IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 13 July 2022, accepted 12 August 2022, date of publication 17 August 2022, date of current version 25 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3199350

RESEARCH ARTICLE

Real-Time Dynamic SLAM Algorithm Based on Deep Learning

PENG SU, SUYUN LUO, AND XIAOCI HUANG
School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China
Corresponding author: Suyun Luo (lsyluo@163.com)

This work was supported by the National Science Foundation of China under Grant 62101314.

ABSTRACT In the traditional visual simultaneous localization and mapping (SLAM), the strong static assumption leads to a large degradation in the accuracy of visual SLAM in dynamic environments. For this reason, many scholars incorporate semantic segmentation networks into the visual SLAM framework to extract dynamic information in images. However, most semantic segmentation networks consume a lot of computing time due to the large model size, which leads to the algorithm's inability to meet real-time requirements in practical applications. In this paper, a real-time visual SLAM algorithm based on deep learning is proposed. This novel algorithm is based on ORB-SLAM2, and a parallel semantic thread based on the lightweight object detection network YOLOv5s is designed, which enables us to get semantic information in the scene more quickly. In the tracking thread, an optimized homography matrix module is proposed, which utilizes semantic information to optimize and solve the homography matrix so that we can calculate a more accurate optical flow vector. In the optical flow module, the semantic information is used to narrow down the calculation range of the optical flow value to improve the real-time performance of the system, and the dynamic feature points in the image are removed by the optical flow mask to improve the accuracy of the system. Experimental results show that compared with ORB-SLAM2, DynaSLAM, and other excellent visual SLAM algorithms, this algorithm can effectively reduce the absolute trajectory error of visual SLAM in dynamic environments. Compared with other deep learning-based visual SLAM algorithms, the real-time performance of this algorithm is also significantly improved.

INDEX TERMS SLAM, dynamic environment, semantic, optical flow method, pose estimation.

I. INTRODUCTION

SLAM refers to the independent construction of the surrounding environment map and updates the position of the robot in real-time through sensors in an unknown environment [1]. Laser SLAM and visual SLAM can be divided into two categories according to the types of sensors carried by robots. Laser SLAM has been widely used due to its high precision and high resolution [2]. However, due to the high price of its sensors and the increasing demand for semantic information in SLAM, many scholars turn to the research of visual SLAM. Visual SLAM takes a camera as its main sensor, which is characterized by low cost, easy to installation, and obtaining a large amount of semantic information, has become a research hotspot in the field of robotics [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Ehsan Asadi.

At present, many excellent visual SLAM algorithms have been proposed, such as DTAM [4], LSD-SLAM [5], SVO [6], and ORB-SLAM2 [7]. But these algorithms are under the assumption that the environment around them is static. Since the pose estimation of the camera is based on the matching of pixels in the image, when there is a moving object in the environment, the camera moves with the pixels on the moving object in the image at the same time, which will lead to a large error in the solution matrix of the pose estimation. The accuracy of pose estimation will be also reduced. Some algorithms eliminate the influence of dynamic objects through the motion law of the camera, but most of the algorithms have harsh constraints or low model accuracy. With the development of deep learning, many scholars use semantic segmentation networks to propose dynamic objects in images, although the semantic segmentation network can effectively segment dynamic objects, the system cannot meet

the real-time requirements due to the huge size of the model.

In this paper, a parallel semantic thread is designed to extract semantic information in images. An optimized optical flow mask module is applied to remove dynamic features in the image. In the semantic thread, YOLOv5s, a target detection network with high real-time performance, is adopted to detect quasi-dynamic information. At the same time, a method of optimizing the homography matrix is designed in the tracking thread to improve the accuracy of the system. In the optimized optical flow mask module, the semantic thread, optimized homography matrix, and optical flow method work collaboratively to enhance the system with real-time performance and robustness in a dynamic environment.

The main contributions of this paper are as follows:

- A real-time visual SLAM algorithm with parallel semantic threads based on ORB-SLAM2 is proposed. The lightweight object detection network YOLOv5s is integrated into the semantic thread, this network can extract semantic information accurately and quickly.
- In the tracking thread, the homography matrix optimization module is added. The static feature points are separated by semantic information, and then the noise points are removed by the RANSAC algorithm to obtain more accurate static feature points. Finally, these static feature points are used to obtain the optimal homography matrix by the L-M algorithm.
- In the part of removing dynamic feature points, an optimized optical flow mask module is constructed. The semantic information is applied to narrow the computation range of the optical flow vector and the optical flow mask is solved by the optimized homography matrix. The dynamic feature points in continuous frames are robustly and quickly removed by the optimized optical flow mask.

II. RELATE WORK

Visual SLAM can be divided into direct and indirect methods according to different ways of estimating camera motion. The direct method estimates camera motion by minimizing the photometric error of image pixels. Newcombe *et al.* [4] proposed the DTAM system, which realized the establishment of a dense map of every frame by the direct method. Engel *et al.* [5] proposed a semi-dense visual odometer LSD-SLAM system to estimate camera motion by selecting the part with a large pixel gradient in the whole image. Forster *et al.* [6] proposed the SVO system, which is a classical sparse direct SLAM system, position estimation is carried out by FAST features in images, and its operational efficiency is very high due to the small number of estimated pixels. However, due to the high sensitivity of images to illumination changes, the robustness of the visual SLAM system using the direct method is poor due to environmental illumination changes and camera exposure. Indirect method, also called the feature point method, estimates camera pose by matching feature points between images. Mur-Artal *et al.* [7] proposed the ORB-SLAM2 system, which is the most classic visual SLAM system based on the feature point method. The system integrates three threads of tracking, local map, and loopback detection, which can effectively improve the running speed and pose accuracy of the system. In addition, repositioning after tracking failure is realized, and the interfaces of monocular, stereo and RGB-D cameras are supported. In the research of visual SLAM, many scholars have taken it as the basic framework of its algorithm. However, when there are moving objects such as dynamic pedestrians, animals, and vehicles in the environment, neither the direct method nor the indirect method can distinguish the static and dynamic pixels in the image, so the accuracy of the algorithm will be greatly degraded [8].

In a dynamic environment, in order to eliminate the influence of dynamic objects on the accuracy of the visual SLAM system, dynamic feature points in the system are generally detected and removed according to the motion law of the camera or mathematical model processing algorithm. Bakkay *et al.* [9] completed the removal of dynamic features according to the different moving speeds of dynamic feature points and static feature points at the same camera's moving speed. Kundu *et al.* [10] discriminated dynamic feature points by calculating the distance between the matched feature point and the core line in the next frame of the image through the method of multi-perspective geometry. Zou *et al.* [11] projected the feature point of the previous frame to the current frame and calculated its reprojection error. When the error was greater than a certain threshold, the feature point was regarded as a dynamic feature point and removed. Sun *et al.* [12] obtained the basic matrix through the color information of the matching feature points between two frames, estimated the dynamic feature points based on this, and then tracked these feature points with the particle filter method to achieve dynamic and static segmentation of feature points. Wang *et al.* [13] used the depth information of images for clustering, and constrained camera motion through the basic matrix to eliminate the mismatched dynamic feature points. Moratuwage *et al.* [14] proposed a random finite set based on feature graphs and measured values to track feature points and estimate probability density through Bayesian recursion to obtain estimates of dynamic features. Chivilo *et al.* [15], Handa *et al.* [16] judged the dynamic feature points in the image through the changes of optical flow values of pixel points. Li *et al.* [17] located dynamic feature points in images by moving probability propagation model. Liu *et al.* [18] used the dense optical flow method to predict semantic labels in images to obtain dynamic semantic information. These algorithms have excellent performance in static environments. The above algorithms all remove the detected dynamic feature points to improve the accuracy of the system in dynamic environments. But, most of the above algorithms cannot accurately distinguish dynamic and static feature points due to harsh constraints or low accuracy of mathematical models.

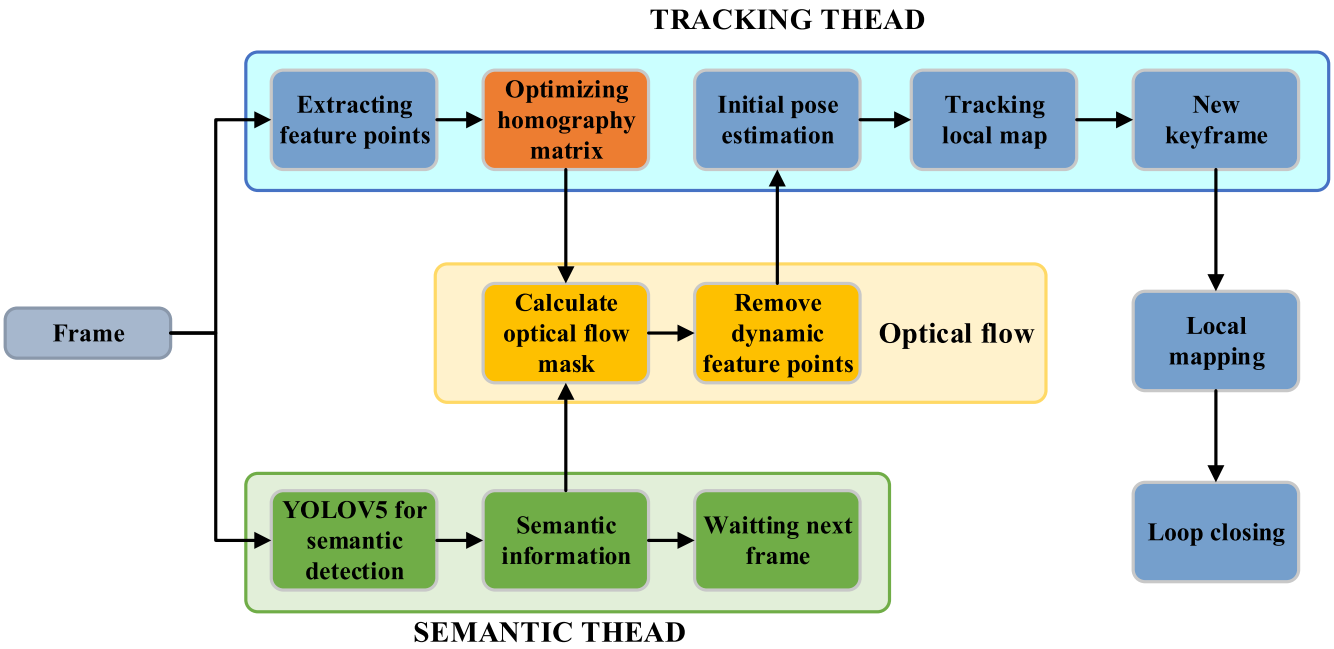

FIGURE 1. The system is built on the ORB-SLAM2 framework, and we propose parallel semantic threads to extract semantic information with YOLOv5s. We introduced the homography matrix optimization module in the tracking thread, which combines semantic information and optical flow module to cooperate to remove dynamic feature points.

With the development of deep learning technology, many scholars began to integrate neural networks into visual SLAM to obtain rich semantic information. Zhong *et al.* [19] added the neural network model of SSD (Single Shot MultiBox Detector) [20] into the SLAM framework to detect dynamic objects such as pedestrians and cars in the image, and removed all the feature points in the detection box. Zhang *et al.* [21] integrated YOLO (You Only Look Once) [22] network model into a visual SLAM framework to eliminate potential dynamic feature points. Yang *et al.* [23] used Faster R-CNN [24] to detect dynamic objects. Since static features may exist on dynamic objects, these methods can effectively eliminate dynamic features, but also cause the loss of many static features, which reduces the SLAM accuracy. Bescos *et al.* [25] proposed Dyna-SLAM, which added the semantic segmentation network Mask R-CNN [26] into the visual SLAM system, combined with the method of multi-view geometry to determine dynamic and static information, and used key frames to repair the image background after removing dynamic objects. Cheng *et al.* [27] proposed DM-SLAM, which utilized an optical flow method to remove dynamic features based on Dyna-SLAM. Yu *et al.* [28] proposed DS-SLAM, which applied semantic segmentation network SegNet [29] to extract semantic information, and can also use key frames to construct octree maps. However, the process of extracting dynamic information by semantic segmentation is time-consuming and cannot meet the real-time requirements of the system. Long *et al.* [30] proposed to eliminate dynamic feature points by combining semantic segmentation network PSPNet [31] with the multi-view geometry of optimal error compensation homography matrix, and added the reverse ant colony search strategy to the multi-view geometry to improve the real-time performance of the system. Yang *et al.* [32] proposed SGC-VSLAM, which extracts semantic detection frames through YOLO and removes dynamic features with semantic and geometric constraints. Zhang *et al.* [33] used semantic information to estimate rigid objects in the scene. Yuan *et al.* [34] constructed a word bag model using semantic tags to reduce the impact of dynamic objects on the SLAM system. However, in the case of actual industrial implementation, visual SLAM needs to satisfy both accuracy and real-time, so there is still a lot of research space for it.

Based on the research of many excellent scholars, this paper integrates the lightweight neural network model YOLOv5s into the ORB-SLAM2 system to extract semantic information from images. Meanwhile, with the help of semantic information, the images are divided into quasi-dynamic regions and quasi-static regions, and a separate thread is set up in the target detection network to improve the real-time performance of the system. In the optimized homography matrix module, the feature points of the quasi-static region are used to optimize the homography matrix of the system through the RANSAC algorithm and L-M algorithm to improve the accuracy of pose estimation. In addition, we design an optimized optical flow mask module and use the optimized homography matrix to only calculate the optical flow vector in the quasi-dynamic region of the image, which greatly saves the time of the optical flow method. The excellent robustness and real-time performance of our system are demonstrated through experiments.

III. SYSTEM DESCRIPTION

In order to solve the problems of accuracy degradation of traditional visual SLAM in dynamic environments and lack of real-time performance of deep learning-based visual SLAM, we take ORB-SLAM2 as the base framework and make some improvements. Our system framework is shown in Fig. 1.

There are three parallel threads in ORB-SLAM2, tracking, local map, and loop detection. The tracking thread mainly extracts and matches the feature points of the input RGB image frames, then estimates the camera pose by using the matched feature points, or initializes the camera pose by global repositioning, and finally optimizes the pose by tracking the local map and decides whether the current frame is a key frame or not. The local map thread is responsible for saving the key frames into the database, and then optimizing the pose of the key frames in the local map through local Bundle Adjustment (BA constraint). The loop detection thread is responsible for searching and detecting closed-loop key frames, and when a closed-loop is detected, the camera pose is optimized through the final global bundling.

In order to obtain semantic information in image frames, this paper adds semantic threads based on three threads. Semantic threads are parallel to tracking threads, which reduces the running time of the system. In the semantic thread, YOLOv5s, a lightweight real-time target detection network, is used to extract semantic information, and the semantic information of the current frame is transmitted to the tracking thread. Then, the semantic thread is suspended by a logic algorithm, and the semantic thread is awakened when the next frame enters.

In the tracking thread, this paper adds an optimized homography matrix module. Because of the dynamic features in the image, when ORB-SLAM2 randomly selects four points to calculate the homography matrix, the matrix deviation will be too large and the pose accuracy will be reduced. Therefore, this paper first selects the quasi-static feature points outside the dynamic semantic frame through semantic information, then uses the RANSAC algorithm to eliminate the external points in the quasi-static feature points, and finally utilizes the L-M optimization algorithm to get the homography matrix with minimized error.

After the optimized homography matrix is obtained, the dynamic feature points are eliminated by the optimized optical flow mask module. Using the optimized homography matrix to only calculate the optical flow vector of pixels in the quasi-dynamic region in the dynamic semantic frame can not only reduce the calculation amount, but also reduce the noise of the optical flow method. After the optical flow vector of image pixels is obtained, a threshold is set to distinguish the dynamic pixels from the static pixels. Finally, the features extracted from the dynamic pixels are eliminated, and only the static features are used to estimate the camera pose.

A. TARGE DETECTION NETWORK

At present, researchers have proposed many excellent convolutional neural network models, the most representative of which are R-CNN series and YOLO series. R-CNN series is a two-stage model, such as Fast R-CNN [35] and Faster R-CNN [24]. The first step of the two-step method is to find out the candidate frame first, and the second step is to determine the probability of the detected object in the candidate frame. This method has a slow detection speed due to the many detection steps. YOLO series is a one-stage representative network model, such as YOLOv4 and YOLOv5. The one-step method omits the step of finding out the candidate box, and directly realizes the regression of the detection results through the establishment of anchor, which not only ensures the accuracy, but also greatly improves the detection speed and is more real-time.

YOLOv5 is the latest work in the YOLO family, which is a network model proposed by Ultralytics LLC in 2020 [36]. It is based on YOLOv4 with some improvements, such as adding the Focus structure for image slicing operations in the backbone network, using two different CSP (Cross Stage Partial Network) [37] modules in Backbone and Neck, etc. And according to the size of depth and feature map width. YOLOv5 is further divided into YOLOv5s, YOLOv5m, YOLOv5l, and YOLO5x to suit different needs, and the size of the model increases from left to right model in order. Compared with YOLOv4, YOLOv5 has higher flexibility and faster detection speed.

To ensure the real-time performance of the system, YOLOv5s is selected as the feature extraction network of the visual SLAM front-end in this paper. the network structure of YOLOv5s is shown in Fig. 2, which mainly consists of four parts as follows.

(1) Input: The input side is fed with a 608*608*3 three-channel image, and the data set is enriched by Mosaic data enhancement to reduce the GPU computation. Adaptive anchor frames and adaptive image scaling are also used to enhance the flexibility of the system.

(2) Backbone: Based on YOLOv4, the Focus structure is added to the backbone network to perform slicing operations on the images and reduce the computation of the input images. The rest of the network uses the CSPDarknet53 structure, which separates the feature mapping of the base layer through the CSP module and then merges them through the cross-stage hierarchy, which can greatly reduce the size of the network model and enhance the learning ability of the convolutional neural network to meet the real-time performance while maintaining the accuracy.

(3) Neck: The connection network is the same as YOLOv4, and the multi-scale feature fusion network structure of FPN (Feature Pyramid Networks) [38] and PAN (PANet) [39] is used to sample the image high-level features and low-level features, and then the parameters are aggregated for multi-level features, which brings

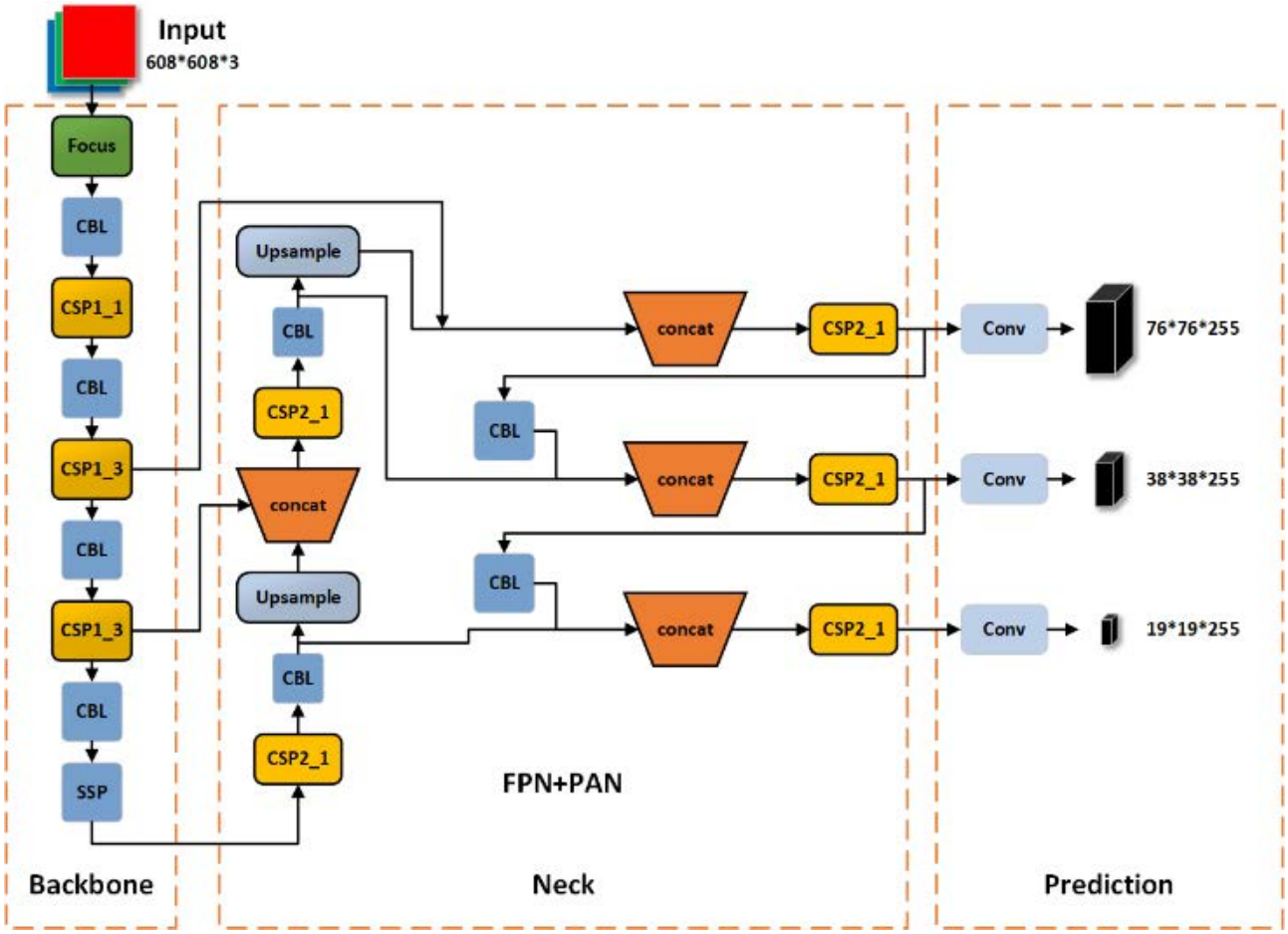

FIGURE 2. The framework of YOLOv5 consists of four parts. The image is transferred from the input layer to the backbone to extract features, and then the FPN+PAN structure is used to fuse features of different dimensions. Finally, the semantic boxes and semantic labels are obtained through the prediction layer.

the rich feature information to the prediction layer. Also, YOLOv5 uses the network structure CSP2 of CSP in Neck to enhance the image feature fusion.

(4) Prediction: YOLOv5 uses the GIOU_Loss function in the prediction classifier. Based on the IOU_Loss, the intersection scale measure is added to solve the problem when the bounding boxes do not overlap. And the semantic bounding box is generated by the weighted NMS (non-maximum suppression) method to output the classification result with the highest prediction probability [40].

To improve the system performance, we open a new thread on top of ORB-SLAM2 and add the YOLOv5s convolutional neural network built under the LibTorch framework to the new thread. The training weights are trained with the ONNX inference framework, which is an open file format designed for machine learning that allows our algorithms and models to migrate between different frameworks. We transformed the trained yolov5s.pt weight file under the Python version into yolov5s.torchscript.pt through the ONNX inference framework, and then put the generated TorchScript file into the SLAM framework for use. There are 24 categories in our training model, such as pedestrian, chair, monitor, and cup in the indoor environment. The final detection effect is shown in Fig. 3.

FIGURE 3. In the semantic thread, YOLOv5s perform semantic box extraction on objects such as pedestrians, books, tables, and chairs in key frames. The semantic box contains almost all objects that have the possibility of moving.

B. OPTIMIZED HOMOGRAPHY MATRIX MODULE

1) FEATURE EXTRACTION AND MATCHING

The feature points in the image can be regarded as the more significant points in the image, some common feature

point extraction algorithms include SIFT [41], SURF [42], FAST [35], ORB [44], and other methods. The ORB feature points use FAST feature points and combine the BRIEF descriptors to represent the attributes of these feature points. In addition to better real-time performance, ORB features also have rotation invariance. When the camera continuously collects video frames, the ORB feature has a relatively robust performance. The ORB-SLAM2 system extracts feature at different scales by building an image pyramid, making it scale-invariant. And using the method of dividing the picture into grids, the feature points of the whole picture are uniformly extracted.

After the feature points are obtained, it is necessary to find the feature point matching between adjacent frames. For the feature point method of visual SLAM, feature matching can provide better pose information for the SLAM front-end, and can also provide initial values for the SLAM back-end optimization. The feature matching in the ORB-SLAM2 system is mainly divided into three steps. First, the bag of words (BOW) vector corresponding to the current frame descriptor is calculated, then a matching threshold is set, and the BOW feature vector matching is performed to determine the optimal match. Finally, the final match is determined by counting the angle deviation of the matching descriptor.

2) HOMOGRAPHY MATRIX OF THE CAMERA

In order to estimate the camera's pose and calculate the subsequent optical flow vector, we need to know the mapping relationship between the matching feature points, so we use the matching points to calculate the homography matrix of the pose changes between adjacent frames. The definition of a homography matrix is the mapping from one plane to another.

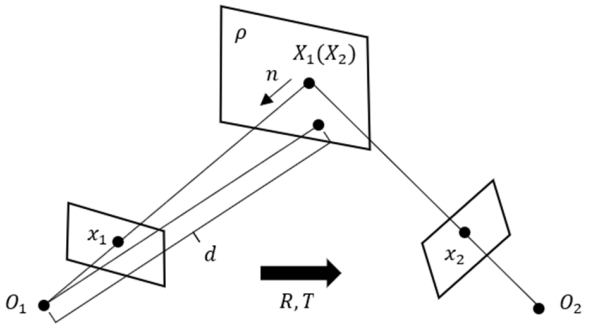

FIGURE 4. The homography matrix describes the mapping relationship between two planes. In the process of camera motion, if the same feature points of the image all fall on the same plane, the motion estimation can be performed through the homography matrix.

In Fig. 4, O_1 and O_2 are the origins of the two camera coordinate systems at different poses of the camera, and R and T denote the rotation and translation matrices of the change between the two coordinate systems. A point on the plane ρ has coordinates X_1 and X_2 under two camera coordinate systems, and the coordinates projected to the pixel coordinate system are x_1 and x_2, respectively. let the normal vector of the plane ρ under the first camera coordinate system be n and its distance to the origin of the coordinate system O_1 be d, then the plane ρ can be expressed as:

$$n^T X_1 = d \tag{1}$$

Transform the equation into:

$$\frac{1}{d} n^T X_1 = 1, \quad \forall X_1 \in \rho \tag{2}$$

At the same time, the relationship between X_1 and X_2 can be represented by the camera's rotation matrix R and translation matrix T

$$X_2 = RX_1 + T \tag{3}$$

Combining equations (2) and (3) has:

$$X_2 = RX_1 + T\frac{1}{d} n^T X_1 = \left(R + T\frac{1}{d} n^T\right) X_1 \tag{4}$$

Let the homography matrix between two camera coordinate systems be H', then H' can be expressed as:

$$H' = R + T\frac{1}{d} N^T \tag{5}$$

Combining equations (4) and (5) has:

$$X_2 = H' X_1 \tag{6}$$

At this point, the homography matrix of two different camera coordinate systems under the same plane is obtained. In order to obtain the homography matrix H between two frames in the pixel coordinate system, we should also convert X_1 and X_2 into the pixel coordinate system. Let K be the internal parameters of the camera, there are:

$$x_1 = KX_1, x_2 = KX_2 \tag{7}$$

Substituting equation (7) into equation (4) has:

$$x_2 = K^{-1}\left(R + T\frac{1}{d} N^T\right) K^{-1} x_1 = Hx_1 \tag{8}$$

In the same way, the relationship between y_1 and y_2 can be obtained. At this time, the homography matrix H between the two images is represented. Since the single response matrix is a 3×3 matrix and has scale invariance, let the scale factor be α, The normalized expression of equation (8) has:

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \alpha \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \tag{9}$$

Expanding equation (9):

$$x_2 = \alpha\left(H_{11}x_1 + H_{12}y_1 + H_{13}\right) \tag{10}$$

$$y_2 = \alpha\left(H_{21}x_1 + H_{22}y_1 + H_{23}\right) \tag{11}$$

$$1 = \alpha\left(H_{31}x_1 + H_{32}y_1 + H_{33}\right) \tag{12}$$

Combine equation (12) with equations (10) and (11) respectively:

$$x_2 = \frac{\left(H_{11}x_1 + H_{12}y_1 + H_{13}\right)}{\left(H_{31}x_1 + H_{32}y_1 + H_{33}\right)} \tag{13}$$

$$y_2 = \frac{(H_{21}x_1 + H_{22}y_1 + H_{23})}{(H_{31}x_1 + H_{32}y_1 + H_{33})} \tag{14}$$

Because the degree of freedom of the homography matrix is eight, we can solve the homography matrix of the camera by only four pairs of matching point pairs. The ORB-SLAM2 system solves the homography matrix by randomly selecting four matching point pairs. However, due to the presence of many dynamic feature points and noise points in the image, the solved homography matrix is less robust.

3) OPTIMIZING HOMOGRAPHY MATRIX

In order to reduce the influence of dynamic features and noise on solving the homography matrix, we pass the semantic information obtained by the semantic thread to the tracking thread. Since the feature points within the semantic frame are highly likely to belong to dynamic feature points, the feature point pairs within the semantic frame are regarded as hypothetical dynamic feature point pairs, and the feature point pairs outside the semantic frame are regarded as hypothesized static feature point pairs, as shown in Fig. 5.

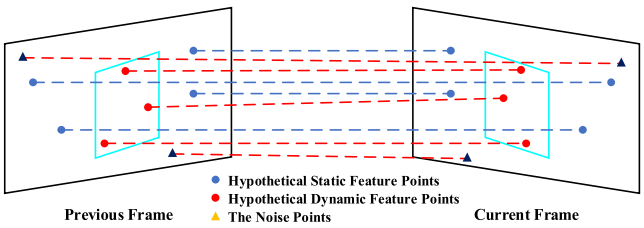

FIGURE 5. The feature points in the previous frame are projected to the current frame through the homography matrix. The blue dotted line indicates that the projection between feature points satisfies the homography transformation, while the red dotted line indicates that the projection between feature points does not. The feature points outside the semantic boxes include quasi-static feature points and noise points, and the feature points within the semantic boxes are assumed to be quasi-dynamic feature points.

Next, the homography matrix is solved using only the assumed static feature point pairs. However, in addition to dynamic features, there are also noise points generated by the camera pose transformation in the feature point pair. In order to reduce the influence of these noise points, we remove outliers through the RANSAC algorithm. RANSAC is also known as random sampling consistency. When a set of data contains many outliers, RANSAC can estimate the mathematical model of the set of data through an iterative method and eliminate outliers. Assume that the homography matrix H is the mathematical model to be iterated by RANSAC, the mathematical model is iterated by continuously substituting four random pairs of points outside the semantic detection frame, and finally, the feature point pairs that do not satisfy the mathematical model are regarded as noise point pairs and eliminated.

After obtaining more robust static feature point pairs, we need to use these feature point pairs to fit the homography matrix, and establish the error equation as shown below:

$$\varepsilon = \sum_i \left(\left(x_t - \frac{H_{11}x_{t-1} + H_{12}y_{t-1} + H_{13}}{H_{31}x_{t-1} + H_{32}y_{t-1} + H_{33}} \right) + \left(y_t - \frac{H_{21}x_{t-1} + H_{22}y_{t-1} + H_{23}}{H_{31}x_{t-1} + H_{32}y_{t-1} + H_{33}} \right) \right) \tag{15}$$

In the equation, ε represents the reprojection error of pixels, x_t and y_t represent the pixel coordinates of frame t, and x_{t-1} and y_{t-1} represent the pixel coordinates of frame $t-1$. At this point, the solution of the homography matrix changes to the least square method to solve the minimum error. Here, the homography matrix is optimized by the L-M algorithm. The L-M algorithm is also called the Levenberg-Marquardt algorithm. The incremental equation is expressed as follows:

$$\left(J_k^T J_k + I\right) \Delta x = -J_k \mathrm{f(x)} \tag{16}$$

$$H_e \approx {J_k}^T J_k + I \tag{17}$$

In the equation, f(x) represents the cost function, Δx represents the incremental iteration during the solution. The L-M algorithm uses the first derivative of the cost function with respect to the optimization variable J_k to fit the second derivative of the cost function with respect to the optimization variable H_e, and then introduces a confidence matrix I on this basis to ensure that the calculated matrix is invertible. Through this method, the optimized homography matrix H can be finally obtained. The specific algorithm flow is shown in Fig. 6.

C. OPTIMIZED OPTICAL FLOW MASK MODULE

1) OPTICAL FLOW METHOD

Optical flow refers to the instantaneous speed of motion of image pixels in space. The optical flow method calculates the motion of objects between adjacent frames through the change of image pixels in the temporal domain between two adjacent frames. There are usually two basic assumptions. One is the assumption of constant brightness, which means that the brightness of the same matching pixel on different frames will not change when the camera moves; the other is the assumption of temporal continuity, that is, during the movement of the target in adjacent frames, the position of the same matching pixel will not change drastically. After two basic assumptions are satisfied, the basic constraint equation can be established.

Let $I(x, y, t)$ be the gray value of the pixel point (x, y) at the t-th frame, and the point (x, y) moves the distance (d_x, d_y) to the next frame with d_t time, then according to the brightness invariance assumption:

$$I\,(x, y, t) = I\left(x + d_x, y + d_y, t + d_t\right) \tag{18}$$

The Taylor series expansion of the expression on the right side of the equal sign in (18) is as follows:

$$I\,(x, y, t) = I\,(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \mu \tag{19}$$

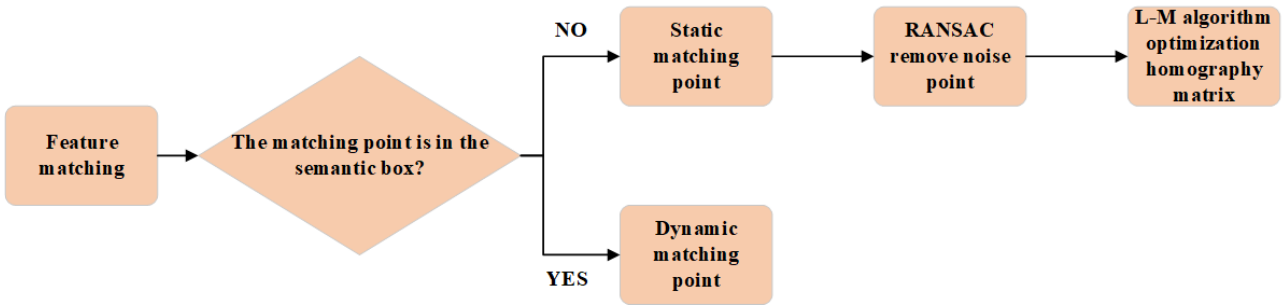

FIGURE 6. The semantic boxes are used for feature point classification. Only the classified static feature points are used to calculate the homography matrix. The noise points are eliminated by the RANSAC algorithm, and the remaining static feature points are optimized and solved by the L-M algorithm.

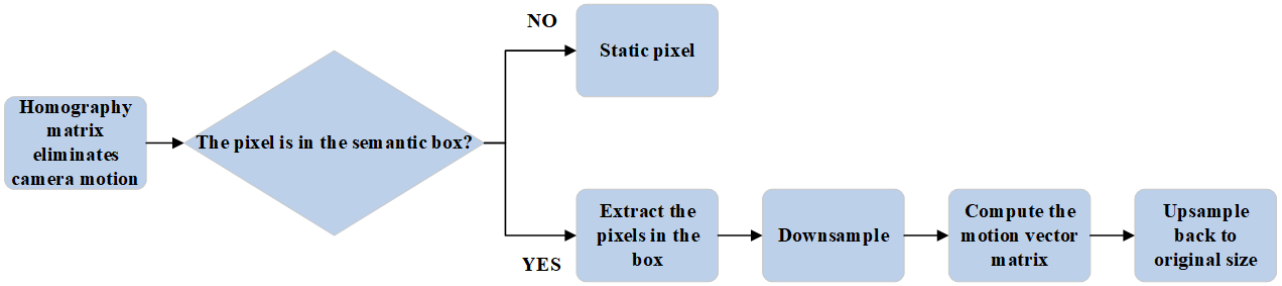

FIGURE 7. After the camera motion is calculated through the optimized homography matrix, the pixels in the semantic boxes are down-sampled to calculate the optical flow vector.

Where μ represents second-order infinitesimal and its value can be ignored. Move the left side of the equation (19) to the right. Then the equation (19) can be expressed as:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0 \tag{20}$$

Let u and v be the velocity vectors of the optical flow along the **X** and **Y** axes, respectively, then we have:

$$u = \frac{dx}{dt}, \quad v = \frac{dy}{dt} \tag{21}$$

Let $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, and $I_t = \frac{\partial I}{\partial t}$ to represent the partial derivatives of the gray value of the image pixel in the three directions of **X**, **Y**, and **T**, respectively. Then equation (20) can be transformed into:

$$I_x u + I_y v + I_t = 0 \tag{22}$$

Since there are two unknowns, we cannot solve the optical flow vector by this equation, so additional constraints need to be introduced. In the Farneback [45] optical flow algorithm, a $(2n + 1) \times (2n + 1)$ neighborhood is set for the pixels in each frame of the image, and the pixels in the neighborhood are used for least-squares fitting. The motion vector (u, v) of the pixel can be obtained.

In our system, since the camera is moving, we first need to use the optimized homography matrix H obtained in 3.2.3 to eliminate the effect of camera motion between two adjacent frames. In order to improve the real-time performance of the system and reduce the influence of optical flow noise, we only need to calculate the motion vector of the pixels in the semantic information frame in the image, and the outside of the semantic information frame is regarded as a static environment. At the same time, we improve the calculation speed of the algorithm by down-sampling the image. The specific algorithm flow is shown in Fig. 7.

2) DYNAMIC FEATURE CULLING

Since there may also be static feature points on dynamic objects, directly through semantic segmentation or culling all the feature points in the semantic information frame will cause many static feature points to be culled, which will affect the pose estimation accuracy. Therefore, we determine the range of feature point culling by calculating the optical flow value of the pixel points in the semantic information frame, which can not only reduce the calculation amount of dense optical flow, improve the real-time performance of the system, but also retain more static feature points and improve the pose estimated accuracy.

Suppose the optical flow value of a pixel is P, then P can be expressed as:

$$P = u^2 + v^2 \tag{23}$$

At this time, a mask matrix with the same size as the current frame image matrix is established, and all values in the mask matrix are set to 1. Next, the pixels in the image are traversed, and if the pixel is within the semantic information box, the optical flow value of the pixel is calculated. Since the optical

flow values of moving objects and stationary objects are quite different in the image, a threshold θ is established. If the optical flow value $P > \theta$, the pixel point is determined as the dynamic point, and the mask matrix value corresponding to this point is set to 0.

From this, a complete optical flow mask of dynamic objects in an image can be obtained. The mask is processed through erosion and dilation operations, and then the mask is passed to the tracking thread. If the feature point of the current frame falls within the optical flow mask of the current frame, it will be culled. The final effect is shown in Fig. 8. Only the feature points of the moving part of the dynamic object are removed from the semantic frame, and the static feature points of the dynamic object are completely preserved.

FIGURE 8. The value of the optical flow vector is used to determine whether the pixel is moving to avoid rejecting static features on dynamic objects. Then, use dynamic pixel to build an optical flow mask, and remove all dynamic feature points in the mask.

IV. EXPERIMENT

The experiments in this paper use the online public dataset TUM RGB-D dataset for verification. The camera pose estimated by the system is compared with the ground truth, and the Absolute trajectory Error (ATE) is used for evaluation. At the same time, we also compared the running time of other deep learning-based systems to verify the real-time performance of our system.

A. SYSTEM SETUP

The hardware configuration of the computer in the experiment is CPU: Intel(R) Xeon(R) W-2159B, GPU: GeForce RTX 3070, memory: 16G. The algorithm is based on C++, Pytorch, OpenCV and compiled under Ubuntu 16.04 operating system.

B. SYSTEM ACCURACY EVALUATION

The TUM dataset [46] is an RGB-D dataset provided by the Technical University of Munich, which contains test datasets under dynamic scenes. According to different motion conditions, it is divided into a high dynamic data set walking series containing pedestrian walking and a low dynamic data set sitting series containing pedestrian slight movement. Meanwhile, the Root Mean Squared Error (RMSE), Standard Deviation (STD), Mean Error, and Median Error are calculated to reflect the robustness and stability of the system.

TABLE 1. Comparison of absolute trajectory error of our algorithm, ORB-SLAM2, and DynaSLAM in low dynamic scenes (unit: m).

Method	RMSE	STD	Mean	Median
ORB-SLAM2	0.008	0.004	0.006	0.006
Dyna-SLAM	0.0076	0.0036	0.0067	0.0063
Ours	**0.0058**	**0.0026**	**0.0049**	**0.0048**

TABLE 2. Comparison of absolute trajectory error of our algorithm, ORB-SLAM2, and DynaSLAM in high dynamic scenes (unit: m).

Method	RMSE	STD	Mean	Median
ORB-SLAM2	0.614	0.392	0.640	0.512
Dyna-SLAM	**0.0158**	**0.0087**	**0.0132**	0.0112
Ours	0.0176	0.0090	0.0147	**0.0109**

The experiment first compares ORB-SLAM2, DynaSLAM based on deep learning and our algorithm under the low dynamic scene dataset fr3/sitting_static and the high dynamic scene dataset fr3/walking_xyz respectively. The comparison results are summarized in Table. 1 and Table. 2, The camera trajectory comparison of the three algorithms are shown in Fig. 9. the low dynamic scene fr3/sitting_static, ORB-SLAM2 performs well. However, in the high dynamic scene fr3/walking_xyz, due to the influence of dynamic objects, the trajectory offset value of ORB-SLAM2 is large. DynaSLAM performs better than our algorithm in high-dynamic scenarios fr3/walking_xyz because the DynaSLAM algorithm removes all the feature points from the semantic mask of the dynamic object. However, there are still static feature points that are not moving and can be used for pose estimation. As a result, DynaSLAM can cause poor pose estimation accuracy or even tracking failure in low-dynamic scenarios by removing too many feature points. Compared with the above two algorithms, our algorithm performs well in both the low dynamic scene and the high dynamic scene.

In order to further verify the robustness of the algorithm in this paper, the SLAM algorithm proposed in some other dynamic scenarios was compared in the experiment. These include MR-SLAM [47] based on motion segmentation; OFB-SLAM [48] based on optical flow method; and DS-SLAM, Detect-SLAM, DM-SLAM, and RDS-SLAM [49] based on semantic information. The absolute trajectory error comparison results are shown in Table. 3.

It can be seen in Table. 3 that when the environment is a low dynamic scene, the absolute trajectory error of the algorithm of this paper is smaller than that of other classical dynamic scene SLAM algorithms. In a scene where only the camera rotates, the proposed algorithm is slightly lower than OFB-SLAM. The main reason is that OFB-SLAM calculates the optical flow value of all pixels in the image, which can

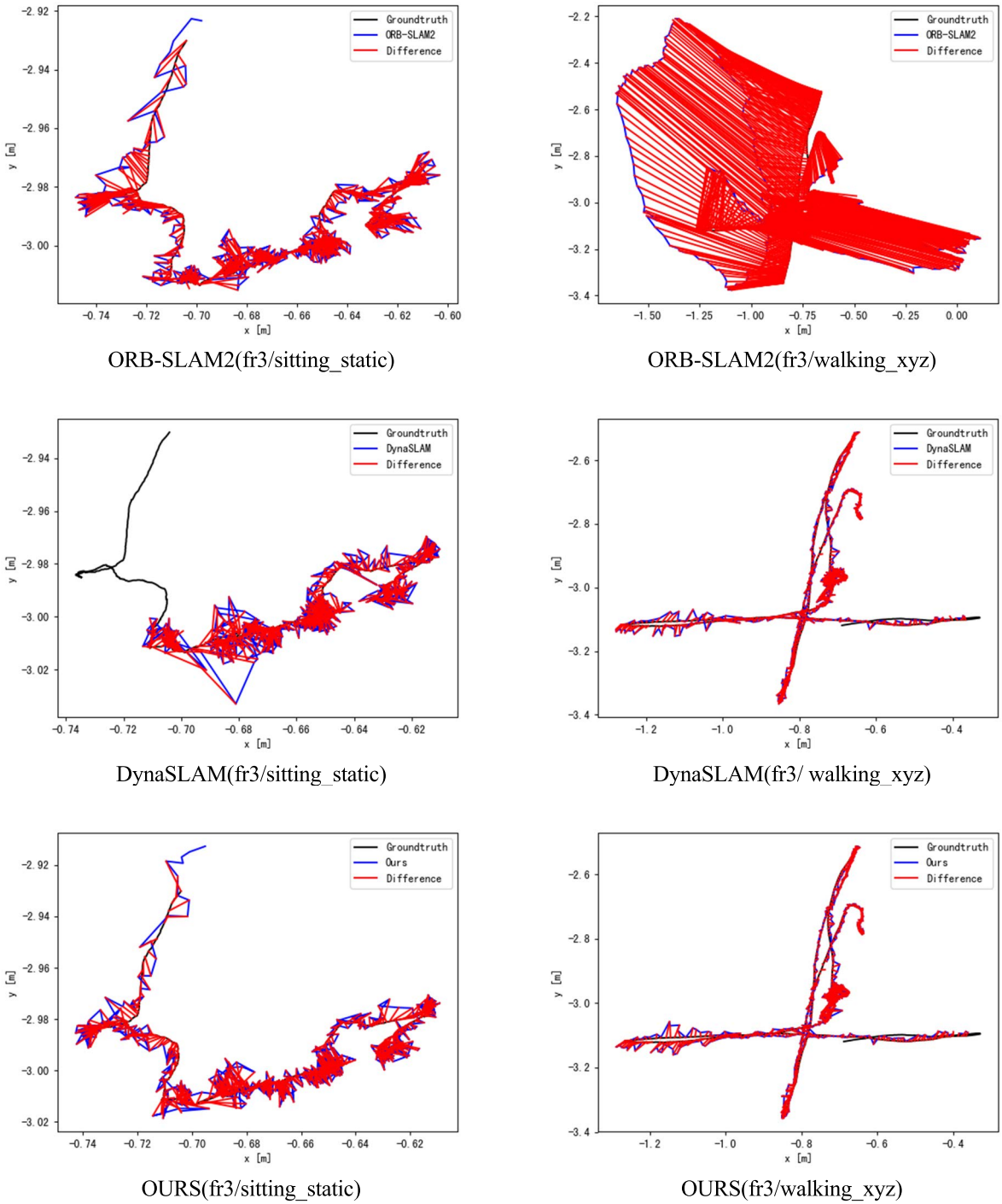

FIGURE 9. The SLAM trajectories of our algorithm, ORB-SLAM2, and DynaSLAM in dynamic and static scenes.

obtain a more accurate optical flow vector when the camera is rotating, but also increases the running time of the system and the optical flow noise when the camera is not rotating. In high dynamic scenes, compared with other excellent algorithms, the accuracy of this paper may not be the best, but the difference is not much. The main reason is that in DynaSLAM and DM-SLAM, the dynamic and static feature points are not distinguished, but all the feature points in the semantic mask are directly eliminated, which increases the fault tolerance rate of the algorithm. But at the same time, in low dynamic scenes, the accuracy of the algorithm will decline.

TABLE 3. Comparison of absolute trajectory error under different algorithms (unit: m).

Dataset	ORB-SLAM2	MR-SLAM	OFB-SLAM	Dyna-SLAM	DM-SLAM	DS-SLAM	Detect-SLAM	RDS-SLAM	Ours
fr3/sitting_static	0.008	—	0.0134	0.0076	0.0063	0.0065	—	0.0088	**0.0058**
fr3/sitting_xyz	0.009	0.0480	0.0130	0.0150	—	—	0.0201	—	**0.0087**
fr3/sitting_rpy	0.019	—	**0.0160**	0.0302	—	0.0187	—	—	**0.0189**
fr3/walking_static	0.390	0.0656	0.0410	**0.0068**	0.0079	0.0081	—	0.0815	**0.0069**
fr3/walking_xyz	0.614	0.0932	0.3060	0.0158	**0.0143**	0.0247	0.0241	0.0213	**0.0176**
fr3/walking_rpy	0.937	0.1333	0.1040	**0.0402**	0.1055	0.4442	0.2959	0.1468	**0.0612**

TABLE 4. The execution time under different deep learning-based algorithms. We use the data in their original paper as possible. If not provided in their papers, we approximate the processing time.

Method	GPU	Semantic	Detection Time(ms)	Time of Other Models Related to Tracking(ms)	Total Time for Each Frame(ms)
DynaSLAM	Nvidia Tesla M40 GPU	Mask R-CNN	195	Multi-view Geometry: 235.98 Background Inpainting: 183.56	>300
DM-SLAM	GeForce GTX 1080 Ti	Mask R-CNN	201.02	Ego-motion: 3.16 Dynamic Point Detection: 40.64	>201
Detect-SLAM	GTX960M	SSD	310	Propagation: 20 Updating: 10	>310
RDS-SLAM	GeForce RTX 2080 Ti	Mask R-CNN	200	Mask Generation: 5.42 Update Moving Probability: 0.17 Semantic-based Optimization: 0.54	50-65
DS-SLAM	P4000	SegNet	37.57	Moving consistency check: 29.50896	>65
Ours	GeForce RTX 3070	YOLOv5s	2	Homography matrix optimization: 6.45 Optical flow estimation: 36.78	**45**

C. SYSTEMS TIME CONSUMPTION

In order to verify the real-time performance of the algorithm in this paper, the experiment compares the time-consuming processing of each frame of pictures by the semantic thread of other SLAM algorithms combined with semantic information and the time of processing each frame of pictures by the tracking thread. The experimental results are shown in Table. 4.

As can be seen from Table. 4, DynaSLAM, DM-SLAM, and Detect-SLAM take a long time to process each frame of pictures due to the semantic thread, so the tracking thread takes more than 200ms to process each frame of the picture. RDS-SLAM and DS-SLAM with excellent real-time performance also take more than 50ms to process each frame of pictures in the tracking thread. Compared with other deep learning-based visual SLAM algorithms, the tracking thread of the algorithm in this paper takes only 45ms to process each frame of pictures, which can meet the real-time requirements of visual SLAM.

V. CONCLUSION

In this paper, A novel real-time visual SLAM algorithm is proposed. we apply the lightweight convolutional neural network YOLOv5s as a parallel semantic thread, which can quickly extract semantic information from the scene. In the tracking thread, we introduce a module for optimizing the homography matrix, which utilizes the semantic information box combined with the RANSAC and L-M algorithms to optimize the homography matrix in the system. In the dynamic feature point culling part, we combine semantic information, optimal homography matrix, and optical flow mask to cull dynamic feature points in the front-end of SLAM to improve the accuracy of the whole SLAM system.

In order to verify the performance of the system, we conduct experiments on the TUM dataset, and compare ORB-SLAM2, DynaSLAM, and other excellent visual SLAM algorithms. Experimental results show that our system improves in both accuracy and real-time performance.

Although the performance of the current system has improved in accuracy and real-time performance, there is still a lot of work that we need to continue to study in the future. On the one hand, we will optimize the backbone network of YOLOv5 to make the network more lightweight and reduce the time-consuming of the semantic system. On the other hand, since the sparse point cloud map constructed by the system cannot be used for navigation, we will complete the construction of the octree map in the system next, further improving the robustness and practicability of the system.

REFERENCES

[1] X. Ruan, P. Guo, and J. Huang, "Semantic visual SLAM based on deep learning in dynamic scenes," *J. Beijing Univ. Technol.*, vol. 48, no. 1, pp. 16–23, Jan. 2022, doi: 10.11936/bjutxb2020060007.

[2] Y. Fan, Q. Zhang, S. Liu, Y. Tang, X. Jing, J. Yao, and H. Han, "Semantic SLAM with more accurate point cloud map in dynamic environments," *IEEE Access*, vol. 8, pp. 112237–112252, 2020, doi: 10.1109/ACCESS.2020.3003160.

[3] K. Wang, X. Yao, Y. Huang, M. Liu, and Y. Lu, "Review of visual SLAM in dynamic environment," *Robot*, vol. 43, no. 6, pp. 715–732, Nov. 2021, doi: 10.13973/j.cnki.robot.200468.

[4] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. Int. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 2320–2327, doi: 10.1109/ICCV.2011.6126513.

[5] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Computer Vision—ECCV*, vol. 8690, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 834–849, doi: 10.1007/978-3-319-10605-2_54.

[6] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, May 2014, pp. 15–22, doi: 10.1109/ICRA.2014.6906584.

[7] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017, doi: 10.1109/TRO.2017.2705103.

[8] C. Zhang, T. Huang, and Z. Wu, "RGB-D SLAM algorithm based on K-means clustering and deep learning," *Comput. Eng.*, vol. 48, no. 1, pp. 236–244, Jul. 2022, doi: 10.19678/j.issn.1000-3428.0060052.

[9] M. C. Bakkay, M. Arafa, and E. Zagrouba, "Dense 3D SLAM in dynamic scenes using Kinect," in *Pattern Recognition and Image Analysis*, vol. 9117, R. Paredes, J. S. Cardoso, and X. M. Pardo, Eds. Cham, Switzerland: Springer, 2015, pp. 121–129, doi: 10.1007/978-3-319-19390-8_14.

[10] A. Kundu, K. M. Krishna, and J. Sivaswamy, "Moving object detection by multi-view geometric techniques from a single camera mounted robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Louis, MO, USA, Oct. 2009, pp. 4306–4312, doi: 10.1109/IROS.2009.5354227.

[11] D. Zou and P. Tan, "CoSLAM: Collaborative visual SLAM in dynamic environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 354–366, Feb. 2013, doi: 10.1109/TPAMI.2012.104.

[12] Y. Sun, M. Liu, and M. Q.-H. Meng, "Motion removal for reliable RGB-D SLAM in dynamic environments," *Robot. Auton. Syst.*, vol. 108, pp. 115–128, Oct. 2018, doi: 10.1016/j.robot.2018.07.002.

[13] R. Wang, W. Wan, Y. Wang, and K. Di, "A new RGB-D SLAM method with moving object detection for dynamic indoor scenes," *Remote Sens.*, vol. 11, no. 10, p. 1143, May 2019, doi: 10.3390/rs11101143.

[14] D. Moratuwage, B.-N. Vo, and D. Wang, "Collaborative multi-vehicle SLAM with moving object tracking," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, May 2013, pp. 5702–5708, doi: 10.1109/ICRA.2013.6631397.

[15] G. Chivilo, F. Mezzaro, A. Sgorbissa, and R. Zaccaria, "Follow-the-leader behaviour through optical flow minimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sendai, Japan, vol. 4, Sep. 2014, pp. 3182–3187, doi: 10.1109/IROS.2004.1389907.

[16] A. Handa, J. Sivaswamy, K. M. Krishna, S. Singh, and P. Menezes, "Person following with a mobile robot using a modified optical flow," in *Proc. Adv. Mobile Robot.*, Coimbra, Portugal, Aug. 2008, pp. 1154–1160, doi: 10.1142/9789812835772_0138.

[17] A. Li, J. Wang, M. Xu, and Z. Chen, "DP-SLAM: A visual SLAM with moving probability towards dynamic environments," *Inf. Sci.*, vol. 556, pp. 128–142, May 2021, doi: 10.1016/j.ins.2020.12.019.

[18] Y. Liu and J. Miura, "RDMO-SLAM: Real-time visual SLAM for dynamic environments using semantic label prediction with optical flow," *IEEE Access*, vol. 9, pp. 106981–106997, 2021, doi: 10.1109/ACCESS.2021.3100426.

[19] F. Zhong, S. Wang, Z. Zhang, C. Chen, and Y. Wang, "Detect-SLAM: Making object detection and SLAM mutually beneficial," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Lake Tahoe, NV, USA, Mar. 2018, pp. 1001–1010, doi: 10.1109/WACV.2018.00115.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision—ECCV*, vol. 9905, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.

[21] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song, "Semantic SLAM based on object detection and improved octomap," *IEEE Access*, vol. 6, pp. 75545–75559, 2018, doi: 10.1109/ACCESS.2018.2873617.

[22] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[23] S. Yang, J. Wang, G. Wang, X. Hu, M. Zhou, and Q. Liao, "Robust RGB-D SLAM in dynamic environment using faster R-CNN," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, Chengdu, Dec. 2017, pp. 2398–2402, doi: 10.1109/CompComm.2017.8322965.

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.

[25] B. Bescos, J. M. Fácil, J. Civera, and J. L. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, Oct. 2018, doi: 10.1109/LRA.2018.2860039.

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," Jan. 2020, *arXiv:1703.06870*.

[27] J. Cheng, Z. Wang, H. Zhou, L. Li, and J. Yao, "DM-SLAM: A feature-based SLAM system for rigid dynamic scenes," *ISPRS Int. J. Geo-Inf.*, vol. 9, no. 4, p. 202, Mar. 2020, doi: 10.3390/ijgi9040202.

[28] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "DS-SLAM: A semantic visual SLAM towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Oct. 2018, pp. 1168–1174, doi: 10.1109/IROS.2018.8593691.

[29] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder–decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, doi: 10.1109/TPAMI.2016.2644615.

[30] X. Long, W. Zhang, and B. Zhao, "PSPNet-SLAM: A semantic SLAM detect dynamic object by pyramid scene parsing network," *IEEE Access*, vol. 8, pp. 214685–214695, 2020, doi: 10.1109/ACCESS.2020.3041038.

[31] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 6230–6239, doi: 10.1109/CVPR.2017.660.

[32] S. Yang, G. Fan, L. Bai, C. Zhao, and D. Li, "SGC-VSLAM: A semantic and geometric constraints VSLAM for dynamic indoor environments," *Sensors*, vol. 20, no. 8, p. 2432, Apr. 2020, doi: 10.3390/s20082432.

[33] J. Zhang, M. Henein, R. Mahony, and V. Ila, "VDO-SLAM: A visual dynamic object-aware SLAM system," Jul. 2022, *arXiv:2005.11052*.

[34] Z. Yuan, K. Xu, X. Zhou, B. Deng, and Y. Ma, "SVG-loop: Semantic-visual–geometric information-based loop closure detection," *Remote Sens.*, vol. 13, no. 17, p. 3520, Sep. 2021, doi: 10.3390/rs13173520.

[35] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.

[36] Y. Chen, K. Alifu, and W. Lin, "CA-YOLOv5 for crowded pedestrian detection," *Comput. Eng. Appl.*, vol. 1, no. 1, pp. 1–10, Mar. 2022, doi 10.3778/j.issn.1002-8331.2201-0058.

[37] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Seattle, WA, USA, Jun. 2020, pp. 1571–1580, doi: 10.1109/CVPRW50498.2020.00203.

[38] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 936–944, doi: 10.1109/CVPR.2017.106.

[39] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," 2018, *arXiv:1803.01534*.

[40] X. Yang and Y. Cai, "Pedestrian detection system based on YOLOV5S and its implementation," *Comput. Inf. Technol.*, vol. 30, pp. 28–30, Feb. 2022, doi: 10.19414/j.cnki.1005-1228.2022.01.006.

[41] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[42] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Computer Vision—ECCV*, vol. 3951, A. Leonardis, H. Bischof, and A. Pinz, Eds. Heidelberg, Germany: Springer, 2006, pp. 404–417, doi: 10.1007/11744023_32.

[43] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision—ECCV*, vol. 3951, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Germany: Springer, 2006, pp. 430–443, doi: 10.1007/11744023_34.
[44] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 2564–2571, doi: 10.1109/ICCV.2011.6126544.
[45] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*, vol. 2749, J. Bigun and T. Gustavsson, Eds. Berlin, Germany: Springer, 2003, pp. 363–370, doi: 10.1007/3-540-45103-X_50.
[46] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura-Algarve, Portugal, Oct. 2012, pp. 573–580, doi: 10.1109/IROS.2012.6385773.
[47] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robot. Auton. Syst.*, vol. 89, pp. 110–122, Mar. 2017, doi: 10.1016/j.robot.2016.11.012.
[48] J. Cheng, Y. Sun, and M. Q.-H. Meng, "Improving monocular visual SLAM in dynamic environments: An optical-flow-based approach," *Adv. Robot.*, vol. 33, no. 12, pp. 576–589, Jun. 2019, doi: 10.1080/01691864.2019.1610060.
[49] Y. Liu and J. Miura, "RDS-SLAM: Real-time dynamic SLAM using semantic segmentation methods," *IEEE Access*, vol. 9, pp. 23772–23785, 2021, doi: 10.1109/ACCESS.2021.3050617.

PENG SU received the B.E. degree from Nanjing Agricultural College University, Nanjing, China, in 2018. He is currently pursuing the master's degree with the Shanghai University of Engineering Science, Shanghai, China. His research direction focuses on the computer vision. His current research interests include VSLAM and deep learning.

SUYUN LUO was born in Jianghua, Hunan, China, in 1975. She received the B.S. degree in computer applied science from Xiangtan University, Hunan, in 1999, and the M.S. degree in traffic information engineering and control from the Wuhan University of Technology, Wuhan, Hubei, in 2003.

From 2003 to 2011, she was a Lecturer with the School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, where she has been an Associate Professor, since 2012. Her research interest includes unmanned vehicle environment perception.

XIAOCI HUANG was born in Harbin, Heilongjiang, China, in 1980. He received the Ph.D. degree in wireless sensor work, positioning, robotics and computer vision from the School of Mechanical and Power Engineering, East China University of Science and Technology.

He joined the Shanghai University of Engineering Science, in 2006, where he has developed research activities in vehicles, robotics, and intelligent control. He has been involved in teaching activities in vehicle engineering as an Associate Professor, since 2016. His current research interests include intelligent vehicle and robotics.