-inertial SLAM with point and line features[J]. arXiv preprint arXiv:2009.07462, 2020.
19. Von Gioi R G, Jakubowicz J, Morel J M, et al. LSD: A fast line segment detector with a false detection control[J]. IEEE transactions on pattern analysis and machine intelligence, 2008, 32(4): 722-732.

20. Burri M, Nikolic J, Gohl P, et al. The EuRoC micro aerial vehicle datasets[J]. The International Journal of Robotics Research, 2016, 35(10): 1157-1163.
21. Yin S, Li H, Laghari A A, et al. An Anomaly Detection Model Based On Deep Auto-Encoder and Capsule Graph Convolution via Sparrow Search Algorithm in 6G Internet-of-Everything[J]. IEEE Internet of Things Journal, 2024. DOI: 10.1109/JIOT.2024.3353337.
22. Yin S. Object Detection Based on Deep Learning: A Brief Review[J]. IJLAI Transactions on Science and Engineering, 2023, 1(02): 1-6.

Biography

Yuhao Jin is with the Software College, Shenyang Normal University. Research direction is computer application and AI.

Hang Li is with the Software College, Shenyang Normal University. Research direction is computer application, big data and AI.

Shoulin Yin is with the Software College, Shenyang Normal University. Research direction is computer application, image processing and AI.