Dynamic Semantics SLAM Based on Improved Mask R-CNN

XINGUANG ZHANG, XIANKUN WANG, AND RUIDONG ZHANG
School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China
Corresponding author: Xinguang Zhang (zxg416@126.com)

ABSTRACT Simulation localization and mapping (SLAM) is a popular research problem in the field of driverless cars, but there are still some difficult problems to solve. The conventional SLAM algorithm does not take into account the dynamic objects in the environment, resulting in problems such as low bit pose estimation. In this study, we provide a deep learning-based SLAM scheme. In order to solve the problem of inaccurate feature point extraction in a dynamic environment, this paper adopts image pyramid to distribute feature points uniformly and extracts feature points by using the adaptive thresholding method. To address the problem of incomplete dynamic object mask segmentation, an improved Mask R-CNN network was proposed to improve the integrity of the mask edges by adding an edge detection end to the Mask R-CNN network. To address the problem of incomplete dynamic feature point rejection, this study uses a motion consistency detection algorithm to detect dynamic feature points, and uses the remaining static features for bit pose estimation. The experimental results on the TUM R-GBD dataset show that the absolute trajectory error of the SLAM algorithm in this study is reduced by 93.2% on average compared to the ORB-SLAM2 algorithm. Compared to the Dyna-SLAM algorithm, the absolute trajectory error of the algorithm in this study was reduced by 36.6% on average.

INDEX TERMS SLAM, dynamic objects, Mask R-CNN, edge detection, TUM R-GBD.

I. INTRODUCTION Because of the progress made in computer and sensor technology, more and more industries are intersecting with computers and sensors, and the driverless industry is no exception. In the driverless industry depending on the sensor it can be divided into laser SLAM, vision SLAM, and multi-sensor fusion SLAM systems. A large number of SLAM researchers have started to transform their research direction to visual SLAM because of the low cost, rich information acquisition, and ease of use of the camera.

Several excellent visual SLAM algorithms have been proposed, such as ORB-SLAM2 [1], PL-SLAM [2], and SVO [3]. However, the above SLAM algorithms all work under the assumption that the surrounding environment is a static one. The pose estimation of the camera is based on matching pixels in the image to obtain the final pose estimate. [4] When there are dynamic objects in the environment, the camera moves while the dynamic objects in the environment are also moving, which can lead to large errors in the pose estimation. In this case, algorithms are required to reduce the impact of dynamic objects on the accuracy of the camera’s pose estimation. Some scholars have built mathematical models to eliminate the effect of dynamic objects through camera motion patterns, but most SLAM algorithms have more constraints. So this method is difficult to eliminate the effect of dynamic objects. With the development of deep learning, many SLAM researchers have considered incorporating deep learning networks into SLAM systems and using deep learning networks to reject dynamic objects in the environment. Therefore, many visual SLAM systems based on deep learning have been developed, including DS-SLAM [5] and Dyna-SLAM [6]. Although the SLAM system based on a deep learning network can segment the dynamic objects in the environment well, there are still some errors in the edge detection part of the dynamic objects. Therefore, this study designs a SLAM system based on an improved Mask R-CNN [7], through which dynamic objects...
as well as their edge parts can be segmented to improve the localization accuracy of the SLAM system.

The main contents of this paper are shown below:

(1) A visual SLAM system based on an improved Mask RC-NN neural network is proposed by adding an edge detection end to the Mask R-CNN network, through which the mask can be made to cover the dynamic objects more completely;

(2) Layering feature points through an image pyramid, and using adaptive thresholding for each layer of the pyramid to eliminate some points that do not match the nature of the feature points;

(3) Rejecting dynamic feature points using dynamic consistency detection algorithm: The geometric constraint relationship of feature points between adjacent frames is determined by the pair-pole geometric constraint, and feature points that do not conform to the pair-pole geometric constraint are considered dynamic feature points that can be rejected.

II. RELATED WORK

Research on SLAM systems in dynamic environments is currently in full swing. Jin [8] et al. used the YOLO as a fast dynamic object detection network, which greatly improved the operation speed of the SLAM system. Ballester [9] et al. proposed a DOT visual SLAM front-end, where DOT first performs instance segmentation of dynamic objects and tracks these dynamic objects by minimizing the reprojection error, and by this method the segmentation accuracy is improved. Liu [10] et al. proposed a multi-sensor fusion visual SLAM system, DMS-SLAM, which uses a combination of GMS and sliding windows to reduce the impact of dynamic feature points, the 3D points of the current frame obtained by reprojection are combined with a reference model to achieve real-time update of map points. Finally, the static 3D map points are constructed by the GMS feature matching algorithm. Li [11] et al. a real-time RGB-D SLAM system based on depth edges was the subject of the proposed technology, which reduces tracking errors by statically weighting the edge points of keyframes. Wang [12] et al. proposed a method for detecting motion targets based on mathematical models and geometric constraints. The goal of this method is to eliminate the dynamic feature points and make the system more efficient. Bahraini [13] et al. proposed a new method for tracking multiple targets in a setting that is constantly changing, the ML-RANSAC algorithm, which roughly estimates the velocity and position of each moving object in a dynamic environment and achieves the simultaneous processing of dynamic objects and tracking of movable objects. Zhang [14] et al. incorporated the direction of arrival (DoA) of the sound source into an RGB-D image and addressed the effect of dynamic obstacles on a multi-robot SLAM system using this method. Bahraini [15] et al. proposed an algorithm for fusing SLAM and multi-target tracking in dynamic environments, using multi-level random sample consistency (ML-RANSAC) for multi-target tracking within the framework of extended Kalman filtering, employing machine learning to distinguish dynamic objects, and fusing LiDAR and vision sensors to detect the depth information of targets. Sheng [16] et al. proposed a semantic monocular direct vision odometry Dynamic-DSO, which generates semantic messages of dynamic objects through convolutional neural networks, obtains photometric errors through a pyramidal tracking model, and finally sliding window method is used to ignore the photometric error, to obtain an accurate camera pose. Xiao [17] et al. added an a priori semantic SSD object detector to the original convolutional neural network layer and added a new algorithm to the existing SSD object detector: a missed detection compensation algorithm for the velocity invariance of adjacent frames, which is optimized for dynamic environment SLAM systems. Sualeh [18] employed Visual-LiDAR’s Multi-Target Detection and Tracking technique to deal with the dynamic regions of a scene. Kawewong [19] et al. proposed a method using position-invariant robust features (PIRFs) that can achieve 100% high accuracy recall in dynamic environments. Qiu [20] et al. proposed a new framework, AirDOS, to jointly optimize the camera poses, dynamic object trajectories, and maps of the surrounding environment via AridOS. Dai [21] et al. grouped static and dynamic feature points in a dynamic environment, created a sparse graph from all map points by using Delaunay triangulation, then removed the edges between unrelated points based on positional correlation, and finally separated dynamic object map points from static scene map points in the remaining graph. You [22] et al. proposed a new dynamic SLAM algorithm, MIDS-SLAM, which achieves simultaneous localization and mapping construction through instance segmentation, dynamic pixel semantic deletion, and 3D map construction. Hu [23] et al. proposed a SLAM algorithm, DOE-SLAM, which combines static target scenes and target features for pose estimation. When there are too few background features, the camera pose is estimated by the features of the object and the features of the predicted object motion. Wang [24] et al. generated semantic maps by extracting semantic information of dynamic objects and overlaying semantic objects on static maps. Zhang [25] et al. propose a new RGB-D SLAM scheme in which optical flow residuals were used to highlight the dynamic semantics in point clouds. Rosinol [26] et al. introduced 3D dynamic scene graphs (DSG) to recognize semantic information in dynamic environments and implemented an automated method for constructing DSG from visual inertial data via Kinera. Chang [27] et al. introduced the YOLACT network for instance segmentation of dynamic objects, used multi-view geometry to reject dynamic feature points outside the mask, and used an adaptive thresholding optical flow method to further reject dynamic feature points. Li [28] et al. proposed a visual SLAM system based on sparse features, which was based on the results of semantic segmentation and geometric constraints combined with Bayesian probability estimation to track dynamic feature points. Campos [29] et al. proposed the ORB-SLAM3 system, which became the first SLAM system
Figure 1. System flow chart.

III. THE SYSTEM FRAMEWORK

In 2015, the first version of ORB-SLAM [30] was open-sourced by the Robotics Perception and Real-Time Research Group at the University of Zaragoza Spain. Due to the widespread interest in its excellent results, the team open-sourced a second version, ORB-SLAM2, in 2016. ORB-SLAM2 has three threads: tracing, local map building, and loopback detection. Improved robustness in tracking keyframes, trajectory construction, and map-building performance through three threads. The system flow in this paper is shown in Figure 1, compared to ORB-SLAM2 the semantic segmentation module has been added and motion target detection module have been added. When an RGB-D image is input, the input image goes into both semantic segmentation and tracking modules. In the semantic segmentation thread, the modified Mask R-CNN network segments the objects, splitting the objects in the entire scene into two categories: dynamic and static parts. The tracking thread identifies dynamic objects in the scene by using the pixel-level semantics of dynamic objects, and finally determines dynamic object features by motion consistency detection, and eliminates dynamic object features to complete the bit pose estimation by using only static features.

A. ORB FEATURE POINT EXTRACTION

An algorithm called ORB-SLAM is widely used because of its good performance. However, there are usually many dynamic objects in the scene, and dynamic objects can cause blurring of the image when they are in motion, which makes it difficult to extract the feature points accurately.

In this study, we use image pyramids for ORB feature point extraction, which was applied to each layer of the image pyramid and established the grid model used while constructing an m-layer image pyramid, with $L_i$ denoting layer $i$ of the image pyramid. The scaling scale for each layer is as follows:

$$S_i = b^i (i = 0, 1, 2, \ldots, m - 1)$$ (1)

In equation (1): $\beta$ is the scale factor and $S_i$ is the scaling scale.

The number of feature points that were taken from each layer of the m-layer picture pyramid that was created before and which is to mesh may be stated as:

$$M_i = M \times \frac{1 - \left(\frac{1}{S_i}\right)}{1 - \left(\frac{1}{S_i}\right)^m} \times \left(\frac{1}{S_i}\right)^i$$ (2)
where $M_i$ is the total amount of feature points in the layer $i$ and $M$ is the total quantity of feature points.

The extraction of feature points is carried out within the grid determined above. For the purpose of achieving a distribution of equal amounts across feature points within the grid, the amount of unique selling points extracted for each grid in a layer $i$ is set in this study as follows:

$$M'_i = \frac{M_i}{S_{\text{row}} + S_{\text{col}}}$$  \hspace{1cm} (3)

where $S_{\text{row}}$ is the number of rows per grid arrangement and $S_{\text{col}}$ is the number of columns per grid arrangement.

In this study, we adopted an adaptive thresholding approach to extract feature points in each grid, with an initial threshold of:

$$T_{\text{imi}} = \frac{M_i}{m \times m_i} \sum_{x=1}^{m_i} (L(x) - m)^2$$  \hspace{1cm} (4)

where $T_{\text{imi}}$ is the initial threshold, the image gray value is $L(x)$, $m_i$ is the number of points made up of pixels, and $\mu$ is the average image gray level.

The number of feature points in the grid extracted by adaptive thresholding is denoted as $M'_{\text{imi}}$, and if $M_{\text{imi}} < M'_i$, the threshold value is increased until the adaptive extraction of feature points in the grid is completed.

### B. THE MASK R-CNN NETWORKS

Mask R-CNN is a framework primarily used for semantic segmentation, and adding other branches may do a range of tasks like human pose estimation, semantic segmentation, instance segmentation, target categorization, and target detection. In this study, the Mask R-CNN algorithm was mostly utilized for semantic segmentation. For semantic segmentation, Mask R-CNN is the addition of a branch for semantic segmentation to the subtree of the Faster R-CNN algorithm responsible for classification and regression [31], and Figure 2 clearly shows the overarching structure of the Mask R-CNN algorithm.

For Mask R-CNN the working principle is as follows:

1. The input of the pre-processed original image;
2. Transformation of the input image into a feature map utilizing a featured network;
3. Then set a fixed number of ROIs at all pixel locations in the feature map and these ROI regions are fed into the RPN network, which then performs binary classification and coordinate regression to obtain the final processed ROI regions;
4. Processing of the refined ROI region obtained in the previous step, first by corresponding the pixels of the original picture to the feature picture and finally by corresponding the feature image to the fixed features;
Finally, these ROI regions are subjected to the following three types of operations to complete the segmentation task: multi-category classification, candidate frame regression, and import into FCN to generate the mask part.

In this study, the Mask R-CNN allows for pixel-by-pixel semantic segmentation and instance labeling. Mask R-CNN classifies potential classes such as pedestrians, bicycles, train buses, and birds. This semantic segmentation list contains the dynamic objects that are likely to be present in most environments. The SLAM algorithm group in this paper is intended for indoor use, so the SLAM algorithm in this paper semantically segments chairs, computer monitors, and people, treating chairs and computer monitors as different kinds of dynamic objects, while mainly treating pedestrians as dynamic objects. The input of this paper is an RGB-D image of size $m \times n \times 3$. The output of the network is a matrix of size $m \times n \times l$, where $l$ is the total number of recognizable objects in the picture, and then combines the $l$ classes of images into one image.

C. EDGE DETECTION SIDE

The Mask R-CNN semantic segmentation network can segment dynamic objects well within dynamic objects, but this segmentation network does not show all the edges of dynamic objects well on the generated mask after performing the segmentation. The edges of the dynamic object are still exposed to the static environment recognized by the segmentation network, when some feature points that appear on the edges of the dynamic mask are recognized in the feature point extraction session, which will make the final SLAM system’s predicted trajectory and the real trajectory differ significantly. In this study, a new detection end, the edge detection end, was added to Mask R-CNN to address this problem. We were inspired by manual instance annotation, when we annotated the instance pixels, we only annotated the edges of the instance, while in the interior of the instance we simply copied the annotation information of the edges. The edge detection side uses the edge information of the instance as a guideline for supervision, i.e. edge filtering of the groundtruth.

This article makes use of the Laplace filter, a second-order differential operator with better edge localization and better sharpening than first-order differentials. It defines a discrete form of the second-order differential operator, which is used to generate an image convolution filter template. For a two-dimensional image, the Laplace operator is defined as shown in equation (5):

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

$$\text{Gray} = 0.3 \times R + 0.59 \times G + 0.11 \times B$$  \(6\)

When the Laplace filter is applied to the mask, it may lead to blurring of the image mask edges, for which Gaussian smoothing was performed before edge detection in this study. The Gaussian smoothing is preceded by grayscale processing of the color image, where the grayscale processing function is shown in equation (6). In Figure 3, the first row is the original picture, the second row is the picture before Gaussian smoothing, and the third row is the image after Gaussian smoothing, after comparison, we can find that the edges of the mask after Gaussian smoothing are more clear and complete than the edges of the mask without Gaussian smoothing.

![Figure 3. Before Gaussian smoothing and after Gaussian smoothing.](image-url)

The final specific new branch added is shown in Figure 4, where a $3 \times 3$ edge detection module is added to the $28 \times 28 \times 80$ dimensional output branch of the mask branch, as shown in the green area of the figure, which detects the convolution of the computed and groundtruth’s edge truth values. The red area in the figure represents a two-dimensional gaussian smoothing noise reduction module that pre-processes the groundtruth module. The two-dimensional gaussian function is shown in equation (7), with $L_{\alpha}$ in the figure representing the calculation of the difference between the two, as shown in equation (8):

$$g(x, y) = \frac{1}{2\pi \sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$  \(7\)

$$L_{\alpha}(y, \hat{y}) = M_{\alpha}(y - \hat{y})^\alpha$$  \(8\)
where the \((x, y)\) is the coordinates of the points, \(\sigma\) is the standard deviation, \(s\) represents the power of the pixel difference parameter, \(y\) represents the predicted value, and \(\hat{y}\) is the actual value.

**D. MOTION CONSISTENCY DETECTION AND POSE ESTIMATION**

The semantic segmentation network described above can be used to obtain the mask of a dynamic object, but in general, the object is only determined to be a dynamic object based on the semantic label, in some special cases it may not be moving. For example, a person determined to be a dynamic object may not be moving at all, so it is necessary to determine whether the identified dynamic object is moving in the sequence or not.

Although frame difference, optical flow, and background cancellation are mainstream methods for detecting dynamic objects, these methods are not very stable in practice. In this study, dynamic feature points were detected using a motion consistency detection method. The motion consistency test is represented here by a schematic, which can be found in Figure 5.

In the above diagram \(O_1\) and \(O_2\) and \(P\) define a plane called the polar plane, \(e_1\) and \(e_2\) are called the poles, \(O_1O_2\) is called the baseline and \(L_1\) and \(L_2\) are called the poles, where \(P_1\) and \(P_2\) are the characteristic points and the normalized coordinates of the characteristic points \(P_1\) and \(P_2\) can be expressed as:

\[
P_1 = [u_1, v_2, 1] \quad P_1 = [u_2, v_2, 1] \tag{9}
\]

The polar lines can be calculated using equation (10), which also expresses the mapping relationship between the feature points \(P_1\) and \(L_i\) in space.

\[
L_i = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = FP_1 = F \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \tag{10}
\]

\(F\) in equation (10) is the basis matrix.

The mapping between feature points and polar lines allows us to establish whether or not a feature point is dynamic feature point. If the feature point \(P_2\) is not on the line of the polar cap, calculate the distance between the point of interest and the line as shown by the equation (11):

\[
S = \frac{p_2^T l_2}{\sqrt{X^2 - Y^2}} \tag{11}
\]

If the distance \(S\) is larger than the pre-set limit at this point, at this point, it is decided that the feature point in question is dynamic., otherwise it is judged to be a static feature point.
The equation shows the formula for calculating the threshold value (12):

\[ T = \beta + \mu d \]  

The above equation \( \beta \) denotes the threshold base value, \( \mu \) denotes the parallax coefficient, and \( d \) is the difference in pixel position between two adjacent frames of the feature point, i.e. the parallax value. Where threshold reference values, parallax coefficients and parallax values are obtained from specific camera parameters or in specific environments.

If at this point the feature point \( P_2 \) exists on the polar line, a block of \( 5 \times 5 \) size around \( P_2 \) and \( P_1 \) is selected for block matching at this point, and the similarity between the two blocks of pixels is calculated using normalized correlation matching at this point, and the similarity between the two blocks of pixels is calculated using normalized correlation with de-averaging, as demonstrated by equation (13):

\[
NCC = \frac{\sum_{i,j} [A(i,j) - \bar{A}(i,j)](B(i,j) - \bar{B}(i,j))}{\sqrt{\sum_{i,j} [A(i,j) - \bar{A}(i,j)]^2 \sum_{i,j} [B(i,j) - \bar{B}(i,j)]^2}}
\]  

In equation (13), \( A \) and \( B \) are the blocks of pixels near the feature point, and \( \bar{A} \) and \( \bar{B} \) are the mean values of the blocks of pixels. When the value of NCC is 1, it means the highest similarity between pixel blocks, therefore 0.7 is set as the boundary value in this paper, and when the value of NCC exceeds 0.7, it is judged as static feature points, otherwise it is judged as dynamic feature points.

After rejecting the point of dynamic feature by motion consistency detection, Static feature points estimate pose. The reprojection error is then used to obtain the positional attitude of the system, and the reprojection error function is shown in equation (14):

\[
\gamma(\xi) = (p' - \frac{1}{SK(RP + t)}), \quad K = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}
\]  

where \( K \) is the internal reference matrix, \( R \) is the rotation matrix, \( t \) is the translation matrix, and \( S \) is the scale factor. Finally, the bit pose of the system is obtained by least squares, which is shown in equation (15):

\[
\xi^* = \arg \min \frac{1}{2} \| \gamma(\xi) \|_2^2
\]  

IV. EXPERIMENTAL RESULTS

In this study, experiments were conducted on the RGB-D dataset from TUM. The main use is in which sequences in which people are in motion are used, where moving people can be treated as dynamic objects, and the dataset covers the rich textural features of the general office scene as well as a large number of dynamic sequences in the office scene. In addition, the TUM RGB-D dataset offers two criteria for evaluating the performance of the SLAM system in terms of tracking, which are trajectory errors of absolute and relative. This study mainly adopts absolute trajectory error as the judging criterion. Absolute trajectory error refers to the disparity that exists between the real camera attitude and the value that the SLAM system has estimated for it, while relative trajectory error refers to the variation in camera posture between the real and estimated poses after the same time, which can reflect the accuracy of the algorithm and the trajectory drift in the algorithm. The tests were performed primarily on an ASUS TUF GAMING F15 FX506LU model computer with the configuration shown in Table 1. A number of third-party software libraries were used in the specific SLAM experiments, including OpenCV, Eigen3, G2o, Pangolin, and others.

**TABLE 1. Computer configuration.**

| type       | model         |
|------------|---------------|
| Processor  | Intel(R) Core(TM) i5-10300H |
| Video Cards| NVIDIA GeForce GTX 1660 |
| Memory     | 16.00 GB      |
| Solid State Drives | 512 GB     |

A. ANALYSIS OF SEMANTIC SEGMENTATION RESULTS

The resulting segmentation in the SLAM system involved in this study is depicted in Figure 6 below. The initial row is a frame from the original collection of data, the next row is the mask segmented in Dyna-SLAM, and the third row is the mask of a moving object segmented by the SLAM system in this study. From the figure, it can be seen that the mask edges of dynamic objects segmented by Dyna-SLAM will have incomplete coverage, and the algorithm in this study can cover the edges of dynamic objects well with the mask. This method can well prevent a situation in which feature points may appear on dynamic objects owing to the incomplete recognition of dynamic object edges, and the SLAM system discussed in this study could benefit from a significant increase in both its accuracy and robustness.

B. ANALYSIS OF SLAM BUILD RESULTS

The absolute trajectory error (ATE) and relative pose error (RPE) are one of the important indicators to determine the performance of the algorithm. The equations for the RPE of frame \( i \) and the ATE at timestamp \( j \) are shown in equations (16) and (17). This experiment focuses on the performance of the system using a highly dynamic sequence of TUM-RGBD sequences of office workers moving around the office. Where half, rpy, and static represent the four different camera movements, and fr3, w and s are an abbreviation for sequence the root-mean-square-error (RMSE) and the standard-deviation (SD) of the absolute trajectories are provided in the text. The RMSE is a measure that compares estimated values to actual values of the trajectory, so the closer the estimated trajectory is to the real trajectory, the smaller the RMSE value will be. The SD captures the dispersion of trajectory estimates in a more quantitative manner. The above two values can be used to reflect the stability and robustness of the SLAM system more intuitively, and Table 2-4 depicts the consequences of the proposed system compared to systems such as ORB-SLAM2. The formulas
for SD and RMSE in this study are shown in equations (18) and (19).

\[
RPE = (Q_i^{-1}Q_{i+1})^{-1}P_i^{-1}P_{i+1} \tag{16}
\]

\[
ATE = Q_i^{-1}SP_j \tag{17}
\]

where the time interval is \( t \), \( Q \) is the true trajectory, and \( P \) is the predicted trajectory.

\[
\delta_{\text{RMSE}} = \frac{\delta - k}{\delta} \times 100\% \tag{18}
\]

\[
\delta_{\text{SD}} = \frac{\mu - r}{\mu} \times 100\% \tag{19}
\]

where \( \delta_{\text{RMSE}} \) represents the change in RMSE of this paper, \( \delta \) represents the RMSE of systems such as ORB-SLAM2, and \( k \) represents the RMSE of this paper. \( \delta_{\text{SD}} \) represents the change in SD of this paper, \( \mu \) represents the SD of SLAM systems such as ORB-SLAM2, and \( r \) represents the SD of this paper.

The aforementioned findings demonstrate that the stability and robustness of the algorithm described in this study are substantially improved in a highly dynamic environment. On the other hand, for fr3_s_static static sequences, the improvement is not significant owing to the excellent performance of the ORB-SLAM2 system in handling low dynamic scenes. The algorithm presented in this work represents a substantial advancement over Dyna-SLAM, which can be observed when comparing the two, and the improvement and operational performance are superior in most sequences.

The SLAM algorithm that is proposed in this article is compared with ORB-SLAM2 and Dyna-SLAM in this paper to further evaluate the SLAM method that is proposed in this article. The low dynamic sequence fr3_w_xyz and the extremely dynamic chain of events fr3_w_half were selected as the main test sequences, and the evo mapping tool was selected to make a comparison of the trajectories of the three algorithms with the original algorithm as shown in Figure 7. The ORB-SLAM2 algorithm’s estimated trajectory is depicted by the green line in the trajectory plot, the estimated trajectory line generated by the Dyna-SLAM

**TABLE 2. Absolute track comparison results (unit: m).**

| Algorithms/Serials | fr3_w_xyz | fr3_w_rpy | fr3_w_half | fr3_s_static |
|--------------------|-----------|-----------|------------|--------------|
| ORB2               | 0.4068    | 0.8823    | 0.7265     | 0.0115       |
| SLAM2              | 0.0892    | 0.4652    | 0.3885     | 0.0036       |
| Dyna-SLAM          | 0.0058    | 0.0456    | 0.0360     | 0.0076       |
| Our SLAM           | 0.0036    | 0.0312    | 0.0326     | 0.0021       |
| SLAM               | 0.0018    | 0.0172    | 0.0193     | 0.0015       |
| DS-SLAM            | 0.0247    | 0.4422    | 0.0303     | 0.0065       |
| SLAM               | 0.0161    | 0.2550    | 0.0159     | 0.0033       |

**TABLE 3. Panning partial results (unit: m).**

| Algorithms/Serials | fr3_w_xyz | fr3_w_rpy | fr3_w_half | fr3_s_static |
|--------------------|-----------|-----------|------------|--------------|
| ORB2               | 0.0295    | 0.0415    | 0.0385     | 0.0152       |
| SLAM2              | 0.0214    | 0.0365    | 0.0254     | 0.0065       |
| Dyna-SLAM          | 0.0071    | 0.0456    | 0.0240     | 0.0142       |
| Our SLAM           | 0.0019    | 0.0084    | 0.0152     | 0.0025       |
| SLAM               | 0.0012    | 0.0072    | 0.0082     | 0.0012       |
| DS-SLAM            | 0.0333    | 0.1503    | 0.0297     | 0.0078       |
| SLAM               | 0.0229    | 0.1168    | 0.0152     | 0.0038       |

**TABLE 4. Rotation section results (unit: m).**

| Algorithms/Serials | fr3_w_xyz | fr3_w_rpy | fr3_w_half | fr3_s_static |
|--------------------|-----------|-----------|------------|--------------|
| ORB2               | 0.5614    | 0.9781    | 0.8564     | 0.5412       |
| SLAM2              | 0.4852    | 0.5654    | 0.4545     | 0.2412       |
| Dyna-SLAM          | 0.2645    | 0.9845    | 0.5462     | 0.4512       |
| Our SLAM           | 0.4055    | 1.0230    | 0.7845     | 0.5641       |
| SLAM               | 0.2254    | 0.7123    | 0.4987     | 0.1874       |
| DS-SLAM            | 0.8266    | 3.0042    | 0.8142     | 0.2735       |
| SLAM               | 0.5826    | 2.3065    | 0.4101     | 0.1215       |
algorithm is represented by the blue trajectory line, the actual trajectory line is depicted by the dotted black line in this illustration, the SLAM algorithm that was used in this study is represented by the red trajectory line, and the purple trajectory line indicates DS-SLAM. As shown in Fig. 7, the predicted trajectory line of ORB-SLAM2 undergoes a very large trajectory drift compared to the real trajectory line. For DS-SLAM, Dyna-SLAM and the SLAM algorithm in this study, although trajectory drift also occurred, it can be seen from Figs. 8-9 that the APE and RMSE values of the algorithm in this study were better than those of Dyna-SLAM, and the APE and RMSE values of DS-SLAM were slightly better than those of the algorithm in this paper. It can be shown that the SLAM system in this study has good robustness even in a low dynamic environment.

Figure 10 shows a comparison of the trajectory errors of ORB-SLAM2, Dyna-SLAM, DS-SLAM, and the SLAM algorithm under the high dynamic sequence fr3_w_half, which is discussed in this paper. Although the trajectories of DS-SLAM, Dyna-SLAM, and the algorithm in this study all drifted under high dynamic sequences, the trajectory error in this study is relatively small compared to that of Dyna-SLAM and DS-SLAM algorithms. As shown in Figures 11-12, the APE, RMSE, median error, and mean error and standard deviation of the algorithm in this study were better than those of Dyna-SLAM and DS-SLAM in the sequence fr3_w_half.
Finally, the above data shows that the algorithm in this study has higher stability and higher positioning accuracy in highly dynamic environments. The primary distinction between this paper’s algorithm and the Dyna-SLAM system is that an edge detection header is added to the Mask R-CNN network, and a part for dynamic feature point detection is added, which greatly reduces the problem of loss of mask edge detection through the edge detection module, and the dynamic feature point detection module reduces the behavior of dynamic feature point false detection to some degree.

V. CONCLUSION
In this article, a Mask R-CNN deep learning network-based SLAM algorithm for dynamic scenes is proposed. This paper focuses on adding an edge detection end to the deep learning network Mask R-CNN network, solving the phenomenon of incomplete edge detection in the Mask R-CNN network by adding an edge detection end. Subsequently, the dynamic feature points are disregarded by the algorithm that detects motion consistency, and estimating the camera’s pose by static feature points in the static region. The findings of the tests performed on the TUM RGB-D dataset demonstrate that the SLAM system in this study is better able to recognize dynamic objects and recognize dynamic objects in their entirety. In this case, the SLAM system in this study can have a higher recognition accuracy as well as greater robustness in environments that are constantly changing. However, the SLAM system in this study still has drawbacks:
the SLAM system runs very slowly, and it is still difficult to achieve real-time performance even with GPU acceleration, and the camera bit pose estimation error is large when there is less texture information in the environment in which it is located. It is proposed to improve the running speed of the SLAM system by adopting a four-thread approach in future research, that is, adding a separate semantic segmentation thread. Simultaneously, a multi-sensor fusion approach is adopted to reduce the error of camera pose estimation.

REFERENCES

[1] 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.

[2] F. A. Moreno, D. Zhuiga-Noel, D. Scaramuzza, and J. Gonzalez-Jimenez, “PL-SLAM: A stereo SLAM system through the combination of points and line segments,” IEEE Trans. Robot., vol. 35, no. 3, pp. 734–746, Jun. 2019.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, “SVO: Fast semi-direct monocular visual odometry,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2014, pp. 15–22.

[4] P. Su, S. Lao, and X. Huang, “Real-time dynamic SLAM algorithm based on deep learning,” IEEE Access, vol. 10, pp. 87754–87766, 2022.

[5] 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), Oct. 2018, pp. 1168–1174.

[6] 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.

[7] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” in Proc. ICCV, Oct. 2017, pp. 2961–2969.

[8] G. Jin, X. Zhong, S. Fang, X. Deng, and J. Li, “Keyframe-based dynamic elimination SLAM system using YOLO detection,” in Proc. Int. Conf. Intell. Robot. Appl., 2019, pp. 697–705.

[9] I. Ballester, A. Fontana, J. Civera, K. H. Strobl, and R. Triebel, “DOT: Dynamic object tracking for visual SLAM,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2021, pp. 11705–11711.

[10] G. Liu, W. Zeng, B. Feng, and F. Xu, “DMS-SLAM: A general visual SLAM system for dynamic scenes with multiple sensors,” Sensors, vol. 19, no. 17, p. 3714, Aug. 2019.

[11] S. Li and D. Lee, “RGB-D SLAM in dynamic environments using static point weighting,” IEEE Robot. Autom. Lett., vol. 2, no. 4, pp. 2263–2270, Oct. 2017.

[12] 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, pp. 1143, May 2019.

[13] M. S. Bahraini, A. B. Rad, and M. Bozorg, “SLAM in dynamic environments via ML-RANSAC,” Mechatronics, vol. 49, pp. 105–118, Feb. 2018.

[14] T. Zhang, H. Zhang, X. Li, J. Chen, T. L. Lam, and S. Vijayakumar, “AcousticFusion: Fusing sound source localization to visual SLAM in dynamic environments,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Sep. 2021, pp. 6868–6875.

[15] M. S. Bahraini, A. B. Rad, and M. Bozorg, “SLAM in dynamic environments: A deep learning approach for moving object tracking using ML-RANSAC algorithm,” Sensors, vol. 19, no. 17, p. 3699, Aug. 2019.

[16] C. Sheng, S. Pan, W. Gao, Y. Tan, and T. Zhao, “Dynamic-DSO: Direct sparse odometry using objects semantic information for dynamic environments,” Appl. Sci., vol. 10, no. 4, p. 1467, Feb. 2020.

[17] L. Xiao, J. Wang, X. Qu, Z. Rong, and X. Zou, “Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environments,” Robot. Auto. Syst., vol. 117, pp. 1–16, Jul. 2019.

[18] M. Sudaeh and G.-W. Kim, “Semantic-aware dynamic SLAM based on 3D MODT,” Sensors, vol. 21, no. 19, p. 6355, Sep. 2021.

[19] A. Kawewong, N. Tongprasit, S. Tangruamsub, and O. Hasegawa, “Online and incremental appearance-based SLAM in highly dynamic environments,” Int. J. Robot. Res., vol. 30, no. 1, pp. 33–55, Jan. 2011.

[20] Y. Qu, C. Wang, W. Wang, M. Henein, and S. Scherer, “AirDSO: Dynamic SLAM benefits from articulated objects,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2022, pp. 8047–8053.