

Article

Research on Visual SLAM Algorithm Based on Improved LSD Line Feature Extraction Algorithm

Yuang Guan, Li Qian * and Jinyang Du

School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; m315123315@sues.edu.cn (Y.G.)

* Correspondence: 01090008@sues.edu.cn

Abstract

Visual SLAM algorithms rely on image sequences to achieve autonomous localization and mapping, where line features act as crucial structural information to enhance system robustness in weakly textured or structured environments. However, conventional line feature-based methods, such as the Line Segment Detector (LSD) algorithm, are prone to over-segmentation during line segment extraction, resulting in a large number of redundant short segments and fragmented line pieces. This phenomenon increases the false matching rate, which in turn degrades the accuracy of pose estimation and the overall stability of the Visual SLAM system. To address the above issues, we perform comparative experiments on multiple public datasets between the proposed improved line feature algorithm and classical counterparts from dimensions of time overhead, line feature number and detection accuracy. The results show that the proposed algorithm incurs a 20% increase in overall time for line feature extraction and matching, yet achieves a 14% higher proportion of long line segments, an 8% improvement in Average Precision (AP) and a 15% rise in Average Recall (AR). It is thus verified that the proposed method retains real-time performance while remarkably improving its line segment matching success rate, with its localization accuracy and system robustness maintained or even enhanced.

Keywords: SLAM; line feature; line segment detection; line segment fusion

1. Introduction

In recent years, Simultaneous Localization and Mapping (SLAM) technology [1] has emerged as a research hotspot, widely applied across various sectors including industrial warehouse robots, commercial autonomous vehicles, drones, food delivery robots, and household cleaning robots [2–5]. The core mission of SLAM involves enabling robots to perceive their surroundings in unknown environments, determine their position and orientation, and construct detailed environmental maps [6]. Based on the types of external sensors used, SLAM can be categorized into laser SLAM and visual SLAM [7,8]. Visual SLAM offers advantages over laser SLAM such as lower costs, reduced power consumption, enhanced flexibility, and precise environmental description. Its integration with machine vision tasks further expands its application scope and development potential [9,10]. Visual SLAM has experienced three developmental stages, with the feature-based method being the mainstream for engineering applications—represented by the ORB-SLAM series [11–13], which realizes the whole process of SLAM tasks. However, visual SLAM still faces core challenges in practical deployment, such as low feature utilization in low-texture environments [14,15]. Additionally, another mainstream approach



Academic Editor: Janos Botzheim

Received: 31 December 2025

Revised: 15 February 2026

Accepted: 16 February 2026

Published: 28 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

is to adopt multi-sensor fusion [16], and integrating line features into such systems has gradually become a research trend [17,18].

Traditional visual SLAM systems often rely on feature-point-based methods, such as ORB or FAST corner detectors [19–24]. However, despite their effectiveness in many scenarios, these algorithms still face challenges in low-texture environments requiring further research and improvement. In low-texture scenes such as white walls, glass corridors and empty warehouses, feature point-based methods suffer from sparse feature extraction and high false matching rate, which directly leads to the failure of robot pose estimation and map construction [25]. In artificial environments like indoor buildings, outdoor structures, and urban streets, linear features are significantly more abundant than point features. They remain insensitive to lighting variations while providing essential geometric structural information. Consequently, linear feature-based SLAM has become a key focus in visual SLAM research, addressing the positioning accuracy limitations of point feature-based systems in low-texture environments. The research of line feature-based SLAM mainly includes three core links: line feature extraction, line feature description and matching, and pose optimization with map construction [26]. Among them, line feature extraction is the foundation, and representative point-line fusion SLAM works such as PL-SLAM [27] have effectively improved positioning accuracy in low-texture environments.

Current line feature extraction algorithms are categorized into two groups: traditional methods such as LSD, EDLines, and FLD, and deep learning-based approaches including L-CNN, HAWP [28–32] and so on. Traditional line feature extraction algorithms have the advantages of low computational complexity and high real-time performance, making them suitable for embedded robot platforms [33], but they are sensitive to noise and weak edges [34]. In contrast, deep learning-based methods have stronger robustness to complex scenes, but suffer from high computational complexity and difficulty in meeting real-time requirements of SLAM systems [35,36].

The working principle of the Line Segment Detector (LSD) algorithm involves computing gradients on a grayscale image, aggregating pixels with similar gradient orientations via region growing to accumulate a “Line Support Region” (LSR), and subsequently applying a series of strict filtering and rigorous validation steps to finally generate the characteristic line segments with high accuracy. The LSD algorithm has rapidly become the industry standard for line feature detection in traditional computer vision due to its plug-and-play convenience, high precision, and outstanding computational efficiency [37,38]. It has been successfully integrated into mainstream vision libraries such as OpenCV and MATLAB, and remains widely used across various vision-related fields, including image processing, target detection, and visual SLAM, to this day.

Compared with other traditional line feature extraction algorithms, LSD has obvious advantages in both accuracy and speed when compared with FLD [39], which directly contributes to its wide application in practical engineering and academic research. In response to the defects of the original LSD algorithm, scholars in the field have proposed a series of improved algorithms such as MLSD [40], whose core goal is to reduce the problem of duplicate line detection. However, these improved methods only focus on solving a single defect of the LSD algorithm and fail to address its inherent comprehensive limitations [41].

However, the most significant issue with the LSD algorithm lies in its susceptibility to duplicate line detection and line segmentation problems, which will lead to the generation of a large number of redundant and fragmented line segments. These problems not only affect the quality of line feature extraction but also bring great interference to the subsequent feature matching and pose estimation of visual SLAM systems. Therefore, optimizing the traditional LSD algorithm to improve its robustness and feature extraction quality while

ensuring its real-time performance has important theoretical significance and practical engineering value for the development and application of line feature-based SLAM systems.

This paper chooses to improve the LSD algorithm to solve the above-mentioned core problems, and the main improvement points include the following:

- An adaptive length suppression strategy is adopted to effectively filter and eliminate invalid short lines that are irrelevant to the structural features of the scene, thereby improving the quality and effectiveness of line features and reducing the interference of redundant information.
- The angle and endpoint search grouping strategy is used to accurately identify and filter candidate line segment groups that meet the fusion criteria, laying a solid foundation for subsequent line segment fusion and ensuring the rationality of the grouping results.
- The similarity evaluation index of line segments is introduced, and the line segment fusion is carried out according to the predefined fusion rules, which effectively solves the problems of line segmentation and duplicate line detection, and enhances the structural consistency of line features.

The technical route of the full text is shown in Figure 1:

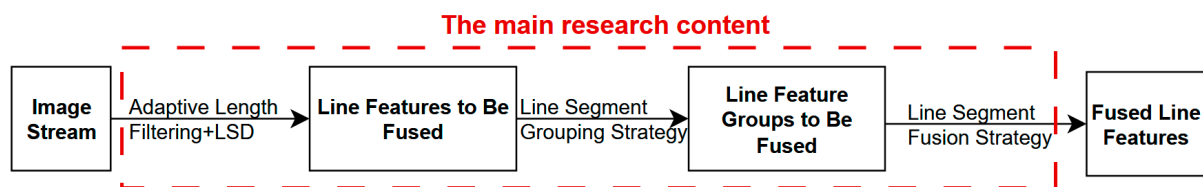


Figure 1. The main research content.

2. Adaptive Length Suppression Strategy

Different from the simple line detection, the purpose of extracting line features in SLAM system is to use the line features to construct constraints for pose estimation. Only a certain number of line features are needed to construct constraints, so the line features have higher accuracy and quality requirements.

Longer line segments typically originate from regions of images with continuous strong gradients, making them more reliable and stable, and thus contributing more significantly to the accuracy and robustness of pose estimation [25]. In contrast, shorter line features usually come from areas with blurred textures, exhibiting smaller gradient variations. This results in a higher number of unstable detected lines that are difficult to reliably track [42]. Based on this, this paper proposes an adaptive-length filtering algorithm.

2.1. Core Formula for Dynamic Threshold

In line feature length filtering algorithms, the threshold is usually set as the product of a scaling factor and a fixed value. While this approach is straightforward, it lacks robustness and fails to account for the continuity between image streams. This paper proposes a novel minimum line feature length threshold (*minLength*), which filters out line features shorter than this threshold in the filtering algorithm. The formula for calculating this threshold is shown in Equation (1):

$$minLength = \left(1 - \frac{|mean - std|}{mean}\right) \times mean \times \alpha \quad (1)$$

The parameters in Equation (1) are defined as follows:

mean: the mean value of line feature length in the current processed data, reflecting the overall length level of line features.

std: the standard deviation of line feature length in the current data, indicating the dispersion of line feature lengths.

α : the inter-frame quantity difference factor, a core parameter for dynamic threshold adjustment, ranges from (0, 1.2] to quantify how adjacent frame segment variations affect the threshold.

The dynamic threshold combines the statistical characteristics of line feature length and the characteristics of inter-frame change, and can be adjusted adaptively to ensure the rationality and adaptability of the threshold.

2.2. Calculation of the Impact Factor for the Difference in Frame Count

Conventional threshold-based methods isolate individual images for screening without considering the image stream as a continuous whole. This paper proposes an inter-frame quantity difference factor α , which dynamically adjusts α values based on the difference in linear feature counts detected between adjacent frames. This adjustment subsequently optimizes the minimum length (*minLength*), effectively preventing both excessive and insufficient linear features caused by inappropriate threshold selection. The specific methodology is outlined below.

Define the basic statistical variables: let the number of line features in the $k - 2$ frame after preliminary screening be N_{k-2} , and the number of line features in the $k - 1$ frame after preliminary screening be N_{k-1} . The inter-frame line segment difference ΔN is calculated to describe the absolute change in line segment count between adjacent frames, as shown in Equation (2):

$$\Delta N = N_{k-1} - N_{k-2} \quad (2)$$

To eliminate the interference of the absolute quantity of line segments on the degree of change, the change rate r is introduced, and the calculation formula is shown in Equation (3):

$$r = \frac{\Delta N}{N_{k-2}} \quad (3)$$

Based on the value of r , the value of α is calculated according to the following Formula (4). This formula, similar to the mechanism of negative feedback, ensures that the number of detected line *minLength* segments remains relatively stable. Subsequently, the calculated data is substituted into Formula (1) for final computation.

$$\alpha = \begin{cases} 1 - 0.2 \times (1 + r), & r < -0.1 \\ 1, & -0.1 \leq r \leq 0.1 \\ 1 + 0.1 \times (r - 0.1), & r > 0.1 \end{cases} \quad (4)$$

The final step involves extracting line feature lengths using the determined minimum threshold. The effects before and after applying the adaptive length filtering are shown in Figure 2. The figure shows that most unstable short-line features are significantly reduced after filtering, while long-line features remain intact.



Figure 2. Comparison of adaptive length filtering effects.

Using 60 px as the threshold for long lines, we compared two length filtering methods with Kitti dataset data, yielding the results shown in Table 1.

Table 1. Comparison of Algorithm Performance.

	Before Length Filtering	After Length Filtering
Per-frame processing time (s)	0.0194832	0.0254832
Number of segments	1286	362
Long-term proportion	0.324101	0.467314
Standard deviation (time, s)	0.0021	0.0035
Standard deviation (segments)	45	28
Standard deviation (long-term proportion)	0.022	0.015

After filtering, the proportion of long lines increased by approximately 14.3% compared to the pre-processing stage. Meanwhile, the reduction in algorithm extraction speed remained within an acceptable range, with the filtered segments primarily consisting of short lines that could interfere with final computational accuracy. Although the number of valid samples decreased, the model's computational accuracy improved significantly. The subsequent computational workload was further reduced by eliminating invalid data, thereby optimizing both the efficiency and reliability of line feature processing.

3. Segmentation Policy

In the LSD algorithm, linear features are segmented into multiple sub-segments during extraction, which fails to restore the true features. Based on this, this paper designs a line segment grouping strategy and performs fusion according to similarity, thereby solving the line segment segmentation problem in LSD.

The line features extracted by the LSD algorithm typically contain over 10^3 features. Therefore, grouping these features is essential before merging them. Line segments with similar directions and spatial positions are more likely to be merged. This paper proposes a combined strategy of angle grouping and endpoint grouping.

3.1. Angle Grouping Strategy

The strategy groups all line segments into 18 clusters at 10° intervals. As shown in Figure 3, these are the $10\text{--}20^\circ$ and $90\text{--}100^\circ$ segment groups. This angular grouping significantly reduces the search space for line feature fusion, thereby substantially lowering the computational complexity and operational scale of the subsequent endpoint distance clustering strategy.



(a) Grouping of line features at $90\text{--}100^\circ$ intervals (b) Grouping of line features at $10\text{--}20^\circ$ intervals

Figure 3. Group line segments by angle.

3.2. Endpoint Distance Grouping Policy

After grouping feature line segments by angle, the endpoint distance is used to perform secondary grouping within each segment group. First, for all segments in the current angle group, the “minimum endpoint distance” between any two segments is calculated—the smallest of four distances between the endpoints of Segment 1 and Segment 2. If this distance falls below a predefined threshold, the two segments are considered “positionally adjacent”. Their association relationship is then recorded in an adjacency list, and all mutually connected (direct or indirect) segments are merged into a final group using depth-first search (DFS). This process yields all subgroups within the angle group, as shown in Figure 4. After endpoint secondary grouping, the line features are divided into multiple groups, each represented by a distinct color. Upon completion of secondary grouping, the number of segments in each group typically converges to a range of 2 to 3, a distribution pattern that significantly reduces the complexity of subsequent segment fusion operations.

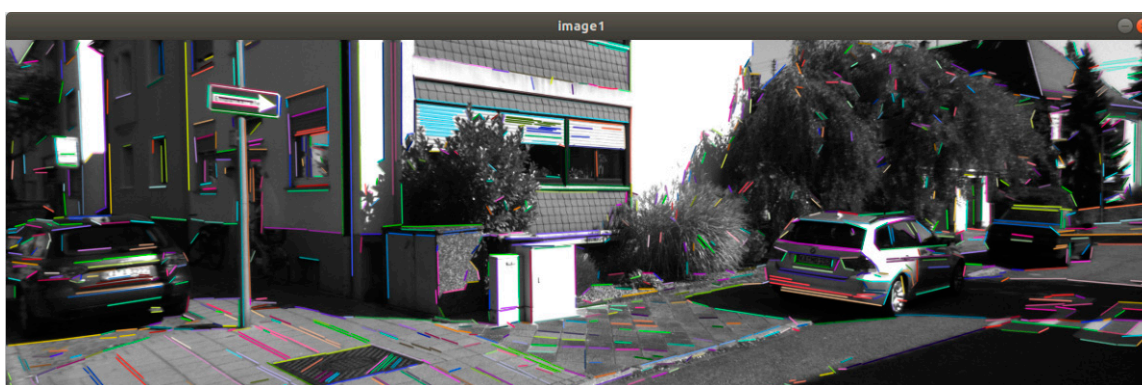


Figure 4. Segment grouping results by angle and secondary endpoint grouping (different colors indicate different groups).

4. Line Segment Merging Strategy

4.1. Similarity Design of Line Segment Combination

In conventional line segment fusion algorithms, a core segment is typically selected as the “baseline” within a segment group. This core segment serves as the fusion benchmark, with all other non-core segments in the group being fused sequentially. However, the algorithm overlooks the possibility of multiple segments being fusible within the group and fails to account for variations in fusion order, which ultimately impacts the final fusion outcome.

Based on this, the paper proposes the concept of baseline groups, where the maximum line segment length is defined as $maxLength$, and the selection interval for baseline groups is set at $0.8 \cdot maxLength$ to $maxLength$. The line segments within a baseline group are independent and do not undergo fusion with each other. Subsequently, the line segments from non-baseline groups are sequentially compared and fused with those in the baseline group.

In the fusion sequence, this paper introduces the concept of similarity. The line segments with high similarity to the baseline are prioritized for fusion, rather than simply sorting them by length, to ensure the accuracy of the fusion.

The similarity is calculated based on three parameters, the length, angle, and positional relationship of the line segments, as shown in Equation (5):

$$S(L_1, L_2) = \omega_1 \cdot S_d + \omega_2 \cdot S_p + \omega_3 \cdot S_l \quad (5)$$

The empirical values of similarity $\omega_1 = 0.4$, $\omega_2 = 0.35$, $\omega_3 = 0.2$, where $S(L_1, L_2)$ represents the similarity between L_1 and L_2 . S_d is the directional similarity, S_p is the positional similarity, and S_l is the length similarity.

The formulas for calculating the three similarity levels are as follows:

1. Directional Similarity S_d Calculation

Calculate the direction vector of a line segment : $\vec{v}_1 = (x_{1s} - x_{1e}, y_{1s} - y_{1e})$, $\vec{v}_2 = (x_{2s} - x_{2e}, y_{2s} - y_{2e})$

$$\text{Calculate the cosine of the direction : } \cos\theta = \frac{\vec{v}_1 \cdot \vec{v}_2}{|\vec{v}_1| \cdot |\vec{v}_2|}$$

$$\text{Angle using vector dot product : } S_d = \max(\cos\theta, 0)$$

2. Position Similarity S_p Calculation

Calculate the minimum spatial $D_{min}D_{th}$ distance between two line segments (the shortest distance between the endpoints of one line segment and the other, or the minimum of the perpendicular distances between them). Set the distance threshold to 10% of the average length of the line segments.

$$\text{Position similarity : } S_p = \max\left(1 - \frac{D_{min}}{D_{th}}, 0\right) \quad (6)$$

3. Length similarity S_l

$$\text{Calculate line segment length: } L_1 = |\vec{v}_1|, \quad L_2 = |\vec{v}_2|$$

$$\text{Length similarity : } S_l = 1 - \frac{|L_1 - L_2|}{\max(L_1, L_2)} \quad (7)$$

4.2. Line Segment Merging Algorithm

After determining the baseline groups and sequence for fusion, the formal fusion process is then initiated. Define the angles of line segments L_1 and L_2 as θ_1 and θ_2 , respectively, with their midpoints being (x_1, y_1) and (x_2, y_2) , respectively. The calculation is then performed as follows:

Step 1: As shown in Figure 5, the weighted average of the center points of the merged line segments (calculated based on their lengths) determines the new point G (x_G, y_G) , through which the combined line passes. The specific formula is as follows:

$$x_G = \frac{l_1 \cdot x_1 + l_2 \cdot x_2}{2(l_1 + l_2)}, \quad y_G = \frac{l_1 \cdot y_1 + l_2 \cdot y_2}{2(l_1 + l_2)} \quad (8)$$

Step 2: As shown in Figure 5, the direction θ_G of the merged line segment is the weighted average of the directional angles of l_1 and l_2 , with the specific formula as follows:

$$\theta_G = \frac{l_1 \cdot \theta_1 + l_2 \cdot \theta_2}{l_1 + l_2} \quad (9)$$

As shown in Figure 6, the line segment segmentation problem of LSD has been significantly improved after line segment fusion.

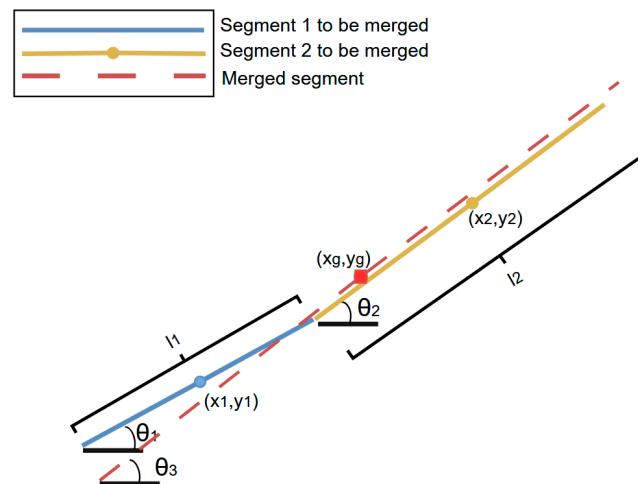
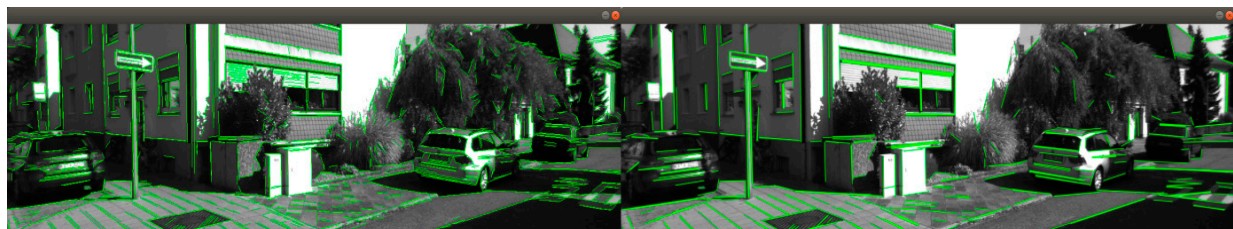


Figure 5. Endpoint Secondary Grouping Based on Angle Grouping.



(a) Example of the fused front segment group (b) Example of the post-fusion segment group

Figure 6. Overall Effect Renderings Before and After Segment Merging.

5. Simulation and Experimental Results

To validate the improved line segment feature algorithm, the proposed method was compared with conventional line segment detection techniques such as LSD and ED-Lines. The laboratory platform configuration includes an Intel®Core-i9-14900HX processor (2.20 GHz) and 16 GB of RAM.

5.1. Linear Feature Detection Experiment 1

The experiment uses the first 100 frames of the left grayscale camera stream from Sequence 01 of the Kitti dataset as the test data. Modified LSD_Advance and LSD algorithms, along with the EDLines algorithm, were evaluated based on four metrics: time required for line feature extraction, proportion of long lines, total extracted line segments, the number of line feature correspondences that match the ground truth, and overall processing time of the entire front-end algorithm.

Through comparative analysis of the aforementioned experimental datasets, Figure 7a demonstrates that LSD_Advance and EDLines exhibit higher computational costs than the LSD algorithm. However, Figure 7b,c reveal that LSD_Advance extracts fewer line segments but with a higher proportion of longer ones. When the subsequent matching algorithm is integrated, Figure 7d shows significantly more stable and high-quality matched line segments. Figure 7e further confirms that its full-process algorithm from detection to matching achieves slightly better runtime than EDLines.

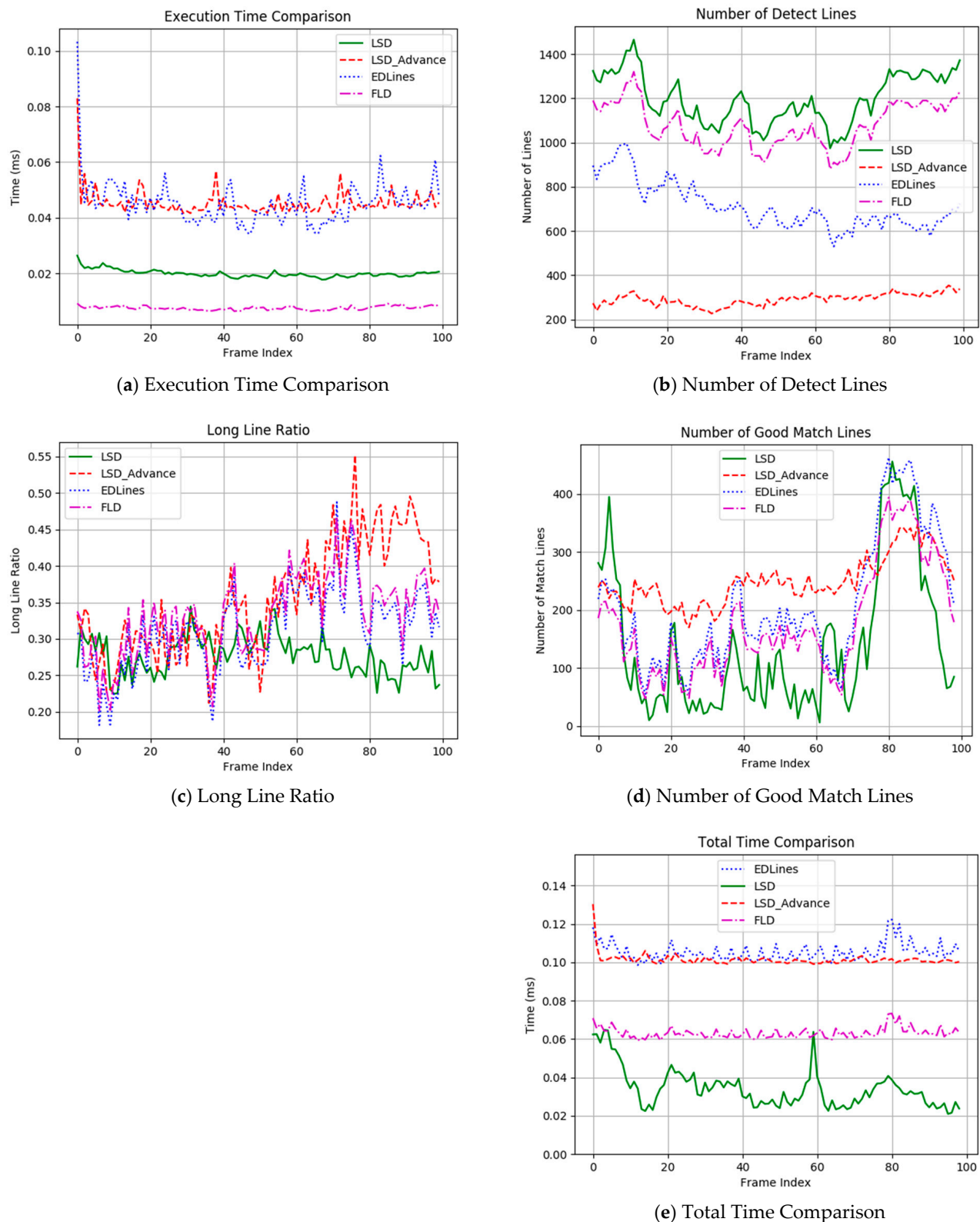


Figure 7. Comparison 1 of performance between LSD, FLD, EDLines, and LSD_Advance algorithms.

5.2. Line Feature Detection Experiment 2

The popular YorkUrban line segment dataset was selected for the line segment extraction algorithm test. This dataset consists of 45 indoor images and 57 outdoor images, with a resolution of 640×480 . To comprehensively evaluate the performance of the improved algorithm, the following metrics were adopted: average precision (AP), average recall (AR),

F-score, and average frame processing time (T , in milliseconds) [43]. Based on the number of correctly detected line segments (true positives, TP), the number of incorrectly detected line segments (false positives, FP), and the number of line segments present in the image but undetected by the algorithm (false negatives, FN), the precision, recall, and F -score were defined as follows [44]:

$$Precision = \frac{TP}{TP + FP} \quad (10)$$

$$Recall = \frac{TP}{TP + FN} \quad (11)$$

$$F - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (12)$$

For each ground truth line segment L_{gt} , find the set $\{L_p\}$ from the detected line segments that satisfy the following conditions:

$$d_p(L_{gt}, L_p \mid \theta_{gt}) \leq \lambda_{dist} \quad (13)$$

$$d_a(\theta_{gt}, \theta_p) \leq \lambda_{ang} \quad (14)$$

In the equations: λ_{dist} and λ_{ang} are the distance threshold and angle threshold, respectively. θ_{gt} , θ_p are the angles of L_{gt} and L_p . Equation (13) is the perpendicular distance between the centers of L_{gt} and L_p along the direction of θ_{gt} . Equation (14) is the angular difference between them.

If the overlap ratio between $\{L_p\}$ and L_{gt} is greater than the overlap threshold $\lambda_{overlap}$, then L_p is considered to match L_{gt} in terms of geometric features.

It is common to set

$$\lambda_{dist} = 1, \lambda_{ang} = 5, \lambda_{overlap} = 0.75.$$

$$\frac{L_{gt} \cap \{L_p\}}{L_{gt}} \geq \lambda_{overlap} \quad (15)$$

As shown in Table 2, the proposed LSD_Advance method achieves the best overall performance. It ranks first in all three accuracy metrics: Average Precision ($AP = 0.3278$), Average Recall ($AR = 0.5836$), and the combined F-score (0.3712). This indicates a superior balance between detection correctness and completeness compared to LSD, FLD, and EDLines. While LSD_Advance is slower than the original LSD ($T = 21.2$), its computational time ($T = 54.8$) is comparable to EDLines ($T = 53.2$). The significant improvement in detection quality justifies this moderate increase in processing time, establishing a favorable trade-off for applications where accuracy is prioritized.

Table 2. LSD, ED Lines, and LSD_Advance.

Method	AP	AR	F-Score	T
LSD	0.2418	0.4324	0.3352	21.2
FLD	0.2126	0.5580	0.3645	32.7
EDLines	0.2784	0.5328	0.3581	53.2
LSD_Advance	0.3278	0.5836	0.3712	54.8

Six representative scene sequences were selected, respectively, from six different typical scenarios of the EuRoC dataset. In the experiments, three key evaluation indicators were calculated to comprehensively measure the performance of each algorithm, including the average time consumption per frame (reflecting real-time efficiency), the number of

line features detected (reflecting feature richness), and the matching rate with the ground truth (reflecting detection accuracy).

As shown in Figure 8a, the LSD_Advance line feature detection algorithm has a longer average time consumption compared with other classical line feature extraction algorithms. However, it is still within a practically acceptable range, which can meet the real-time requirements of most feature-based visual SLAM systems. Specifically, LSD_Advance consumes the most time due to the additional line segment grouping and merging operations involved in its implementation. These two optimization steps effectively solve the line segment segmentation and duplicate detection problems of the original LSD algorithm, achieving a balance between computational efficiency and feature quality.

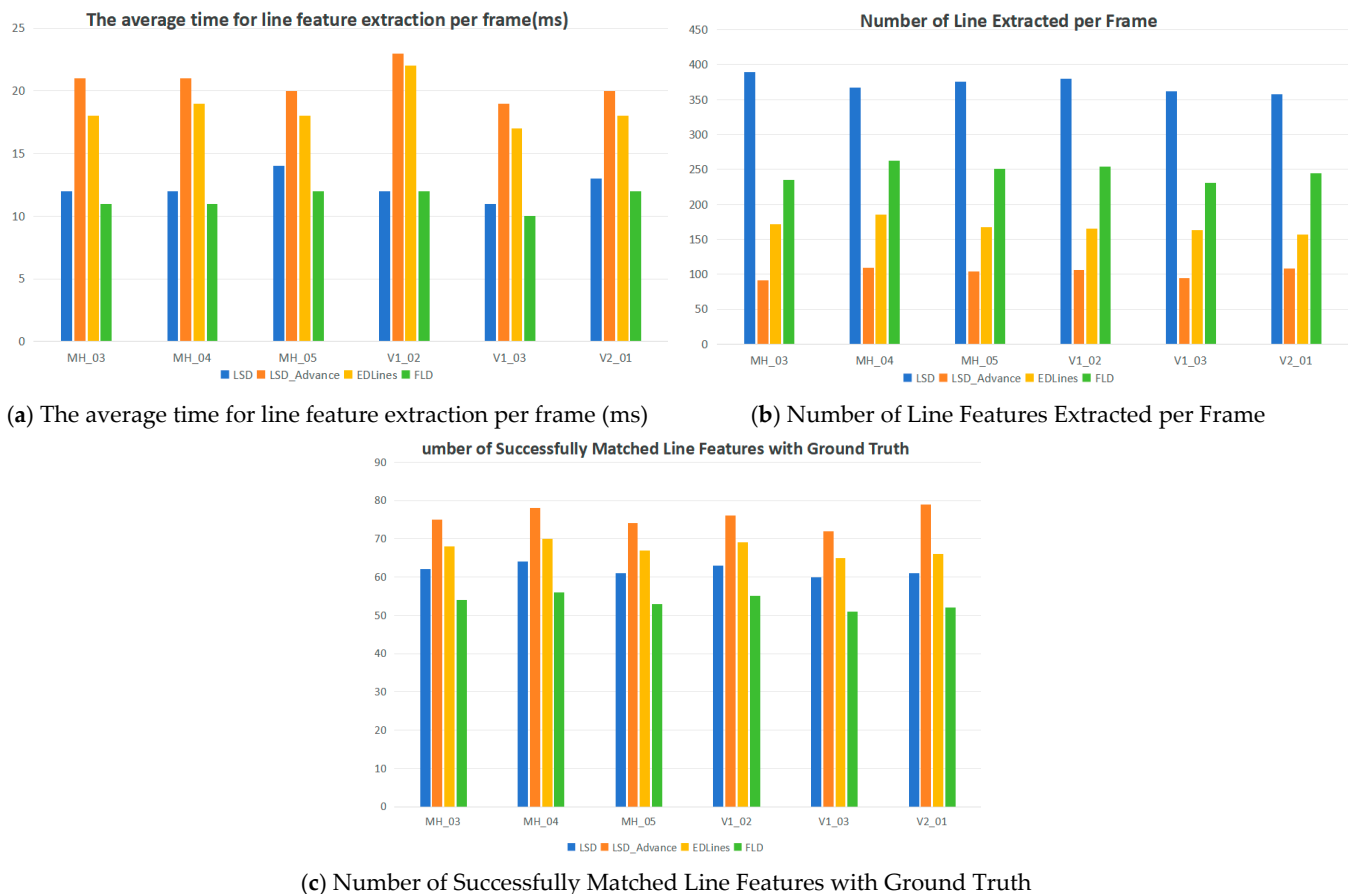


Figure 8. Comparison 2 of performance between LSD, FLD, EDLines, and LSD_Advance algorithms.

Combined with the results presented in Figure 8b,c, through adaptive short line filtering and intelligent line segment merging, the improved EDLines algorithm extracts relatively fewer line features compared with other classical algorithms. This is because the short line filtering step eliminates invalid and redundant short line segments, improving feature quality while reducing useless features. Nevertheless, its matching rate with the ground truth provided by the EuRoC dataset has been improved by approximately 13.4% in contrast to the original EDLines algorithm, which demonstrates that the proposed optimized strategies can effectively enhance the accuracy and structural consistency of line feature extraction, providing reliable support for subsequent visual SLAM stages.

6. Conclusions

To address the inherent limitations of the traditional LSD (Line Segment Detector) line feature extraction algorithm—such as the tendency to generate excessive redundant short segments, poor robustness to complex structured or low-texture scenarios, and insufficient consistency in extracted line segment structures—this study proposes three innovative and targeted optimization strategies: adaptive line segment length suppression, adaptive segment grouping based on spatial adjacency and directional similarity, and an intelligent segment merging mechanism with weighted direction fusion. These targeted improvements are designed to mitigate the key drawbacks of the original LSD algorithm, thereby significantly enhancing the quality, completeness, and structural consistency of line segment extraction while reducing the interference of invalid or redundant features.

To fully verify the effectiveness and superiority of the proposed LSD_Advance algorithm, comprehensive experimental validation was conducted on three widely recognized and publicly available benchmark datasets, namely YorkUrban, Kitti and EuRoC which cover diverse scene types including urban buildings, road environments, and low-texture regions—scenarios where traditional line feature extraction algorithms often struggle. The experimental results clearly demonstrate the superior performance of the improved LSD_Advance algorithm compared to the original LSD method and other state-of-the-art line segment extraction approaches.

Specifically, quantitative evaluation indicators show that LSD_Advance achieves a 14.3% significant increase in the long-segment ratio (a critical indicator reflecting the completeness of structural line features), and effectively raises the F-score—an integrated metric balancing precision and recall—from 0.3352 to 0.3712. This notable improvement in F-score fully reflects that the proposed algorithm achieves a better trade-off between precision (reducing false positive line segments) and recall (retaining more valid structural line segments), which is crucial for subsequent visual SLAM feature matching and pose estimation.

Although the per-frame processing time of the algorithm rises moderately from the original 19.5 ms to 25.5 ms due to the addition of the three optimization strategies, this slight increase in computational cost is well compensated by the significant improvements in feature quality. Specifically, the effective reduction in redundant line segments streamlines the subsequent key stages of visual SLAM, including feature matching, bundle adjustment, and loop closure detection, ultimately contributing to an overall 12.39% reduction in the feature mismatch rate and a notable improvement in the overall robustness and stability of the visual SLAM system.

In summary, the proposed LSD_Advance approach successfully maintains real-time efficiency (meeting the basic real-time requirements of most feature-based visual SLAM systems) while delivering more structurally consistent, complete, and reliable line features. These advantages make it a suitable and effective component for feature-based visual SLAM systems operating in complex structured environments, low-texture scenes, or other challenging scenarios where high-quality line feature extraction is essential for system performance.

Author Contributions: Conceptualization, Y.G.; methodology, Y.G.; software, Y.G.; validation, Y.G., L.Q. and J.D.; investigation, Y.G., L.Q. and J.D.; writing—original draft preparation, Y.G.; writing—review and editing, Y.G. and L.Q.; visualization, Y.G. and L.Q.; supervision, L.Q. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to acknowledge the financial support of the National Natural Science Foundation of China (Grant No. 52172372).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, Y.; Zhang, P.; Yao, J.; Luo, Z.; Ren, X. Optimization design of laser SLAM system based on resampling technology. *China Sci. Technol. Pap.* **2020**, *15*, 125–130.
2. Zhang, B.; Zhu, M.; Lin, C.; Zhu, D. Research on AGV map building and positioning based on SLAM technology. In Proceedings of the 2022 IEEE 5th International Conference on Automation, Electronics and Electrical Engineering (AUTE EE), Shenyang, China, 18–20 November 2022; pp. 707–713.
3. Li, J.; He, J. Localization and Mapping for UGV in Dynamic Scenes with Dynamic Objects Eliminated. *Machines* **2022**, *10*, 1044. [\[CrossRef\]](#)
4. Hu, J.; Hu, J.; Shen, Y.; Lang, X.; Zang, B.; Huang, G.; Mao, Y. 1D-LRF Aided Visual-Inertial Odometry for High-Altitude MAV Flight. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 5858–5864.
5. Pan, S.; Xie, Z.; Jiang, Y. Sweeping robot based on Laser SLAM. *Procedia Comput. Sci.* **2022**, *199*, 1205–1212. [\[CrossRef\]](#)
6. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [\[CrossRef\]](#)
7. Chen, P. Active Mapping and Obstacle Avoidance for Wheeled Robots Based on Indoor Visual Positioning. Master's Thesis, University of Electronic Science and Technology of China, Chengdu, China, 2020. [\[CrossRef\]](#)
8. Gu, J.; Bellone, M.; Pivoňka, T.; Sell, R. CLFT: Camera-LiDAR Fusion Transformer for Semantic Segmentation in Autonomous Driving. *IEEE Trans. Intell. Veh.* **2024**, early access. [\[CrossRef\]](#)
9. Tian, Y.; Chen, H.; Wang, F.; Chen, X. A review of SLAM algorithms for indoor mobile robots. *Comput. Sci.* **2021**, *48*, 223–234.
10. Luo, Y.; Shen, J.-X.; Li, F.-Y. Review of visual SLAM research based on deep learning in dynamic environments. *Semicond. Optoelectron.* **2024**, *45*, 1–10.
11. Jin, Y.; Liu, H.; Li, Z.; Zhong, Y. ORB-SfMLearner: ORB-Guided Self-supervised Visual Odometry with Selective Online Adaptation. In Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, 19–23 May 2025; pp. 1046–1052.
12. Campos, C.; Elvira, R.; Gómez Rodríguez, J.J.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. *IEEE Trans. Robot. (TRO)* **2023**, *39*, 1874–1890.
13. Zhang, M.; Jiang, J.; Wang, J.; Han, X. Stereo visual-inertial simultaneous localization and mapping combined with line feature based on ORB-SLAM3. In Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, 19–23 May 2025; pp. 789–795.
14. Li, X.; Chen, W.; Scaramuzza, D.; Yang, R. Line Feature Matching Optimization for ORB-SLAM3 in Low-Texture Structured Environments. *IEEE Trans. Robot. (TRO)* **2024**, *40*, 2109–2125.
15. Wang, Q.; Zhang, L.; Davison, A.J. Adaptive Point-Line Fusion for Monocular Visual-Inertial ORB-SLAM with Long Line Segment Prior. In Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 13–17 May 2024; pp. 5689–5695.
16. Wang, J.-K.; Zuo, X.-X.; Zhao, X.-R.; Lv, J.-J.; Liu, Y. Review of multi-source fusion SLAM: Current status and challenges. *J. Image Graph.* **2022**, *27*, 368–389. [\[CrossRef\]](#)
17. Zhang, J.; Ye, P. A Visual-Inertial Fusion SLAM Method Based on Deep Learning. *China Sci. Online* **2024**, 1–8.
18. Wang, H.; Ai, K.; Zhang, Q. Monocular Visual-Inertial Simultaneous Localization and Mapping Method Based on Feature Collaboration. *Comput. Eng.* **2025**, *51*, 305–316. [\[CrossRef\]](#)
19. Fan, Z.; Zhang, L.L.; Wang, X.Y.; Shen, Y.L.; Deng, F. LiDAR, IMU, and camera fusion for simultaneous localization and map-ping: A systematic review. *Artif. Intell. Rev.* **2025**, *58*, 174. [\[CrossRef\]](#)
20. Kerbl, B.; Kopanas, G.; Leimkühler, T.; Drettakis, G. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph. (TOG)* **2023**, *42*, 139. [\[CrossRef\]](#)
21. Cai, Z.P.; Müller, M. CLNeRF: Continual learning meets NeRF. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 1–6 October 2023; pp. 23185–23194.
22. Li, B.C.; Yan, Z.K.; Wu, D.; Jiang, H.Q.; Zha, H.B. Learn to memorize and to forget: A continual learning perspective of dynamic SLAM. In Proceedings of the 18th European Conference on Computer Vision, Milan, Italy, 29 September–4 October 2024; Springer: Berlin/Heidelberg, Germany, 2024; pp. 41–57.
23. Li, X.D.; Dunkin, F.; Dezert, J. Multi-source information fusion: Progress and future. *Chin. J. Aeronaut.* **2024**, *37*, 24–58. [\[CrossRef\]](#)

24. Gao, Q.; Lu, K.-F.; Ji, Y.-H.; Liu, J.-J.; Xu, L.; Wei, G.-R. Survey on the research of multi-sensor fusion SLAM. *Mod. Radar* **2024**, *46*, 29–39.
25. Sun, X.; Zhao, Y.; Wang, Y.; Li, Z.; He, Z.; Wang, X. UPL-SLAM: Unconstrained RGB-D SLAM With Accurate Point-Line Features for Visual Perception. *IEEE Access* **2024**, *13*, 8676–8690. [[CrossRef](#)]
26. Guo, Y.; Zhou, Y.; Huang, S.; Shuo, L.; Xie, G.; Qin, X. VI-SLAM System Based on Point-line Features in Structured Environment. *J. Mech. Eng.* **2024**, *60*, 296–305. [[CrossRef](#)]
27. Pumarola, A.; Vakhitov, A.; Agudo, A.; Sanfeliu, A.; Moreno-Noguer, F. PL-SLAM: Real-time monocular visual SLAM with points and lines. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation(ICRA), Singapore, 29 May–3 June 2017; pp. 4503–4508.
28. Vongioi, R.G.; Jakubowicz, J.; Morel, J.-M.; Randall, G. LSD: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *32*, 722–732. [[CrossRef](#)] [[PubMed](#)]
29. Akinlar, C.; Topal, C. EDLines: A real-time line segment detector with a false detection control. *Pattern Recognit. Lett.* **2011**, *32*, 1633–1642. [[CrossRef](#)]
30. Lee, J.H.; Lee, S.; Zhang, G.; Lim, J.; Chung, W.K.; Suh, I.H. Outdoor place recognition in urban environments using straight lines. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation(ICRA), Hong Kong, China, 31 May–7 June 2014.
31. Lym, J.; Gu, G.H.; Jung, Y.; Vlachos, D.G. Lattice convolutional neural network modeling of adsorbate coverage effects. *J. Phys. Chem. C* **2019**, *123*, 18951–18959. [[CrossRef](#)]
32. Xue, N.; Wu, T.; Bai, S.; Wang, F.-D.; Xia, G.-S.; Zhang, L.; Torr, P.H.S. Holistically-Attracted Wireframe Parsing: From Supervised to Self-Supervised Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 14727–14744. [[CrossRef](#)]
33. Hu, X.; Zhu, L.; Wang, P.; Yang, H.; Li, X. An Improved Visual Algorithm Based on Multi-Feature Fusion for Mobile Robots. *IEEE Access* **2023**, *11*, 100659–100671. [[CrossRef](#)]
34. Zhang, L.; Na, W.; Yao, C.; Liu, C.; Chen, Y. Stereo Vision SLAM Based on Feature Extraction Network. In Proceedings of the 2024 Photonics & Electromagnetics Research Symposium (PIERS), Chengdu, China, 21–25 April 2024. [[CrossRef](#)]
35. Shen, L.; Li, C.; Wen, S.; Zhao, Y.; Jian, L. A Hybrid Visual SLAM through Deep Learning-based Point and Line Feature Fusion. In Proceedings of the 44th Chinese Control Conference (CCC 2025), Shanghai, China, 25–27 July 2025. [[CrossRef](#)]
36. Qu, H.; Zhang, L.; Mao, J.; Tie, J.; He, X.; Hu, X.; Shi, Y.; Chen, C. DK-SLAM: Monocular Visual SLAM with Deep Keypoint Learning, Tracking, and Loop Closing. *Appl. Sci.* **2025**, *15*, 7838. [[CrossRef](#)]
37. Lin, Z.H.; Zhang, Q.; Tian, Z.; Yu, P.; Lan, J. DPL-SLAM: Enhancing Dynamic Point-Line SLAM Through Dense Semantic Methods. *IEEE Sens. J.* **2024**, *24*, 14596–14607. [[CrossRef](#)]
38. Liang, R.G.; Yuan, J.; Kuang, B.F.; Liu, Q.; Guo, Z. DIG-SLAM: An accurate RGB-D SLAM based on instance segmentation and geometric clustering for dynamic indoor scenes. *Meas. Sci. Technol.* **2024**, *35*, 015401. [[CrossRef](#)]
39. Li, G.; Zeng, Y.; Huang, H.; Song, S.; Liu, B.; Liao, X. A Multi-Feature Fusion Slam System Attaching Semantic Invariant to Points and Lines. *Sensors* **2021**, *21*, 1196. [[CrossRef](#)] [[PubMed](#)]
40. Sha, Z.; Zhong, B.; Chen, X.; Wang, Z. DEALSD: A deep edge assisted line segment detector. *Expert Syst. Appl.* **2025**, *273*, 129417. [[CrossRef](#)]
41. Zhou, F.; Zhang, L.; Deng, C.; Fan, X. Improved Point-Line Feature Based Visual SLAM Method for Complex Environments. *Sensors* **2021**, *21*, 4604. [[CrossRef](#)] [[PubMed](#)]
42. Zhao, L.; Jin, R.; Zhu, Y.; Gao, F. A binocular inertial SLAM algorithm based on point-line feature fusion. *J. Aeronaut.* **2022**, *43*, 363–377.
43. Zhou, Z.; Li, Y.; Liu, D.; Ji, G. A video instance segmentation method based on motion tracking and feature fusion. *Comput. Technol. Dev.* **2022**, *32*, 43–49.
44. Luo, K.; Deng, J.; Cai, W.; Zhou, Y.; Zhang, J. Optimization method for line segment extraction algorithm based on Shi-Tomasi corner verification. *J. South China Norm. Univ. (Nat. Sci. Ed.)* **2022**, *54*, 113–121.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.