


Article

DFD-SLAM: Visual SLAM with Deep Features in Dynamic Environment

Wei Qian¹ , Jiansheng Peng^{1,2,3,4,*} and Hongyu Zhang¹

¹ College of Automation, Guangxi University of Science and Technology, Liuzhou 545000, China; 20220203053@stdmail.gxust.edu.cn (W.Q.); 221077113@stdmail.gxust.edu.cn (H.Z.)

² Department of Artificial Intelligence and Manufacturing, Hechi University, Hechi 547000, China

³ Key Laboratory of AI and Information Processing, Hechi University, Education Department of Guangxi Zhuang Autonomous Region, Hechi 547000, China

⁴ Guangxi Key Laboratory of Sericulture Ecology and Applied Intelligent Technology, School of Chemistry and Bioengineering, Hechi University, Hechi 546300, China

* Correspondence: pengjs@hcnu.edu.cn

Abstract: Visual SLAM technology is one of the important technologies for mobile robots. Existing feature-based visual SLAM techniques suffer from tracking and loop closure performance degradation in complex environments. We propose the DFD-SLAM system to ensure outstanding accuracy and robustness across diverse environments. Initially, building on the ORB-SLAM3 system, we replace the original feature extraction component with the HFNet network and introduce a frame rotation estimation method. This method determines the rotation angles between consecutive frames to select superior local descriptors. Furthermore, we utilize CNN-extracted global descriptors to replace the bag-of-words approach. Subsequently, we develop a precise removal strategy, combining semantic information from YOLOv8 to accurately eliminate dynamic feature points. In the TUM-VI dataset, DFD-SLAM shows an improvement over ORB-SLAM3 of 29.24% in the corridor sequences, 40.07% in the magistrale sequences, 28.75% in the room sequences, and 35.26% in the slides sequences. In the TUM-RGBD dataset, DFD-SLAM demonstrates a 91.57% improvement over ORB-SLAM3 in highly dynamic scenarios. This demonstrates the effectiveness of our approach.

Keywords: visual SLAM; deep features; dynamic SLAM; YOLOv8; HFNet



Citation: Qian, W.; Peng, J.; Zhang, H. DFD-SLAM: Visual SLAM with Deep Features in Dynamic Environment. *Appl. Sci.* **2024**, *14*, 4949. <https://doi.org/10.3390/app14114949>

Academic Editor: Alessandro Gasparetto

Received: 6 May 2024

Revised: 1 June 2024

Accepted: 4 June 2024

Published: 6 June 2024



Copyright: © 2024 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) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Simultaneous Localization and Mapping (SLAM) is an important research direction in the field of robotics. Visual SLAM has become a key focus in SLAM research due to its low cost advantages. Visual SLAM systems utilize cameras as sensor inputs to extract image information for localization and mapping. Besides using monocular cameras as sensor inputs, numerous studies also integrate IMU data to enhance accuracy or employ various cameras for more accurate depth information [1]. Visual SLAM has seen numerous significant feature-based contributions, including the VINS-MONO and ORB-SLAM [2,3]. These classic studies provide excellent system robustness and accuracy.

However, this feature-based visual SLAM system is heavily influenced by the quality of feature extraction. First, manually extracted features are less robust to the environment with changing light conditions, and global features extracted with bag-of-words (BOW) can disrupt spatial information in scenes [4], reducing loop closure performance [5]. Second, dynamic feature points in dynamic environments significantly interfere with the system's accuracy and robustness [6]. Today, some excellent work has begun to address these potential issues using deep learning. For the first issue, some studies use deep learning to extract feature points, achieving better tracking accuracy [7]. For the second issue, some use deep-learning-derived semantic information to remove dynamic feature points,

enhancing system robustness. However, these solutions still have problems. In HFNet-SLAM [7], issues with poor rotation robustness in CNN-based feature extraction networks are mentioned. Moreover, in Crowd-SLAM, the authors found that removing too many feature points can reduce accuracy [8]. This problem was also confirmed by the authors of CDS-SLAM [9].

To tackle these problems, this paper proposes a precise and robust indoor SLAM system, supporting monocular, monocular-inertial, and RGB-D inputs. By leveraging TensorRT for accelerated model inference [10], the system integrates HFNet for feature extraction and utilizes YOLOv8 for semantic information, harnessing the advantages of deep features to adapt to dynamic environments [11,12]. The design of the system circumvents issues of deep feature loss under rotation, effectively combining manual and deep learning descriptors. The specific contributions of this paper are as follows:

1. A real-time dynamic SLAM system based on the ORB-SLAM3 framework is proposed, supporting multiple sensor input modalities. Utilizing HFNet for both local and global feature extraction significantly enhances the tracking and loop closure performance of ORB-SLAM3. Simultaneously, YOLOv8 contributes to the precise removal of feature points, leveraging semantic information. The system maximizes the benefits of deep features extraction in dynamic SLAM settings.
2. A frame rotation estimation method is introduced, where geometric consistency detection calculates possible rotation centers based on optic flow vectors to determine the appropriate usage of descriptors generated by HFNet or re-extraction of ORB descriptors under different circumstances. The system effectively combines the advantages of deep features and traditional manual methods.
3. A better feature point removal strategy is proposed, integrating geometric consistency detection to accurately filter semantic information from YOLOv8 and ensure precise removal of dynamic feature points, avoiding over-removal scenarios.

The paper is structured as follows: Section 2 introduces related work and discusses the differences in our approach. Section 3 provides a detailed overview of the system's construction and the specific algorithmic processes. Section 4.1 introduces the experimental design and evaluation metrics. Section 4.2 details the performance of DFD-SLAM on the TUM-VI dataset, analyzing its tracking and loop closure capabilities. Section 4.3 thoroughly tests the performance of DFD-SLAM on the TUM-RGBD dataset and examines its removal strategy in detail. In Section 4.4, we compare the real-time performance of DFD-SLAM with other outstanding dynamic SLAM systems. Section 5 delves into discussions and future prospects for the system.

2. Related Works

2.1. Visual SLAM

The research field of visual SLAM systems has made significant progress. In its early stages, the MonoSLAM system, employing an Extended Kalman Filter (EKF) and Shi-Tomasi feature points [13], pioneered image tracking for monocular SLAM [14]. Despite its simplicity, this method had low efficiency and posed challenges in accuracy improvement. The PTAM system, utilizing a different strategy, optimized only keyframes with Bundle Adjustment (BA) and separated tracking and mapping tasks into different threads, achieving higher precision and laying the foundation for BA-based SLAM research [15]. Given the high dependency of pure visual SLAM on image quality and feature points, researchers have explored combining visual SLAM with other sensors, like the IMU. VIN-Mono, by tightly coupling visual and IMU information, achieved a significant improvement in accuracy and robustness [2]. ORB-SLAM3 integrated various research ideas, introduced multiple data association patterns, and implemented the first complete multi-map system [16]. This enhancement improved the performance of loop detection and relocation. The ORB-based approach demonstrated efficiency and robustness [17]. However, challenges remain, including the reduced matching performance of manually extracted feature points in sparsely textured or dynamically changing scenes, and the limitations of bag-of-

words methods in handling complex environments [4]. Additionally, the impact of feature points from dynamic objects on tracking performance remains a concern. In response to these challenges, research into visual SLAM and dynamic SLAM systems leveraging deep features has attracted significant attention.

2.2. SLAM with Deep Features

With the widespread application of deep learning in computer vision, researchers have found that deep-learning-based local feature extraction methods exhibit significant advantages over traditional handcrafted approaches. For example, the Superpoint model, introduced by DeTone in 2019, utilizes a fully convolutional network for extracting feature points and descriptors, showcasing superior performance and robustness over traditional methods [18]. GCN-SLAM improves ORB-SLAM2's local feature extraction using the GCNv2 network, enhancing the system's tracking performance [19]. In 2021, LIFT-SLAM, integrating the LIFT network with ORB-SLAM and fine-tuned on the KITTI dataset, achieved a substantial improvement in system accuracy via automatic threshold adjustments [20].

In terms of global feature extraction, the limitations of traditional bag-of-words methods become increasingly apparent in complex environments. DXSLAM emphasizes the inefficacy of traditional bag-of-words descriptors in accurately depicting spatial relationships between objects, leading to loop detection errors and ultimately resulting in decreased performance of the SLAM system [5]. NetVLAD generated high-dimensional vector descriptors for images through end-to-end trained CNN architecture, showcasing superior performance compared to traditional methods [21]. It finds wide application in the SLAM domain, as evidenced in DOOR-SLAM and Yang's research, demonstrating enhanced robustness to lighting variations [22,23].

Despite the excellent performance of deep-learning-based feature extraction, incorporating both local and global feature extraction in SLAM requires higher computational costs. Some studies attempt to fuse local and global feature extraction to enhance efficiency. In 2020, HFNet achieved a balanced performance and real-time capability by combining MobileNet with Superpoint and NetVLAD [24], optimized through model distillation. DX-SLAM combines HFNet with ORB-SLAM2, effectively improving the system's tracking accuracy and loop detection performance. The authors also found that DX-SLAM has a certain adaptability to dynamic environments [5]. The state-of-the-art HFNet-SLAM completely replaces the feature extraction part of ORB-SLAM3. While its performance surpasses that of ORB-SLAM3, it exposes issues in descriptor loss leading to tracking failure in frame rotation scenarios [7]. Despite the advancements in accuracy and robustness with deep feature-based SLAM methods, challenges remain to be addressed.

Our work builds on the use of HFNet for local and global feature extraction by adding a new extraction and matching mechanism. By applying dynamic modules to calculate optical flow vectors, we roughly identify the frame's rotation center and potential rotation angles. This allows the system to select different descriptors for matching. This approach overcomes the issue of rotation loss that is encountered when solely using deep-learning-based descriptors.

2.3. Dynamic SLAM

Feature-based SLAM systems encounter challenges in dynamic environments due to the interference of dynamic object feature points. The mainstream strategy involves identifying and removing these dynamic feature points. Early methods used geometric approaches, leveraging dense optical flow and motion models to identify dynamic objects. For example, SUN proposed online motion removal [25], and Dai introduced a method using Delaunay triangulation and frame comparison to identify dynamic points [26]. Although these methods operate in real-time, they do not offer the robustness and accuracy characteristic of semantic approaches.

Advancements in deep learning have propelled the development of dynamic SLAM. DynaSLAM utilizes Mask R-CNN for instance segmentation to remove feature points associated with dynamic objects, along with background restoration [27]. However, DynaSLAM's reliance on PyTorch for model inference results in suboptimal real-time performance due to frame-by-frame instance segmentation. Moreover, DynaSLAM indiscriminately removes all predefined dynamic objects, lacking a precise mechanism for judgment. To improve real-time performance, RDS-SLAM segments keyframes and employs Bayesian probability propagation, enhancing real-time capability but potentially compromising removal accuracy [28]. For more precise removal of dynamic points, DS-SLAM combines semantic and geometric information, achieving accurate removal through optical flow tracking and epipolar line computation [29].

In recent approaches, many systems incorporate acceleration techniques for model inference to obtain semantic information, meeting real-time requirements. SG-SLAM employs NCNN to accelerate semantic information model inference, blending geometric data with distinct threshold decisions for varied semantic areas to facilitate dynamic feature point removal [30]. However, this geometry-focused approach could result in diminished robustness, causing tracking losses across multiple RGBD-TUM sequences. CDS-SLAM employs TensorRT for accelerated inference and designs specific strategies for indoor human detection to achieve more accurate removal [9]. This approach aids in addressing the challenges of Crowd-SLAM, characterized by excessive removal through object detection, leading to the loss of tracking. Despite these advancements, precise removal of dynamic feature points remains a significant challenge.

In fact, excessive removal can lead to performance decline, and when there are too many dynamic objects in an image, most feature points may be removed, leading to tracking loss and affecting robustness. Our method further detects each dynamic object and more precisely removes the moving parts on dynamic objects. Moreover, our system is a dynamic SLAM system based on deep features, and we also demonstrate that the application of deep features can enhance the tracking performance of dynamic SLAM.

3. Materials and Methods

3.1. System Architecture

Our proposed method enhances the ORB-SLAM3 framework by incorporating three core modules: the feature point and descriptor extraction module, the semantic segmentation module, and the geometric detection module. The feature point and descriptor extraction module uses HFNet to replace traditional ORB feature extraction, generating both local and global descriptors. The semantic segmentation module generates initial masks using YOLOv8-seg. The geometric detection module employs Lucas–Kanade optical flow tracking combined with motion consistency checks for tracking feature points and executing frame rotation. The system framework is illustrated in Figure 1.

We utilize TensorRT on the GPU for parallel inference of HFNet and YOLOv8 to conduct feature extraction and semantic segmentation. This process results in obtaining feature points along with their HFNet descriptors, as well as the semantic information and masks provided by YOLOv8-seg. The RGB input image is converted to grayscale, and an image pyramid is constructed, where HFNet extracts local features at each level. YOLOv8 infers the semantic information of the input RGB image. Subsequently, feature points are tracked using Lucas–Kanade optical flow and undergo moving consistency checks to accurately remove dynamic feature points. Optical flow vectors are also used to estimate frame rotation angles, determining whether to employ HFNet's local descriptors or recalculate ORB descriptors. If the current frame is set as a keyframe, the system utilizes HFNet to extract global features for it. The global descriptors from HFNet replace the bag-of-words approach of ORB-SLAM3 for describing keyframes, with each new keyframe being described by a 4096-dimensional floating-point vector. The system calculates the Euclidean distance between the new keyframe and the keyframes stored in the keyframe library to identify the closest keyframes as candidate keyframes. Subsequent processes remain

consistent with ORB-SLAM3. This method supports both monocular and monocular-inertial modes, with all other algorithms and processes remaining as per ORB-SLAM3. Our approach is a visual SLAM system that combines deep features with dynamic SLAM, which we call DFD-SLAM.

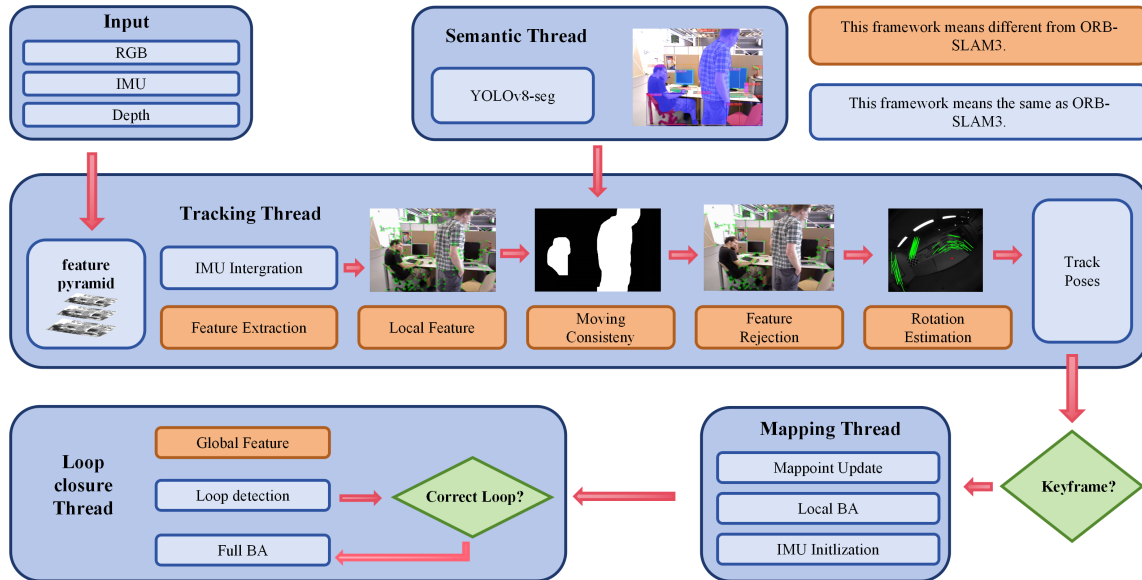


Figure 1. System architecture.

3.2. Dynamic Points Culled Algorithm

Our method combines optical flow and epipolar lines for the initial assessment of dynamic feature points, refining the removal process with semantic information. This approach draws inspiration from the LK optical flow and epipolar line motion consistency check methods, resembling the dynamic detection strategy of DS-SLAM. Initially, we apply the LK optical flow method to track feature points extracted by HFNet from the previous frame, obtaining optical flow vectors according to the results of the current frame, as illustrated in Figure 2b. The points successfully tracked across the previous and current frames are denoted as P_1 and P_2 , as depicted in Equation (1), where u and v represent pixel coordinates.

$$P_1 = [u_1, v_1, 1] \quad P_2 = [u_2, v_2, 1] \tag{1}$$

After applying the RANSAC algorithm to filter out anomalous optical flow vectors and obtain the fundamental matrix between the two frames [31], the epipolar line L_1 can be represented as shown in Equation (2):

$$L_1 = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = FP_1 = F \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \tag{2}$$

The distance D between a pixel in the current frame and its corresponding epipolar line can be expressed as shown in Equation (3). If the distance is too far, the point is considered a potential dynamic point.

$$D = \frac{|P_2^T FP_1|}{\sqrt{\|X^2\| + \|Y^2\|}} \tag{3}$$

This method has been widely applied in various dynamic SLAM systems. Although this method can approximately identify dynamic feature points, significant errors in optical flow tracking and epipolar line calculations, coupled with issues when dynamic feature

points move along epipolar lines, necessitate its combination with semantic information. Over-removal or under-removal can severely impact the system's robustness. Despite heightened attention to this issue in recent research, a satisfactory solution remains elusive. Our proposed Algorithm 1 addresses this issue to a certain extent, facilitating more precise removal.

Algorithm 1 Dynamic Points Culled Algorithm

Input: Dynamic points P_D , Static points P_S , Detect boxes D_B , Mask; Thresholds σ_1, σ_2 ;

Output: Precise mask, Final mask;

```

1: for each  $db$  in  $D_B$  do
2:   Divide  $db$  into nine boxes  $DDB_n$ ;
3:   for each  $ddb$  in  $DDB_n$  do
4:     Initialize  $Dib = \{\}, Sib = \{\}$ ;
5:      $Dib \leftarrow \text{AppendTheDynamicPoints}(P_D, ddb)$ ;
6:      $Sib \leftarrow \text{AppendTheStaticPoints}(P_S, ddb)$ ;
7:      $dynamicRatio \leftarrow \text{Len}(Dib) / (\text{Len}(Dib) + \text{Len}(Sib))$ ;
8:     if  $dynamicRatio < \sigma_1$  then
9:       Append  $ddb$  to  $Staticboxes$ ;
10:    else
11:      Append  $ddb$  to  $Dynamicboxes$ ;
12:    end if
13:  end for
14:  Check and merge near boxes from  $Staticboxes$  to  $Dynamicboxes$ ;
15:  if  $\text{Len}(Dynamicboxes) > \sigma_2$  then
16:     $Dynamicboxes \leftarrow Dynamicboxes \cup Staticboxes$ ;
17:  end if
18:  Remove the corresponding mask from  $dynamicboxes$ ;
19: end for

```

After the initial selection of dynamic and static points on the current frame using the optical flow–epipolar line method, the system divides the target detection boxes into nine areas, as shown in Figure 2d. Between lines 3 and 6, each box's dynamic points are evaluated and categorized into Dynamic In Box (Dib) or Static In Box (Sib). From lines 7 to 12, based on the ratio of dynamic to static points, each small box is determined to be either a static or dynamic sub-box. At line 14, the system identifies dynamic sub-boxes within the mother detection box and adjusts adjacent boxes to a dynamic state to ensure no potential dynamic regions are missed in the mask, with the final result shown on the left side of Figure 2e. Moreover, considering dynamic objects may move between frames, lines 15 to 18 assess the dynamic level of the target. If the majority of the mother box associated with the target is occupied by dynamic sub-boxes, the target is considered highly dynamic and marked for complete removal, as depicted on the right side of Figure 2e. This process (lines 2 to 18) outlines the assessment procedure for a single target.

The system evaluates all targets on the frame, preserving the masks within all dynamic sub-boxes and performing dilation. This strategy avoids probability calculations for dynamic objects and does not rely on subjective judgments, providing an accurate assessment of all potentially moving regions on the frame. It maximizes the utilization of the optical flow–epipolar line judgment method.

3.3. Frame Rotation Estimation and Feature Point Matching

The feature extraction based on HFNet provides high-quality feature points and descriptors, thereby improving matching efficiency and triangulation accuracy, ultimately enhancing the accuracy of pose estimation. However, in instances of frame rotation, as noted in the HFNet-SLAM paper [7], the performance of deep-learning-based feature extraction significantly deteriorates, resulting in feature matching failures and tracking loss.

In contrast, ORB descriptors used in ORB-SLAM3 have demonstrated excellent robustness in rotation, which is the foundation of our work.

To tackle this issue, we assess the frame rotation between the current and previous frames to decide on the descriptor to be used and to select a suitable distance calculation method for various descriptors. We utilize the optical flow vectors obtained from the previous tracking step and estimate the frame rotation angle using these vectors. The detailed process of frame rotation angle estimation is outlined in Algorithm 2.

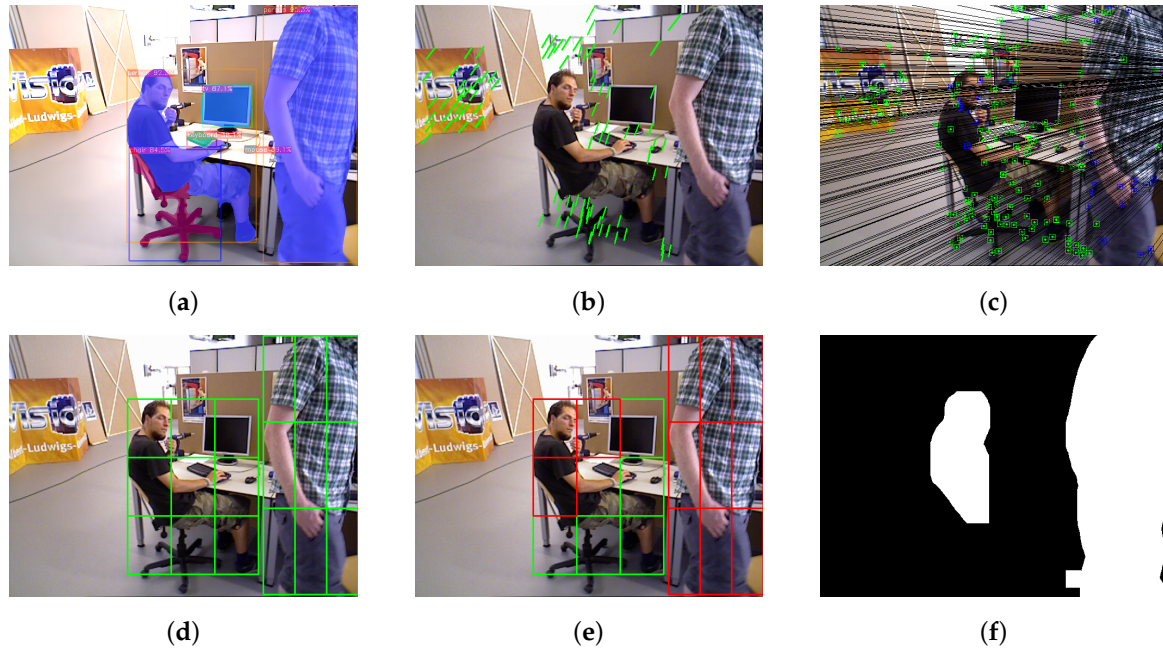


Figure 2. The complete process of precise elimination. (a) The segmentation results of YOLOv8. (b) The results of optical flow tracking on the extracted feature points. (c) The results of epipolar constraints. (d,e) The system dividing the detected potential dynamic objects into sub-frames and identifying the dynamic regions within them. In (e), the red boxes indicate dynamic regions, while the green areas indicate static regions. (f) The final retained segmentation results after dilation processing.

When a frame rotation occurs in the frame, the distributed optical flow vectors around the image's center point exhibit a characteristic pattern, as shown in the figure. While motion between frames is complex and may involve rotations along multiple axes, including frame rotation, the perpendicular bisector of the optical flow vectors may not necessarily pass exactly through the center of the frame rotation. Additionally, the tracking of optical flow vectors may not be entirely accurate. Therefore, we employ a least-squares method to optimize and solve this problem.

In Algorithm 2, the first line calculates the perpendicular bisector for each optical flow vector. Given a base optical flow vector, the midpoint and slope are used to determine the equation of its perpendicular bisector, represented by Equation (4):

$$ax + by + c = 0 \quad (4)$$

The formula represents the line on which the optical flow vector lies, where a , b , and c are the coefficients corresponding to this line. x and y represent the points on the line of the optical flow vector, including the two endpoints of the optical flow vector.

In the second line, the center point of the frame is chosen as the starting point for optimization. Since most detected and recognized frame rotations typically occur near the center of the image, this choice significantly reduces optimization time.

Algorithm 2 Rotation Estimation Algorithm

Input: Optical flow vectors FV ; Frame's dimensions H , W ; Distance threshold $DistanceThreshold$.

Output: Optimal point OP ; Rotation angle RA .

- 1: $LineParamsList \leftarrow$ List of perpendicular bisectors for each $f_v \in FV$
- 2: $InitialPoint \leftarrow (H//2, W//2)$
- 3: $OP \leftarrow$ Solve minimization problem for $InitialPoint$ using $LineParamsList$
- 4: $GoodLines \leftarrow$ Empty list
- 5: $TotalAngle \leftarrow 0$
- 6: **for** each $line$ in $LineParamsList$ **do**
- 7: Calculate distance of OP to both ends of $line$
- 8: **if** difference in distances $< DistanceThreshold$ **then**
- 9: $TotalAngle \leftarrow TotalAngle +$ angle between line endpoints
- 10: Append $line$ to $GoodLines$
- 11: **end if**
- 12: **end for**
- 13: **if** length of $GoodLines$ is sufficient **then**
- 14: $RA \leftarrow TotalAngle /$ length of $GoodLines$
- 15: **else**
- 16: Indicate rotation did not happen
- 17: **end if**

The third line constructs and solves the optimization function $f(x, y)$ as outlined in Equation (5), aiming to minimize the distance between OP and all perpendicular bisector lines:

$$f(x, y) = \sum_{i=1}^n \left(\frac{|a_i x + b_i y + c_i|}{\sqrt{a_i^2 + b_i^2}} \right)^2 \quad (5)$$

We seek a rough estimate of the frame rotation angle for real-time applications. BFGS optimization is employed for faster iteration. The gradient of the objective function is computed as shown in Equation (6), where $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are given by Equations (7) and (8):

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \quad (6)$$

$$\frac{\partial f}{\partial x} = \sum_{i=1}^n 2 \left(\frac{a_i x + b_i y + c_i}{\sqrt{a_i^2 + b_i^2}} \right) \frac{a_i}{\sqrt{a_i^2 + b_i^2}} \quad (7)$$

$$\frac{\partial f}{\partial y} = \sum_{i=1}^n 2 \left(\frac{a_i x + b_i y + c_i}{\sqrt{a_i^2 + b_i^2}} \right) \frac{b_i}{\sqrt{a_i^2 + b_i^2}} \quad (8)$$

Each update of OP occurs in a specified direction, allowing $f(x, y)$ to converge quickly. The updated $f(x, y)$ is represented by Equation (9):

$$f(x_k + \alpha_k p_{k,x}, y_k + \alpha_k p_{k,y}) \quad (9)$$

Here, $p_{k,x}$ and $p_{k,y}$ are the iteration directions, determined by Equation (10):

$$p_k = -B_k^{-1} \nabla f(x_k, y_k) \quad (10)$$

The *Hessian* matrix B_k^{-1} reflects local curvature information of the objective function near the latest iteration point, providing a more accurate descent direction. The formula for B_k^{-1} is given by Equation (11):

$$B_k^{-1} = (I - \rho_{k-1} s_{k-1} y_{k-1}^T) B_{k-1}^{-1} (I - \rho_{k-1} y_{k-1} s_{k-1}^T) + \rho_{k-1} s_{k-1} s_{k-1}^T \quad (11)$$

Here, s_k represents the change vector, y_k represents the gradient change vector, and ρ_k is used to adjust the update magnitude, ensuring positivity. The specific formulas are as follows:

$$s_k = (x_{k+1} - x_k, y_{k+1} - y_k) \quad (12)$$

$$y_k = \nabla f(x_{k+1}, y_{k+1}) - \nabla f(x_k, y_k) \quad (13)$$

$$\rho_k = \frac{1}{y_k^T s_k} \quad (14)$$

After iterative optimization, $f(x_k, y_k)$ obtains the optimal OP point as the frame rotation center. Lines 6 to 12 consider that if the motion between the two frames is relatively close to a frame rotation, the distances from OP to the ends of all optical flow vectors should be similar. As shown in Figure 3, there will be a certain difference $dist$ between the distance from P_3 to OP and the distance from P_4 to OP . If $dist$ is too large, this vector will be filtered out. This process is expressed in Equation (15), where OP is the frame rotation center, and $P1$ and $P2$ are the two endpoints of the optical flow vector.

$$\|(OP - P1)\|_2 - \|(OP - P2)\|_2 \leq \text{Distance threshold} \quad (15)$$

Therefore, lines 13 to 17 utilize this principle to filter some optical flow vectors. If too many vectors are discarded, it means that the motion between the two frames does not clearly involve a frame rotation. The remaining vectors' angles from both ends to the frame rotation center are then used to estimate the rotational movement between the current and previous frames. This is shown in Equation (16), where $P1_i$ and $P2_i$ represent the two endpoints of each optical flow vector:

$$\theta = \frac{1}{n} \sum_{i=1}^n \arccos \left(\frac{(OP - P1_i) \cdot (OP - P2_i)}{\|OP - P1_i\| \|OP - P2_i\|} \right) \quad (16)$$

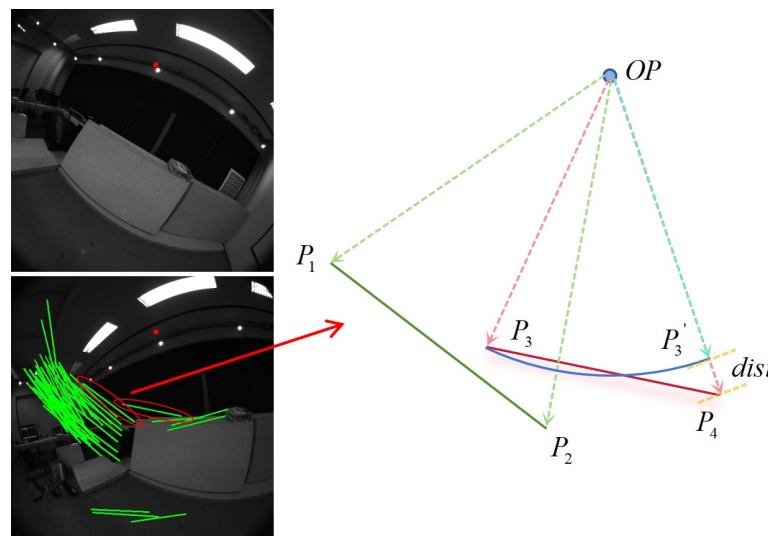


Figure 3. Filter out optical flow vectors that do not meet the requirements.

The threshold angle is set at 20 degrees because performance typically starts to degrade once estimated angles exceed 15° in testing. If the angle is bigger than the threshold angle, the system assumes that HFNet-generated descriptors should not be used due to potential frame rotation. Instead, it recalculates ORB descriptors for feature points between the current and previous frames, employing the same method as ORB-SLAM3 for subsequent matching. Conversely, if the angle is smaller, the descriptors from HFNet are used for

the current frame, and the descriptors from the previous frame are retrieved from storage. In this case, BOW is still employed to accelerate matching, but the distance calculation shifts from Hamming distance to Euclidean norm, as shown in Equation (17), where $des1$ and $des2$ represent the descriptors being matched:

$$dist = \|des1 - des2\|_2 \quad (17)$$

After this step, certain scenarios where frame rotation loss occurred due to the use of HFNet descriptors will be switched to ORB descriptors. In summary, after this step, when the system estimates that a scene has undergone rotation, it selects HFNet descriptors for previously extracted feature points on keyframes based on the estimated angle, or re-extracts ORB descriptors, and chooses the matching computation method based on whether rotation is detected. This step effectively combines the rotation robustness of handcrafted extraction methods with the accuracy advantages of deep features, allowing deep-features-based systems to navigate through scenarios where significant frame rotations could otherwise lead to a drop in matching performance and tracking loss, as shown in Figure 4.

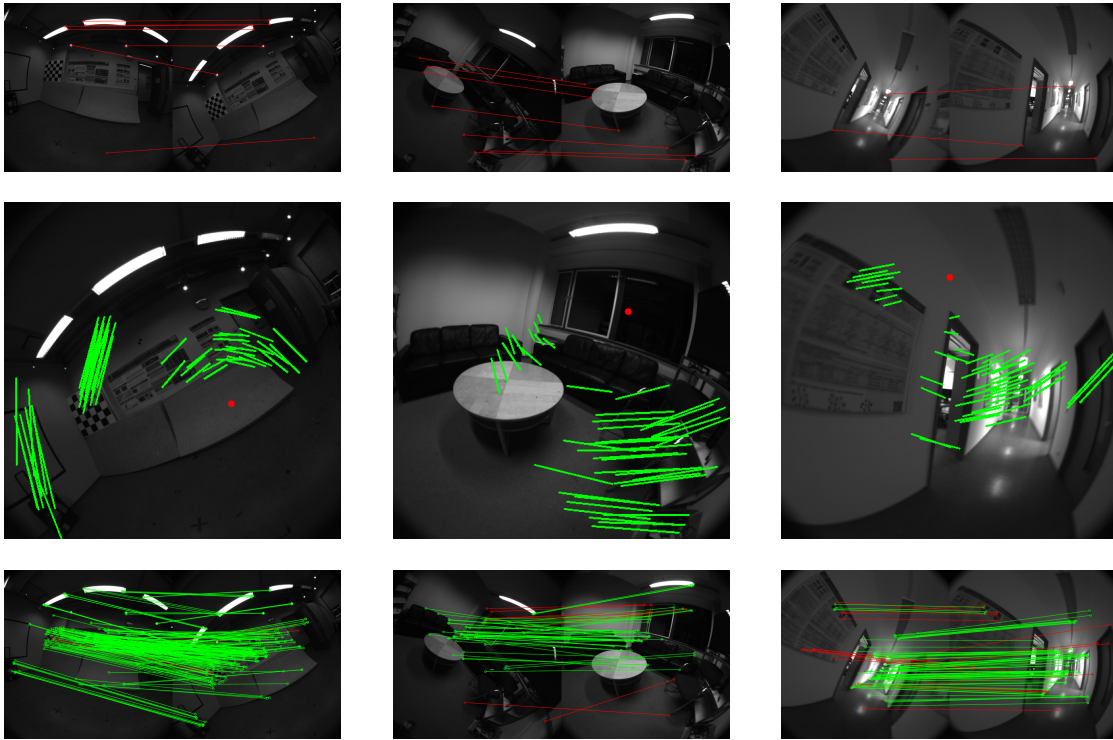


Figure 4. In a rotating scene, detect the matching situation before and after frame rotation. The first row uses HFNet descriptors. The second row is the frame identified as rotating, with red points indicating the optimized rotation center. The third row uses ORB descriptors instead.

3.4. Loop Closure

The ORB-SLAM3, on which our system is based, utilizes the bag-of-words (BOW) method for loop closure detection. However, BOW-based approaches have limited descriptive capabilities for scenes and tend to lose spatial information about depicted objects. Hence, we replace the original BOW method with global descriptors generated by HFNet. When the local mapping thread receives a new keyframe, it computes its global descriptor vector, denoted as $gobaldes1$, and saves the current keyframe's global descriptor to the keyframe library. It then calculates the Euclidean norm (ℓ_2 -norm) between the global descriptor vectors of the current keyframe and other keyframes in the library:

$$Gdist = \|gdes1 - gdes2\|_2 \quad (18)$$

Here, $gdes1$ and $gdes2$ represent the global descriptors extracted by HFNet, each consisting of 4096 floating-point numbers. After obtaining global descriptors for keyframes, the system calculates the distance with all stored global descriptors in the keyframe library. A smaller $Gdist$ indicates higher similarity between the two frames, increasing the likelihood of a loop closure. The system selects frames with the highest similarity as loop closure candidates based on their similarity with all descriptors in the keyframe library. Following this, in a manner akin to ORB-SLAM3, a geometric verification is conducted on co-visible keyframes to ascertain the occurrence of a loop closure. Our approach effectively leverages the global descriptor capabilities of HFNet, providing more accurate descriptions of keyframes compared to ORB-SLAM3 in experiments. This leads to more accurate loop closure detection, thereby mitigating substantial trajectory deviations.

4. Results

4.1. Experiment Introduction

We evaluated DFD-SLAM's performance on TUM-VI and TUM-RGBD datasets against leading visual SLAM algorithms [32,33]. TUM-VI features complex scenes with frame rotations and loop closures, while TUM-RGBD focuses on indoor dynamics. On the TUM-VI dataset, we compared ORB-SLAM3 [16], VINS-Mono [2], and HFNet-SLAM [7]. ORB-SLAM3 serves as our benchmark framework, and the improvements in DFD-SLAM are based on ORB-SLAM3. VINS-Mono is a classic visual-inertial SLAM system. HFNet-SLAM, a recently developed SLAM system based on HFNet, demonstrates exceptional performance and is a similar type of work to DFD-SLAM when dealing with static environments. In the TUM-RGBD dataset, we compared our system with ORB-SLAM3 (O3) [16], DynaSLAM (Dyna) [27], DS-SLAM (DS) [29], Crowd-SLAM (Crowd) [8], Lccrf (Lccrf) [34], CDS-SLAM (CDS) [9], and PR-SLAM (PR) [35] systems. Among them, DynaSLAM and DS-SLAM are classic dynamic SLAM systems, and even today, DynaSLAM still demonstrates outstanding practical accuracy. Crowd-SLAM highlights the degradation of tracking accuracy due to excessive feature point removal, making it an important comparison object. Lccrf applies an innovative method to ensure accuracy in dynamic environments and has high real-time performance. CDS-SLAM and PR-SLAM are the newest dynamic SLAM systems before the publication of this paper, showing relatively superior accuracy and real-time performance. We use absolute trajectory error (ATE) and relative pose error (RPE) to evaluate trajectories. ATE and RPE are critical metrics in SLAM systems as they provide comprehensive insights into the global consistency and local accuracy of the estimated trajectory, respectively. Calculating the RMSE (Root Mean Square Error) for these metrics allows for quantifiable and comparable evaluation of a SLAM system's precision and robustness in different scenarios. ATE is the Euclidean distance between transformed trajectories, representing the absolute error between two trajectory paths. RPE measures the pose differences at regular time intervals in the transformed trajectory, providing a finer representation of the accuracy of the SLAM system. All calculations are performed using Root Mean Square Error (RMSE). When performing a quantitative analysis of the improvement in absolute trajectory error before and after the enhancement, we use the following formula for calculation. Let the value before the improvement be ATE_{pre} and the value after the improvement be ATE_{cur} . The calculation method for the improvement boost is shown by Equation (19):

$$boost = \frac{ATE_{pre} - ATE_{cur}}{ATE_{pre}} \quad (19)$$

In the tables below, the boost is calculated using this method. When calculating the overall improvement, the boost for different scenarios is averaged across all sequences and reported as a percentage. Tests were conducted on Ubuntu 18.04 (British company Canonical, London, UK) with an Intel-12700H CPU (Intel, Santa Clara, CA, USA), GeForce GTX 1070Ti GPU (Nvidia, Santa Clara, CA, USA), and 16 GB RAM.

4.2. Test on TUM-VI Dataset

We conducted detailed tests and comparisons on the TUM-VI dataset by isolating the semantic thread of YOLOv8 and comparing it with ORB-SLAM3 [16], VINS-Mono [2], and HFNet-SLAM [7]. ORB-SLAM3 and VINS-Mono are classic works in the field of visual SLAM, while HFNet-SLAM utilizes deep-learning-based feature extraction for front-end odometry tracking. Table 1 illustrates that SLAM systems utilizing deep-learning-based feature extraction achieve superior tracking accuracy overall. It was observed that DFD-SLAM excels over ORB-SLAM3 in scenarios with significant viewpoint changes or lighting variations, as depicted in Figure 5. Compared to the original ORB-SLAM3, our improvements show enhancements across various scenarios in the TUM-VI dataset. By calculating the improvement rate of DFD-SLAM over ORB-SLAM3 based on the data in Table 1, we found that the average improvement (boost) for different sequences is 29.24% in the *corridor* sequences, 40.07% in the *magistrale* sequences, 28.75% in the *room* sequences, and 35.26% in the *slides* sequences. Additionally, compared to these advanced visual SLAM systems, we maintain an advantage in 14 out of the 20 sequences. This demonstrates the superiority of our system.

Table 1. Absolute trajectory error results tested in TUM-VI.

Sequence	TUM-VI (ATE)			
	ORB-SLAM3	VINS-Mono	HFNet-SLAM	DFD-SLAM
corridor1	0.04	0.63	0.023	0.018
corridor2	0.02	0.95	0.048	0.015
corridor3	0.31	1.56	0.036	0.112
corridor4	0.17	0.25	0.227	0.183
corridor5	0.03	0.77	0.051	0.027
average	0.11	0.83	0.077	0.071
Magistrale1	0.56	2.19	0.130	0.144
Magistrale2	0.52	3.11	0.471	0.319
Magistrale3	4.89	0.40	2.903	2.478
Magistrale4	0.13	5.12	0.184	0.113
Magistrale5	1.03	0.85	0.874	0.956
Magistrale6	1.30	2.29	0.604	0.547
average	1.41	2.33	0.861	0.760
Room1	0.01	0.07	0.008	0.008
Room2	0.02	0.07	0.012	0.009
Room3	0.04	0.11	0.013	0.013
Room4	0.01	0.04	0.016	0.011
Room5	0.02	0.20	0.012	0.008
Room6	0.01	0.08	0.006	0.012
average	0.02	0.10	0.011	0.010
Slides1	0.97	0.68	0.414	0.402
Slides2	1.06	0.84	0.803	0.776
Slides3	0.69	0.69	0.611	0.549
average	0.91	0.74	0.609	0.576

Our method can estimate frame rotation angles and select appropriate descriptors for matching, allowing DFD-SLAM to provide better accuracy in *corridor* sequences with frequent frame rotations. In other scenarios, although frame rotations do not result in tracking loss, they degrade matching performance and impact overall accuracy. DFD-SLAM's targeted strategy effectively identifies these frames and compensates appropriately. In the *room* scenario, due to the smaller environment and timely loop closures, there are no significant accuracy differences between ORB-SLAM3, HFNet-SLAM, and DFD-SLAM. It is only in sequences with fewer frame rotations where systems utilizing deep learning feature descriptors perform better. Overall, our method demonstrates excellent performance in

various indoor scenarios and effectively addresses the limitations of state-of-the-art deep learning feature extraction methods, leading to superior performance in specific scenarios.

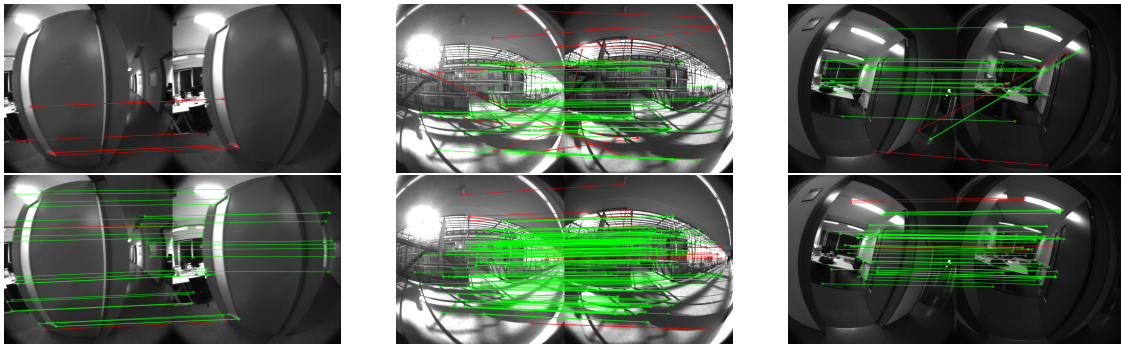


Figure 5. Matching performance of DFD-SLAM and ORB-SLAM3 under varying lighting and scene conditions. The first row shows the matching performance of ORB-SLAM3 using its strategies. The second row illustrates the matching performance of DFD-SLAM using HFNet for feature point extraction and descriptor matching. In most cases, the deep-features-based extraction method still holds advantages.

To further evaluate the impact of the frame rotation strategy on system performance, we conducted detailed ablation experiments in the *corridor* scene. The specific data are shown in the Table 2. The experiments compared DFD-SLAM using only ORB descriptors (DFD-SLAM(B)), only HFNet descriptors (DFD-SLAM(H)), and the combination of frame rotation detection (DFD-SLAM(HB)). The findings indicated that in scenes characterized by frequent frame rotations, HFNet descriptors underperformed compared to manually designed Rotated BRIEF descriptors. For instance, in the *corridor2* scene, using only HFNet descriptors is prone to tracking loss. While IMU data and relocalization can mitigate tracking loss, precision remains compromised. Despite the enhanced robustness of manually designed descriptors, they do not match the effectiveness of HFNet in terms of matching. Our DFD-SLAM(HB) strategy effectively combines the strengths of both, resulting in improved system performance. In our tests, we found that DFD-SLAM(H) performed best in the *corridor3* sequence. We think the main reason for this is that, in this sequence, even using descriptors generated by HFNet alone does not result in frame loss during frame rotations. However, such rotations typically cause frame loss in most other sequences. This is primarily because the dramatic plane rotations in this sequence do not involve significant viewpoint changes, and there are many features in these scenes, allowing the system to relocalize accurately. As a result, the final absolute trajectory error remains low. Most tracking processes without dramatic frame rotations use descriptors generated by HFNet, which perform better in feature matching. Moreover, the sequence involves many scene changes. As shown in Figure 5, HFNet demonstrates more significant advantages at these corners. This leads to the method using only descriptors generated by HFNet performing better overall.

Table 2. Ablation experimental result of absolute trajectory error in TUM-VI.

Sequence	TUM-VI (ATE)		
	DFD-SLAM(H)	DFD-SLAM(B)	DFD-SLAM(HB)
corridor1	0.024	0.031	0.018
corridor2	0.052	0.027	0.015
corridor3	0.068	0.247	0.112
corridor4	0.196	0.201	0.183
corridor5	0.039	0.058	0.027
average	0.076	0.113	0.071

We designed an ablation study to further investigate the improvement in loop closure performance of DFD-SLAM using HFNet for global descriptor generation. To eliminate the assistance of the IMU, the ablation experiment was conducted exclusively under cam0 of TUM-VI for a purely monocular test, ensuring that results were solely influenced by tracking and loop closure. By comparing DFD-SLAM and ORB-SLAM3 under different modes, we found that in most cases, the use of HFNet for global descriptor generation yields better loop closure performance, as shown in Table 3. DFD-SLAM (BOW) refers to the version that still uses bag-of-words descriptions for loop closure. DFD-SLAM (HFNet) refers to the version that uses global descriptors generated by HFNet for loop closure. Particularly in the *Magistrale5* sequence, all methods relying on bag-of-words descriptions failed to detect loop closures successfully, whereas methods utilizing global descriptors achieved timely and effective loop closure detection. Furthermore, the performance of DFD-SLAM with loop closure detection activated surpasses that of ORB-SLAM3 with loop closure detection activated in terms of accuracy. Additionally, we found that utilizing global descriptors for loop closure detection provides a significantly greater improvement in accuracy compared to using BOW-based methods, both within DFD-SLAM and relative to ORB-SLAM3. Finally, we found that the DFD-SLAM system, which employs bag-of-words descriptions, consistently outperforms ORB-SLAM3 with activated loop closure detection across several sequences. This underscores the effectiveness of our front-end tracking approach. The experiment demonstrates that our tracking and loop closure strategies significantly improve the system’s performance.

Table 3. Ablation of loop closure in TUM-VI.

Sequence	TUM-VI (ATE)							
	DFD-SLAM	DFD-SLAM (BOW)		DFD-SLAM (HFNet)		ORB-SLAM3		
	Without Loop	With Loop	Boost	With Loop	Boost	Without Loop	With Loop	Boost
Magistrale1	5.724	4.427	0.227	0.417	0.927	12.943	9.897	0.235
Magistrale2	1.153	0.879	0.238	0.716	0.379	1.120	0.757	0.324
Magistrale4	1.013	0.523	0.484	0.761	0.249	3.376	0.893	0.735
Magistrale5	2.119	2.012	0.050	1.472	0.305	2.547	2.539	0.003
Magistrale6	3.587	3.226	0.101	2.124	0.408	4.568	4.035	0.117
average	2.719	2.213	0.220	1.098	0.454	4.911	3.624	0.282

In pure monocular scenarios, we found significant accuracy discrepancies between DFD-SLAM and ORB-SLAM3 in the *Magistrale1* sequence with and without loop closure detection enabled. We have already demonstrated that DFD-SLAM’s deep feature strategy and frame rotation estimation strategy yield excellent tracking performance without loop closure. Furthermore, to investigate the substantial improvement brought by loop closure detection, we designed more precise experiments. Specifically, we statistically compared the number of loop closures between the two methods, with DFD-SLAM experiencing five loop closures and ORB-SLAM3 only encountering two loop closures, as shown in Figure 6. Figure 6a–e represent the scenes where DFD-SLAM detects loop closures during camera traversal, while Figure 6f,g represent the loop closure scenes encountered by ORB-SLAM3. Our approach is able to detect more loop closures at intersections and corners in scenes where features are not distinctly clear. The global descriptors based on deep features offer advantages over bag-of-words descriptors in such scenarios. The superior descriptive capability of global descriptors significantly enhances DFD-SLAM’s loop closure detection in environments with less distinctive features, thereby markedly improving system performance.

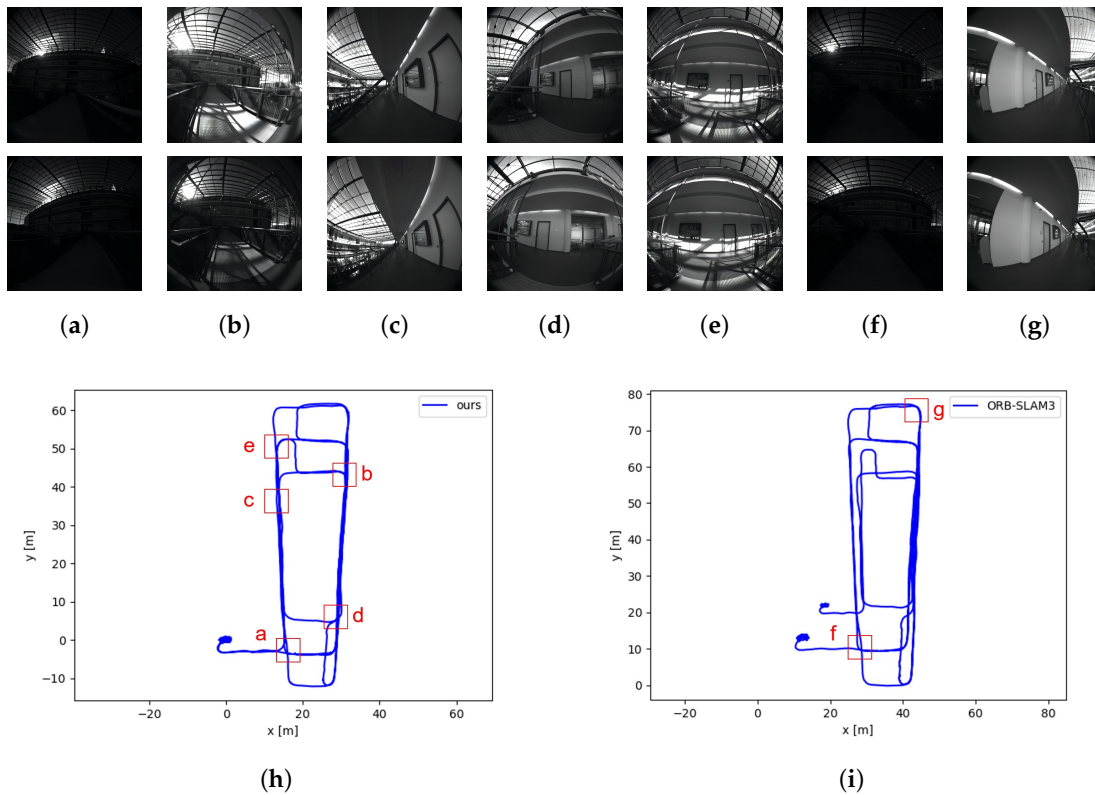


Figure 6. Comparison of loop closure detection in monocular mode. The final trajectory maps are shown in (h,i). The numbers annotated above indicate the positions where loop closure detection occurred in each system. Scenes (a–g) correspond to the occurrences of loop closure detection, where the second row indicates the frames where the systems correctly detected loop closures relative to the first row.

4.3. Test on TUM-RGBD Dataset

To validate the specific performance of DFD-SLAM in indoor dynamic environments, we compared it with several well-known dynamic SLAM systems on the TUM-RGBD dataset. This dataset provides comprehensive depth information and complete trajectory data, allowing us to verify the advantages of our novel removal method in RGBD mode. The TUM-RGBD dataset contains indoor sequences with dynamic objects, with the *Walk* sequence being our main focus due to their high dynamic environment. Additionally, we used two *Sitting* sequences to assess the system’s performance in typical dynamic environments. We compared our system with some outstanding dynamic SLAM systems and conducted detailed comparisons of ATE and RPE. All the test data are provided in Tables 4 and 5. In our test data, we achieved the best results in five out of six sequences presented in Table 4. Additionally, in the relative pose error results shown in Table 5, our approach has certain advantages over many outstanding works. We also demonstrate significant advantages in highly dynamic scenarios. Calculations show that compared to ORB-SLAM3, our improvements result in a significant reduction in absolute trajectory error in highly dynamic sequences such as *W/static*, *W/xyz*, and *W/rpy*, with an average improvement (boost) of 91.57%.

Our comparison includes Dyna-SLAM [27], DS-SLAM [29], Lccrf [34], CDS-SLAM [9], and PR-SLAM [35]. Dyna-SLAM, DS-SLAM, Crowd-SLAM, CDS-SLAM, and PR-SLAM are all dynamic SLAM systems that perform feature point selection based on semantic information; this is similar to DFD-SLAM, which primarily uses semantic information for feature point selection in dynamic environments. Across the high-dynamic scenes of the TUM-RGBD dataset, DFD-SLAM demonstrated exceptional performance in all sequences, with the exception of *W/half*, where its performance was average. Furthermore, detailed testing was conducted in our environment using open-source code. In Figure 7, we illustrate

the trajectories of ORB-SLAM3, DYNA-SLAM, CDS-SLAM, and DFD-SLAM, highlighting that our trajectory was notably smoother compared to those systems. In testing open-source systems, we found that DFD-SLAM shows a significant performance improvement over ORB-SLAM3 in dynamic environments. Additionally, it exhibits higher accuracy compared to other outstanding dynamic SLAM systems. When compared to DynaSLAM, DFD-SLAM performs better in all four highly dynamic sequences, with improvements of 10.34% in *W/half*, 17.14% in *W/rpy*, 17.67% in *W/static*, and 56.25% in *W/xyz*. In comparison with CDS-SLAM, DFD-SLAM is superior in two out of four highly dynamic sequences, with improvements of 45.28% in *W/rpy* and 46.15% in *W/xyz*, while both systems perform similarly in *W/static*. These data are also reflected in the trajectories shown in Figure 7. Compared to DynaSLAM, which is also based on semantic segmentation, our system demonstrates significantly better accuracy, as indicated by the more convergent red lines representing the error. This advantage is particularly evident in the *W/rpy* sequence, where the proportion of dynamic objects is the highest. In comparison with the advanced CDS-SLAM system, although our system shows a slight disadvantage in ATE data only in the *W/half* sequence, the actual tested trajectory of DFD-SLAM is more stable. CDS-SLAM, being based on object detection, still suffers from excessive removal or missed removal in its exclusion strategy, resulting in sudden errors in some parts of the *W/half* sequence trajectory. Moreover, in the *W/rpy* sequence, the red lines representing the error are noticeably more convergent in DFD-SLAM compared to CDS-SLAM.

Table 4. Absolute trajectory error results tested in TUM-RGBD.

Sequence	TUM-RGBD (ATE)							
	O3	Dyna	DS	Crowd	Lccrf	CDS	PR	OURS
W/half	0.424	0.029	0.030	0.026	0.028	0.019	0.025	0.026
W/rpy	0.726	0.035	0.044	0.044	0.035	0.053	0.034	0.029
W/static	0.022	0.006	0.008	0.007	0.011	0.005	0.006	0.005
W/xyz	0.825	0.016	0.024	0.020	0.016	0.013	0.017	0.007
S/half	0.019	0.018	-	0.020	-	0.013	0.015	0.011
S/xyz	0.012	0.012	-	0.018	0.009	0.011	0.007	0.007

Table 5. Relative pose error results tested in TUM-RGBD.

Sequence	TUM-RGBD (RPE)							
	O3	Dyna	DS	Crowd	Lccrf	CDS	PR	OURS
W/half	0.023	0.028	0.030	0.037	0.035	0.018	0.013	0.024
W/rpy	0.138	0.044	0.150	0.065	0.050	0.035	0.017	0.026
W/static	0.011	0.008	0.010	0.010	0.014	0.006	0.006	0.005
W/xyz	0.042	0.021	0.033	0.025	0.021	0.017	0.012	0.009
S/half	0.014	0.023	-	0.022	-	0.012	0.011	0.010
S/xyz	0.016	0.014	-	0.020	0.012	0.012	0.010	0.011

Our removal strategy played a significant role as well. In the first row of Figure 8, the man on the left exhibited subtle rotation above the waist and movement of the hands in the few preceding frames while his lower body remained stationary on the chair. Our strategy accurately retained the feature points of the man's lower body. In most cases, moving objects were accurately selected and examined to determine which parts to remove. This precise removal strategy significantly improves accuracy in scenes where dynamic objects undergo local motion. In the last row of Figure 8, some dynamic SLAM methods based on semantic segmentation would remove all objects labeled as chairs upon detecting movement of the chair on the left, such as DS-SLAM. However, our method can precisely identify which specific chair in the scene has moved.

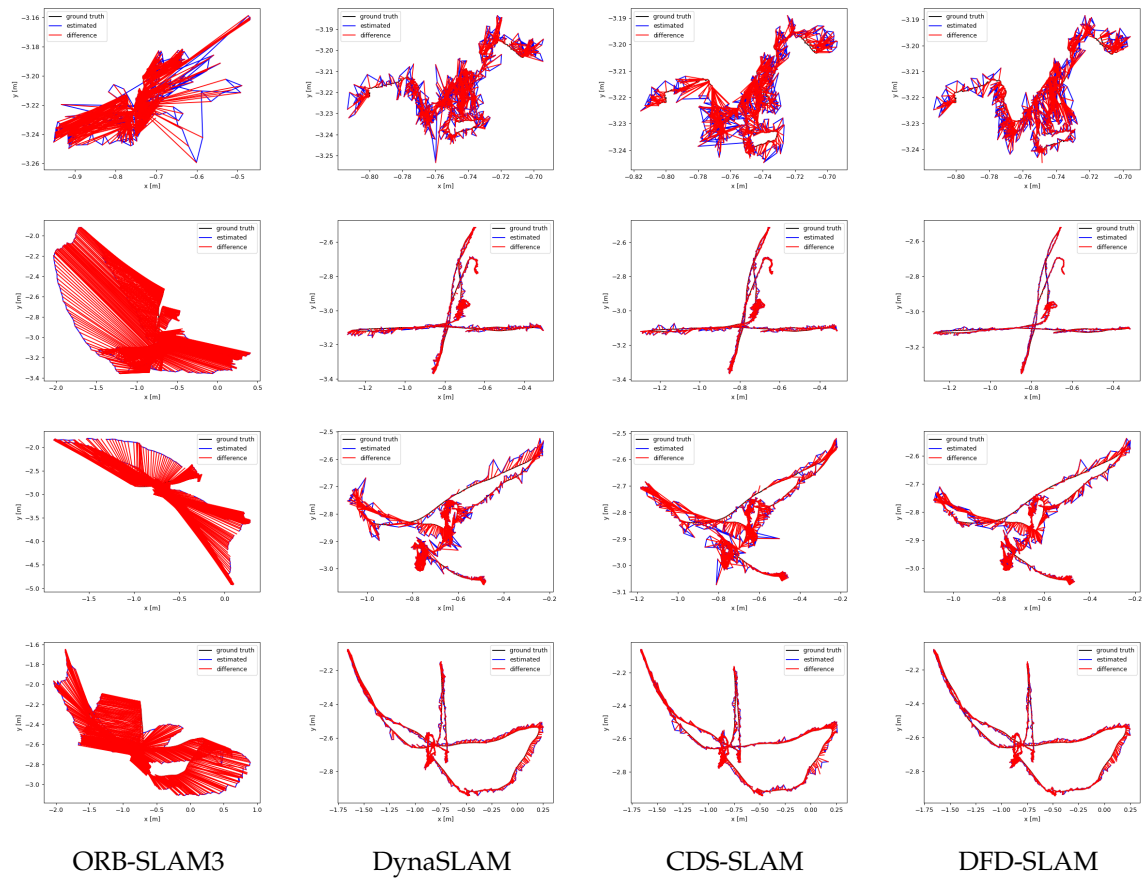


Figure 7. Comparison of trajectories between outstanding dynamic SLAM systems and our method in highly dynamic environments. The first row shows the trajectory map for the *W/static* sequence, the second row for the *W/xyz* sequence, the third row for the *W/rpy* sequence, and the fourth row for the *W/half* sequence. The blue lines represent the system's result trajectory, the black lines indicate the ground truth, and the red lines show the difference between the two. More prominent and numerous red lines indicate a higher absolute trajectory error, signifying lower tracking accuracy of the system.

To further demonstrate the effectiveness of our proposed method, we conducted more detailed ablation experiments, and the results are presented in Table 6. Here, DFD-SLAM(H) indicates no use of any removal strategy; DFD-SLAM(HS) represents complete removal of feature points related to dynamic objects; DFD-SLAM(HSD) signifies the application of our precise removal method; and DFD-SLAM(BSD) signifies combining our removal strategy with the use of ORB descriptors, and not using HFNet descriptors. Notably, in scenarios featuring significant rotations, such as *W/rpy*, employing HFNet-generated descriptors still yielded enhanced accuracy. This demonstrates that DFD-SLAM not only relies on superior removal strategies but also exhibits enhanced performance in feature point extraction descriptors, leading to overall accuracy improvements.

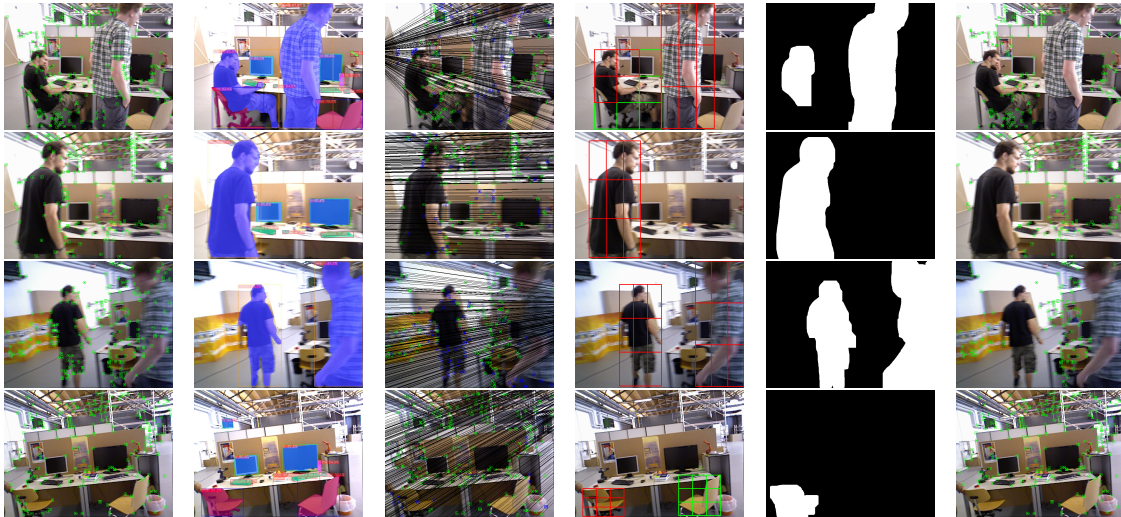


Figure 8. Dynamic point culling flowchart in *W/rpy* sequence. Each of these lines represents a complete culling process. Each column represents a cull step.

Table 6. Ablation experimental result of absolute trajectory error in TUM-RGBD.

Sequence	TUM-RGBD (ATE)						
	DFD (H)	DFD (HS)		DFD (HSD)		DFD (BSD)	
	ATE	ATE	Boost	ATE	Boost	ATE	Boost
W/half	0.162	0.024	0.852	0.026	0.840	0.028	0.827
W/rpy	0.117	0.032	0.726	0.029	0.752	0.035	0.701
W/static	0.021	0.006	0.714	0.005	0.762	0.008	0.619
W/xyz	0.077	0.009	0.883	0.007	0.909	0.011	0.857
average	0.094	0.018	0.794	0.017	0.816	0.021	0.751

4.4. Computation Cost

We also compared the real-time performance of several semantic-based dynamic SLAM methods. The specific tracking times and experimental equipment are listed in Table 7. Despite our lower computational power, our method maintains an average tracking speed of 20FPS on the TUM-RGBD datasets. Considering the real-time requirements, we employed HFNet and YOLOv8, deploying them with TensorRT for performance optimization. Tasks were allocated to two CUDA streams for parallel processing, ensuring no conflicts between semantic information retrieval and feature point extraction. While subsequent geometric processing might be complex and potentially lengthen processing times, in most scenarios without complex rotations, directly utilizing HFNet-generated descriptors ensures that overall processing times remain manageable.

Table 7. Computational cost.

Systems	Tracking Cost (ms)	Hardware
ORB-SLAM3	18.92	Intel12700h (Intel)
CDS-SLAM	37.96	Ryzen7-5800H RTX3070 (AMD Santa Clara, CA, USA) (Nvidia)
DynaSLAM	195.00	Nvidia Tesla M40 GPU (Nvidia)
PR-SLAM	50–60	R5-3600 RTX3070 (AMD) (Nvidia)
Ours	47.83	Intel12700h GTX1070TI (Intel) (Nvidia)

5. Conclusions

In this paper, we present a dynamic SLAM system based on deep features, named DFD-SLAM. The system utilizes HFNet for feature extraction combined with semantic in-

formation from YOLOv8, addressing the shortcomings of related works. This combination ensures high precision and robust performance across various environments. We introduce a frame rotation estimation strategy to allow the system to select different descriptors for appropriate scenes. Additionally, we developed a more accurate strategy for dynamic feature point elimination. Experimental results demonstrate that our deep features integrated with frame rotation estimation outperform traditional manual extraction and pure deep feature extraction in terms of tracking accuracy in static environments. In dynamic environments, our dynamic feature point elimination strategy is more precise in excluding dynamic feature points, without over- or under-eliminating them. These advancements enable our system to achieve the highest accuracy among similar outstanding works. Moreover, we accelerated inference with TensorRT, ensuring that this dual-model system maintains good real-time performance with GPU acceleration. We believe that there is still some method to improve the real-time performance. Future work could consider integrating semantic information from keyframes, similar to PR-SLAM, combined with dynamic probability for dynamic point elimination. This approach could significantly reduce computational time and load, potentially enabling operation on less powerful mobile platforms.

Author Contributions: Conceptualization, W.Q. and J.P.; methodology, W.Q., J.P. and H.Z.; software, W.Q.; validation, W.Q., J.P. and H.Z.; formal analysis, W.Q.; investigation, J.P.; resources, J.P.; data curation, W.Q.; writing—original draft preparation, W.Q. and H.Z.; writing—review and editing, W.Q., J.P. and H.Z.; visualization, W.Q.; supervision, J.P.; project administration, J.P. All authors have read and agreed to the published version of the manuscript.

Funding: The authors are highly thankful to the National Natural Science Foundation of China (No. 62063006), the Natural Science Foundation of Guangxi Province (No. 2023GXNSFAA026025), the Innovation Fund of Chinese Universities Industry-University-Research (ID: 2023RY018), to the Research Project for Young and Middle-Aged Teachers in Guangxi Universities (ID: 2020KY15013), and the Special Research Project of Hechi University (ID: 2021GCC028).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Acknowledgments: Guangxi Colleges and Universities Key Laboratory of AI and Information Processing (Hechi University), Education Department of Guangxi Zhuang Autonomous Region.

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

References

1. Kazerouni, I.A.; Fitzgerald, L.; Dooly, G.; Toal, D. A Survey of State-of-the-Art on Visual SLAM. *Expert Syst. Appl.* **2022**, *205*, 117734. [\[CrossRef\]](#)
2. Qin, T.; Li, P.; Shen, S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [\[CrossRef\]](#)
3. Mur-Artal, R.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [\[CrossRef\]](#)
4. Gálvez-López, D.; Tardós, J.D. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. Robot.* **2012**, *28*, 1188–1197. [\[CrossRef\]](#)
5. Li, D.; Shi, X.; Long, Q.; Liu, S.; Yang, W.; Wang, F.; Wei, Q.; Qiao, F. DXSLAM: A robust and efficient visual SLAM system with deep features. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 4958–4965. [\[CrossRef\]](#)
6. Pu, H.; Luo, J.; Wang, G.; Huang, T.; Liu, H. Visual SLAM integration with semantic segmentation and deep learning: A review. *IEEE Sens. J.* **2023**, *23*, 22119–22138. [\[CrossRef\]](#)
7. Liu, L.; Aitken, J.M. HFNet-SLAM: An Accurate and Real-Time Monocular SLAM System with Deep Features. *Sensors* **2023**, *23*, 2113. [\[CrossRef\]](#)
8. Soares, J.C.V.; Gattass, M.; Meggiolaro, M.A. Crowd-SLAM: Visual SLAM Towards Crowded Environments using Object. *Detect. J. Intell. Robot. Syst.* **2021**, *102*, 50. [\[CrossRef\]](#)

9. Zhang, Q.; Li, C. Semantic SLAM for mobile robots in dynamic environments based on visual camera sensors. *Meas. Sci. Technol.* **2023**, *34*, 085202. [[CrossRef](#)]
10. Jeong, E.; Kim, J.; Tan, S.; Lee, J.; Ha, S. Deep learning inference parallelization on heterogeneous processors with tensorrt. *IEEE Embed. Syst. Lett.* **2021**, *14*, 15–18. [[CrossRef](#)]
11. Sarlin, P.E.; Cadena, C.; Siegwart, R.; Dymczyk, M. From coarse to fine: Robust hierarchical localization at large scale. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 12716–12725. [[CrossRef](#)]
12. YOLOv8. Available online: <https://github.com/ultralytics/ultralytics> (accessed on 10 January 2023).
13. Shi, J. Good features to track. In Proceedings of the 1994 IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 21–23 June 1994; pp. 593–600. [[CrossRef](#)]
14. Davison, A.J.; Reid, I.D.; Molton, N.D.; Stasse, O. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1052–1067. [[CrossRef](#)]
15. lein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 225–234. [[CrossRef](#)]
16. Campos, C.; Elvira, R.; Rodriguez, J.J.G.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [[CrossRef](#)]
17. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G.R. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the ICCV, Barcelona, Spain, 6–13 November 2011; pp. 2564–2571. [[CrossRef](#)]
18. DeTone, D.; Malisiewicz, T.; Rabinovich, A. Superpoint: Self-supervised interest point detection and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 224–236. [[CrossRef](#)]
19. Tang, J.; Ericson, L.; Folkesson, J.; Jensfelt, P. GCNv2: Efficient correspondence prediction for real-time SLAM. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3505–3512. [[CrossRef](#)]
20. Bruno, H.M.S.; Colombini, E.L. LIFT-SLAM: A deep-learning feature-based monocular visual SLAM method. *Neurocomputing* **2021**, *455*, 97–110. [[CrossRef](#)]
21. Arandjelovic, R.; Gronat, P.; Torii, A.; Sivic, J. NetVLAD: CNN architecture for weakly supervised place recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 5297–5307. [[CrossRef](#)]
22. Lajoie, P.Y.; Ramtoula, B.; Chang, Y.; Carlone, L.; Beltrame, G. DOOR-SLAM: Distributed, Online, and Outlier Resilient SLAM for Robotic Teams. *IEEE Robot. Autom. Lett.* **2020**, *5*, 1656–1663. [[CrossRef](#)]
23. Yang, Y.; Tang, D.; Wang, D.; Song, W.; Wang, J.; Fu, M. Multi-camera visual SLAM for off-road navigation. *Robot. Auton. Syst.* **2020**, *128*, 103505. [[CrossRef](#)]
24. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520. [[CrossRef](#)]
25. Sun, Y.; Liu, M.; Meng, M.Q.H. Motion removal for reliable RGB-D SLAM in dynamic environments. *Robot. Auton. Syst.* **2018**, *108*, 115–128. [[CrossRef](#)]
26. Dai, W.; Zhang, Y.; Li, P.; Fang, Z.; Scherer, S. RGB-D SLAM in Dynamic Environments Using Point Correlations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 373–389. [[CrossRef](#)]
27. Bescos, B.; Facil, J.M.; Civera, J.; Neira, J. DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4076–4083. [[CrossRef](#)]
28. Liu, Y.; Miura, J. RDS-SLAM: Real-Time Dynamic SLAM Using Semantic Segmentation Methods. *IEEE Access* **2021**, *9*, 23772–23785. [[CrossRef](#)]
29. Yu, C.; Liu, Z.; Liu, X.J.; Xie, F.; Yang, Y.; Wei, Q.; Fei, Q. DS-SLAM: A semantic visual SLAM towards dynamic environments. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1168–1174. [[CrossRef](#)]
30. Cheng, S.; Sun, C.; Zhang, S.; Zhang, D. SG-SLAM: A Real-Time RGB-D Visual SLAM Toward Dynamic Scenes with Semantic and Geometric Information. *IEEE Trans. Instrum. Meas.* **2022**, *72*, 1–12. [[CrossRef](#)]
31. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [[CrossRef](#)]
32. Schubert, D.; Goll, T.; Demmel, N.; Usenko, V.; Stücker, J.; Cremers, D. The TUM VI benchmark for evaluating visual-inertial odometry. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1680–1687. [[CrossRef](#)]
33. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 573–580. [[CrossRef](#)]

34. Du, Z.J.; Huang, S.S.; Mu, T.J.; Zhao, Q.; Martin, R.R.; Xu, K. Accurate dynamic SLAM using CRF-based long-term consistency. *IEEE Trans. Vis. Comput. Graph.* **2020**, *28*, 1745–1757. [[CrossRef](#)] [[PubMed](#)]
35. Zhang, H.; Peng, J.; Yang, Q. PR-SLAM: Parallel Real-Time Dynamic SLAM Method Based on Semantic Segmentation. *IEEE Access* **2024**, *12*, 36498–36514. [[CrossRef](#)]

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.