Computer Systems
Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2023.039272
Article

RO-SLAM: A Robust SLAM for Unmanned Aerial Vehicles in a Dynamic Environment

Jingtong Peng*

Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai, 200120, China
*Corresponding Author: Jingtong Peng. Email: jingtong_peng@163.com
Received: 19 January 2023; Accepted: 18 April 2023; Published: 28 July 2023

Abstract: When applied to Unmanned Aerial Vehicles (UAVs), existing Simultaneous Localization and Mapping (SLAM) algorithms are constrained by several factors, notably the interference of dynamic outdoor objects, the limited computing performance of UAVs, and the holes caused by dynamic objects removal in the map. We proposed a new SLAM system for UAVs in dynamic environments to solve these problems based on ORB-SLAM2. We have improved the Pyramid Scene Parsing Network (PSPNet) using Depthwise Separable Convolution to reduce the model parameters. We also incorporated an auxiliary loss function to supervise the hidden layer to enhance accuracy. Then we used the improved PSPNet to detect whether there is a movable object in the scene. If there is a movable object, its feature points will be removed in the tracking thread, and the removed feature points will not participate in the pose estimation of the camera. In addition, we proposed a filling method based on Generative Adversarial Networks (GANs) for the holes caused by dynamic object removal in the map, which employs a new auxiliary descriptor to assist GANs in restoring static scenes based on semantic information. The proposed system is evaluated on the TUM dataset, and the results indicate that the proposed method performs better than DynaSLAM and DS-SLAM on the TUM dataset. We experimented on the Cityscapes dataset, the improved PSPNet achieving an Intersection Over Union (IOU) of 0.812.

Keywords: UAVs; SLAM; semantic segmentation; dynamic points remove; GANs

1 Introduction

With the evolution of communication methods [1], UAVs are increasingly utilized in a broader range of applications, particularly in map building. Visual SLAM (VSLAM) is a vision-based localization and mapping technique extensively researched in recent years. Thanks to the development of deep learning, SLAM in dynamic environments has produced relatively good research results, such as DynaSLAM [2] and DS-SLAM [3]. The semantic information of the scene can help the visual SLAM system resist the interference of dynamic objects in the environment and provide additional auxiliary information for camera pose estimation. However, most existing visual SLAM systems are designed for small indoor areas. In indoor scenes, the SLAM algorithm is often used for wheeled robots

or fixed equipment, and high-performance computers can be used to process SLAM algorithms. The indoor scene environment is relatively simple. Therefore, the map-building task for indoor SLAM is relatively easy.

When the SLAM algorithm is used on UAVs, the working scene of SLAM changes from indoors to outdoors. Three problems need to solve: (1) outdoor scenes have more dynamic distractions than indoor scenes. (2) UAVs are constrained by battery life and load, so it is challenging to use high-performance computers conditionally. The computing platform carried by the UAVs is generally an embedded computing platform and computing performance is relatively weak. (3) Since many holes will be generated in the constructed map after dynamic object removal, the traditional method is to fill the holes through the multi-view geometric filling method, but this method is greatly affected by pose estimation. Currently, several scholars have proposed SLAM algorithms designed explicitly for UAVs. Aguilar et al. [4] proposed a high-precision real-time SLAM system. The system can run the SLAM on the UAVs by using an RGB-D sensor, the Microsoft Kinect, and a small but powerful computer. Bu et al. [5] proposed a method for real-time incremental stitching of large-scale aerial images using a monocular SLAM system to estimate camera position and pose while generating a 3D point cloud. However, both of these methods don't solve the above problems, and there still needs to be satisfactory solutions for SLAM for UAVs in dynamic environments.

In this paper, we proposed a new SLAM system based on ORB-SLAM2 [6] for UAVs in a dynamic environment. We named the SLAM system RO-SLAM "A Robust Outdoor SLAM." Firstly, we used Depthwise Separable Convolution (DSC) to reduce the model size of PSPNet [7] and used an auxiliary loss function to improve the accuracy. Then, we used the improved PSPNet to detect whether there is a movable object in the scene. If there is a movable object, its feature points will be removed in the tracking thread, and the removed feature points will not participate in the pose estimation of the camera. In addition, we proposed a filling method based on GANs, using an auxiliary descriptor to fill static scenes according to semantic information, which enhances the fault tolerance rate of the SLAM system. We used the TUM dataset and the Cityscapes dataset to verify our method. The results show that the proposed method performs better than others.

In summary, we highlight our contributions here:

1. We proposed a SLAM system working in a dynamic environment, using a semantic segmentation network to eliminate moving objects, the system has achieved outstanding results in trajectory estimation precision, and the results are better than DS-SLAM and Dyna-SLAM.

2. We also proposed a static background restoration method based on GANs. We used auxiliary descriptors to compensate for the shortcomings of the multi-view geometry method's shortcomings and improve the SLAM system's fault tolerance.

3. Given the current situation that the computing performance of the computing platform carried by the UAVs is generally not high, we used DSC to reduce the model size of PSPNet and an auxiliary loss function to improve the accuracy. This approach increases the potential for utilizing semantic-based SLAM algorithms on UAVs.

2 Related Works

VSLAM have developed over the years, and they can be divided into two categories in a dynamic environment: (1) Semantic methods based. (2) Geometry methods based.

2.1 Semantic Methods Based

In recent years, Deep Neural Networks (DNNs) have made significant strides in data analysis. Many researchers have utilized DNNs to extract semantic information from data. For instance, Flint et al. [8] proposed an indoor mapping system that makes use of photometric cues, pose information, and sparse point cloud data obtained from a metric SLAM system to create a semantically meaningful map of the indoor environment. Kundu et al. [9] proposed a novel framework for simultaneously performing semantic segmentation and 3D reconstruction with monocular video sequences. The authors used a DNN to estimate the per-pixel depth and semantic labels and then used a fusion network to incorporate the estimated labels into the 3D reconstruction process. This approach significantly improved the accuracy of both semantic segmentation and 3D reconstruction, especially in challenging scenarios like environments with poor lighting or cluttered scenes. Hermans et al. [10] employed a probabilistic graphical model to jointly estimate the 3D geometry and semantically labeled objects in the scene. The model is trained on a large dataset of RGB-D images and object annotations and incorporates both depth and color information to achieve robustness against cluttered scenes and occlusions. Reference [11] introduced a method for generating dense maps of object-class semantics in real-time, utilizing RGB-D videos captured by a depth sensor. The approach employs a hierarchical Bayesian framework to jointly estimate the 3D geometry and semantic labels of objects in real-time. The method uses a deep neural network for object detection and a Gaussian process model for semantic segmentation, enabling the system to handle varying object appearances and dynamics. Masaya et al. [12] proposed a new SLAM algorithm that leverages the benefits of semantic segmentation to improve feature detection and tracking. DeepLab v2 [13] was employed by the algorithm to eliminate dynamic objects through masking, enhancing the precision of camera pose estimation and stability of the system. The approach outperformed baseline algorithms in challenging environments. Bescos et al. [2] proposed a dynamic SLAM algorithm capable of handling fast-moving and deformable objects in the environment. The algorithm combines semantic segmentation with traditional visual odometry and mapping techniques, resulting in improved tracking and mapping of the environment. Inpainting techniques also fill in missing areas due to dynamic objects. Riazuelo et al. [14] presented a SLAM algorithm that incorporates semantic segmentation to overcome challenges in densely populated environments. The algorithm reduces the impact of dynamic obstacles by selectively integrating the observations from the static background while using semantic segmentation to identify and track non-static objects. Finally, Yu et al. [3] proposed a dynamic SLAM algorithm that combined SegNet [15] to handle dynamic environments. The algorithm can detect dynamic objects, track their movements, and update the map accordingly. The approach achieves SOTA performance on several datasets, demonstrating its effectiveness in dynamic environments.

2.2 Geometry Methods Based

The BaMVO [16] algorithm was proposed by Kim et al. to handle RGB-D sensors in dynamic environments. The algorithm estimated non-parametric background models from depth scenes in to reduce the residual weight of dynamic objects. Another motion removal method was presented by Sun et al. [17], who utilized particle filtering to improve motion detection and then applied a map to identify the foreground. Raluca [18] proposed a novel method to address dynamic objects, whereas most current methods still deploy outlier filtering techniques. Their approach utilized segmentation information to assign weights for dense RGB-D fusion. Emanuele et al. [19] employed a robust, geometric approach to moving objects without relying on scene semantic interpretation. Yang et al. [20] presented a meshing-based and geometric constraint visual SLAM algorithm that uses both sparse feature points and dense depth images. This algorithm divides the scene into small blocks utilizing meshing techniques, matches the blocks using geometric constraints, and excludes the influence of

moving objects through dynamic object detection. Sun et al. [21] proposed a moving-object removal approach for dynamic scene modeling with an RGB-D camera. The method analyzes the depth image to detect dynamic objects and removes them to generate a 3D model of the static scene. Liu et al. [22] proposed a general visual SLAM system, named DMS-SLAM, for real-time localization and map construction in dynamic environments with multiple sensors. By using motion segmentation, they improved the precision and robustness of the SLAM algorithm by distinguishing static and dynamic objects. Moreover, the system utilizes multi-view geometric constraints and depth consistency checks to optimize the quality of the generated map. Song et al. [23] presented a robust Bundle Adjustment (BA) that can reject features from dynamic objects by leveraging the pose prior estimated by IMU pre-integration. Then, they proposed keyframe and constraint grouping, based on multiple assumptions, to decrease the impact of temporarily stationary objects on loop closure.

The methods mentioned above have shown promising results in pose estimation. However, there is a need to develop a solution that can effectively handle real-time and dynamic environments while resisting interference. Additionally, these methods may have limitations in restoring static scenes. Table 1 displays a comparison between various SLAM systems in dynamic environments.

Table 1: Different SLAM systems in dynamic environments

	Type	Framework	Speed	Hardware	Scenes
Flint et al. [8]	SMB	-	-	CPU	M
Kundu et al. [9]	SMB	-	-	-	M
Hermans et al. [10]	SMB	-	0.75	CPU	D
Stuckler et al. [11]	SMB	-	-	GTX 675M	D
Masaya et al. [12]	SMB	ORB-SLAM	-	-	D
Bescos et al. [2]	SMB	ORB-SLAM2	2	Titan X	D/M/S
Riazuelo et al. [14]	SMB	ORB-SLAM2	-	-	D
Yu et al. [3]	SMB	ORB-SLAM2	17	P4000	D
Kim et al. [16]	GMB	DVO	23	CPU	D
Sun et al. [17]	GMB	DVO	-	CPU	D
Raluca [18]	GMB	-	-	CPU	D
Emanuele et al. [19]	GMB	-	-	CPU	D
Yang et al. [20]	GMB	ORB-SLAM2	23	CPU	D
Sun et al. [21]	GMB	DVO	0.1	CPU	D
Liu et al. [22]	GMB	ORB-SLAM2	30	CPU	D/M/S
Song et al. [23]	GMB	-	-	CPU	M/S

Note: The "D" means RGB-D camera, the "M" means monocular camera, the "S" means stereo camera, the "SMB" means "Semantic Methods Based," and the "GMB" means "Geometry Methods Based."

3 Proposed Method

3.1 Framework of RO-SLAM

The SLAM system in this paper is based on the ORB-SLAM2 framework. In this paper, ORB-SLAM2 is improved because of its shortcomings in dynamic environments and its application requirements on UAVs. The improved SLAM system is named RO-SLAM.

Once the SLAM system has been activated, it initializes the Intel RealSense camera to capture images. The tracking and semantic segmentation threads work in parallel to process these images simultaneously. The tracking thread first extracts ORB (Oriented FAST and Rotated BRIEF) feature points from the captured images. Subsequently, it awaits the output of the semantic segmentation thread, which provides information on the semantic meaning of each pixel in the images. When the semantic segmentation results are available, the tracking thread generates semantic descriptors based on the identified semantic labels, which helps to identify dynamic feature points and exclude them from the map construction process. Concurrently, an auxiliary descriptor is also produced to ensure robustness against potential point misplacements. By this process of identifying semantic feature points and removing dynamic ones, only stable static feature points are preserved. These points are then deployed for feature matching and map construction, which ultimately enables the system to accurately map the environment and navigate through it. After removing dynamic feature points, the semantic segmentation mask covers the image to simulate the missing image. Then, the pre-trained GAN will combine the auxiliary descriptor for inpainting in specific regions. The system will ultimately combine the inpainting results, semantic information, and stable static feature points to map construction. The system structure is shown in Fig. 1.

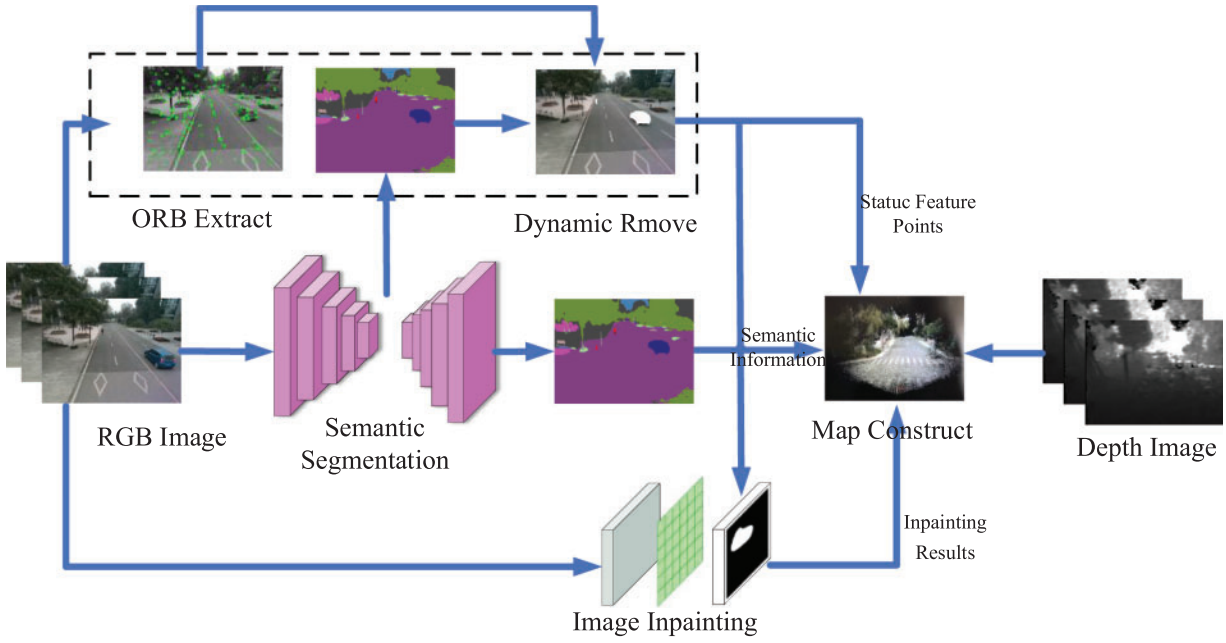


Figure 1: RO-SLAM structure

3.2 Semantic Segmentation and Dynamic Points Remove

In order for this system to be practical in real-world scenarios, there needs to be a balance between accuracy and real-time performance. To reduce the model size, we use Depthwise Separable Convolution [24] to improve the backbone of PSPNet. The structure is shown in Fig. 2.

The improved backbone of PSPNet consists of 5 parts. The parameter of each is shown in Table 2.

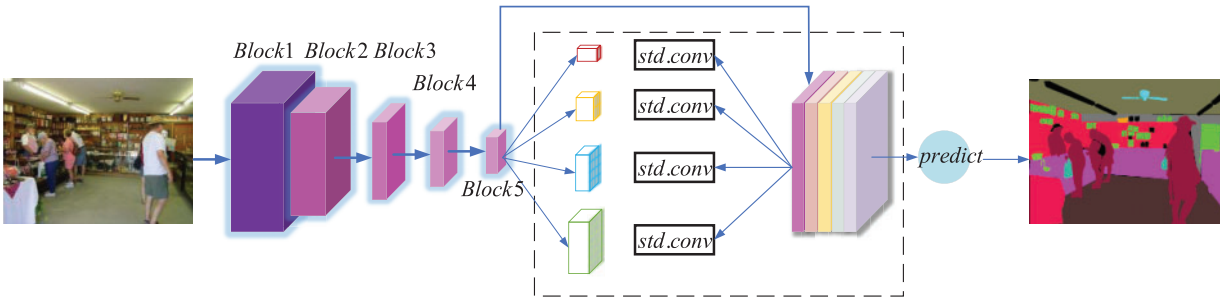


Figure 2: Improved PSPNet structure

Table 2: The parameter of the new backbone

Stage	Components	Output stride
Block 1	$[3 \times 3\ std.conv,\ 8]$	2
Block 2	$[3 \times 3\ max-pooling,\ 8]$	4
Block 3	$\begin{bmatrix} 3 \times 3\ dw.conv,\ 64 \\ 1 \times 1\ pw.conv,\ 16 \\ 3 \times 3\ dw.conv,\ 16 \\ 1 \times 1\ pw.conv,\ 64 \end{bmatrix} \times 4$	8
Block 4	$\begin{bmatrix} 3 \times 3\ dw.conv,\ 128 \\ 1 \times 1\ pw.conv,\ 32 \\ 3 \times 3\ dw.conv,\ 32 \\ 1 \times 1\ pw.conv,\ 128 \end{bmatrix} \times 8$	8
Block 5	$\begin{bmatrix} 3 \times 3\ dw.conv,\ 256 \\ 1 \times 1\ pw.conv,\ 64 \\ 3 \times 3\ dw.conv,\ 64 \\ 1 \times 1\ pw.conv,\ 256 \end{bmatrix} \times 4$	32

Block 1 consists of a standard convolution layer with a single layer and a convolution kernel size of (3×3), which can be utilized for extracting shallow features of the input image, most of which are contour and corner features. *Block* 2 is a max-pooling layer. Semantic segmentation focuses on distinguishing the boundary relationship between each instance and the scene and considering the demand for parameters of the lightweight network. Therefore, we use max-pooling to reduce the dimensionality of interior features. Although this approach may decrease accuracy, the balance between accuracy and inference speed is the key for lightweight real-time networks. *Block* 3 is a Depthwise Separable Convolution with four core sizes $(3 \times 3,\ 1 \times 1)$ and channels (64, 16, 16, 64). It plays the role of deep feature extraction together with the next *Block* 4 and *Block* 5. The entire backbone network used 33 layers of Depthwise Separable Convolutions.

Although, the improved backbone network significantly reduces the number of parameters through the alternate use of two sets of Depthwise Separable Convolution. However, this bottleneck

structure may suffer from a vanishing gradient. We integrate deep supervision information within the loss function used for training to solve this problem and use the auxiliary loss to supervise *Block* 4 and *Block* 5. The loss function is described as follows:

$$L = L_p + \sum_{i}^{K} \alpha_i L_a \tag{1}$$

Among them, L_p is the main loss function. L_a is the auxiliary loss function, K represents the number of auxiliary loss branches, and K is equal to the number of categories, and the α_i represents the proportion of different categories in the total sample. The cross-entropy at each pixel is averaged to calculate L_p and L_a according to the following equation:

$$L = \frac{1}{HW} \sum_{i}^{HW} -\log \frac{e^{s_{\hat{c}}^i}}{\sum_c^C e^{s_c^i}} \tag{2}$$

HW represents the dimensions of the input image. s_c^i represents the probability that pixel i belongs to category $c, \hat{c}$, represents the ground truth in the dataset, and C represents the overall count of categories within the dataset. And we define the objective function as:

$$F(W) = P(W) + A(W) \tag{3}$$

$P(W)$ is the main objective function, $A(W)$ is the auxiliary objective function for the hidden layer and W represents convolution kernel weight. The $P(W)$ is the optimization object of the output layer and is described as follows:

$$P(W) = ||w^{(out)}||^2 + L_p \tag{4}$$

L_p is described as follows:

$$L_p = l_p(W, w^{(out)}) \tag{5}$$

$w^{(out)}$ represents the output layer weight. The $A(W)$ can be characterized as the following:

$$A\,(W) = \sum_{i=1}^{K} \alpha_i \left[||w^{(i)}||^2 + L_a - \gamma\right] \tag{6}$$

$w^{(i)}$ represents the input layer weight, γ is an artificial bias and L_a can be characterized as the following:

$$L_a = l_a(W, w^{(i)}) \tag{7}$$

We combine Eqs. (4)–(7):

$$F\,(W) = ||w^{(out)}||^2 + l_p\left(W, w^{(out)}\right) + \sum_{i=1}^{K} \alpha_i \left[||w^{(i)}||^2 + l_a\left(W, w^{(i)}\right) - \gamma\right] \tag{8}$$

Among them, l_p and l_a are defined as follows:

$$l_p = \sum\nolimits_{y_i} [1 - < w^{(i)}, \phi(Z^{(K)}, y) - \phi(Z^{(K)}, y_i) >]_+^2 \tag{9}$$

$$l_a = \sum\nolimits_{y_k \neq y} [1 - < w^{(i)}, \phi(Z^{(i)}, y) - \phi(Z^{(i)}, y_i) >]_+^2 \tag{10}$$

$Z^{(\cdot)}$ represents hidden layer variables, y represents ground truth, and y_i represents the prediction of each branch.

In addition to learning the weight of the convolution kernel, the model can also learn sensitive features in the hidden layer.

After we get the semantic information from semantic segmentation model, we combine the ORB feature points to mark the feature points which belong to the movable objects. Then the ORB feature points that are marked as dynamic are eliminated from the original set.

$$P_{static} = P_{all} - P_{dynamic} \tag{11}$$

In Eq. (11), the P_{all} refers to the original set of ORB feature points, while P_{dynamic} represents the set of feature points that pertain to objects which have the potential to move within the surrounding environment. To ensure accurate feature point matching and pose estimation, we utilize the subset P_{static}, which only includes feature points with static positions in the environment.

Algorithm 1: Dynamic points remove

Input: RGB images queue, Dynmatic objects classes

While RGB images queue **do**
  do semantic segmentation, $(class,\ M) \leftarrow results$
  do ORB feature extract $\leftarrow P_{all}$
  if $class \in Dynamic\ objects\ classes$ **do**
    remove ORB feature points in $results \leftarrow P_{static}$
  End

3.3 Auxiliary Descriptor and Static Scene Restoration

The SIFT [25], SURF [26], and BRIEF [6] descriptors are all common descriptors without any semantic information, even though they supply photometric information of key points. To restore static scenes after removing dynamic points, we proposed an auxiliary descriptor to record dynamic objects' semantic and location information. The example of the auxiliary descriptor is described as follows:

$$D^{4\times4} = \begin{pmatrix} l_1 & u_1 & u_2 & l_2 \\ v_1 & p & c & v_2 \\ v_3 & b & n & v_4 \\ l_3 & u_3 & u_4 & l_4 \end{pmatrix} \tag{12}$$

In Eq. (12), the $\begin{pmatrix} p & c \\ b & n \end{pmatrix}$ represents the class of the feature point (p represents the person, c represents the car, b represents the bicycle and n represents the unknown object). The current area class belongs to the corresponding class whose corresponding position value is 1. The $(u,\ v)$ represents the pixel coordinate of the center position of the area, and l represents the max value between height and width.

The removed feature points will no longer participate in the construction of the point cloud map, and there will be many holes in the process of map construction, which is not friendly to downstream tasks. Therefore, the SLAM system must restore each frame's static scene. The traditional method uses the multi-view geometric filling method, which means projecting each pixel of the keyframe to the dynamic area of the current frame for filling. But this filling method does not guarantee that each removed area can be filled because the feature points of the current frame do not appear in the key

frame database, and the corresponding pixels will not be filled. If the estimation of camera pose is inaccurate, the filling method will be highly ineffective and may even lead to ghosting artifacts.

We used a GAN to restore the static background after removing the dynamic objects. This method has the advantage of camera-independent pose estimation. Feature Normalization (FN) is a widely used technique in training neural networks, which normalizes the features of input data across spatial dimensions. However, in the context of image inpainting, previous methods utilizing fuzzy neural networks have neglected the impact of damaged areas in the input image on normalization. The alterations to the mean and variance resulting from full-space FN have the potential to limit the effectiveness of training image inpainting networks. To overcome this issue, the Region Normalization (RN) approach partitions the pixels of the input image into distinct regions, utilizing the input mask to calculate the mean and variance of each region for normalization. This technique facilitates improved training of image inpainting networks. RN can be described as follows:

$$\hat{R}_{b,c}^{k} = \frac{1}{\sigma_{b,c}^{k}} \left(R_{b,c}^{k} - u_{b,c}^{k} \right) \tag{13}$$

In Eq. (13), $u_{b,c}^{k}$ is the mean value and $\sigma_{b,c}^{k}$ is the standard deviation. $X \in R^{B \times C \times S \times W}$ is assumed to be the full feature of the input, and $(B,\ C,\ S,\ W)$ represent bacth_size, the number of channels, length, and width, respectively. We set $B,\ C$ as an index and divide $X_{b,c}$ into k subregions:

$$X_{b,c} = R_{b,c}^{1} \cup R_{b,c}^{2} \cup \ldots \cup R_{b,c}^{K} \tag{14}$$

$x_{b,c,s,w}$ is a pixel in the input feature while $x_{b,c,s,w} \in R_{b,c}^{k} \in X_{b,c}$, $(b,\ c,\ s,\ w)$ is the index of $x_{b,c,s,w}$ on $(B,\ C,\ S,\ W)$. $u_{b,c}^{k}$ and $\sigma_{b,c}^{k}$ are described as follows:

$$u_{b,c}^{k} = \frac{1}{|R_{b,c}^{k}|} \sum_{x_{b,c,s,w} \in R_{b,c}^{k}} x_{b,c,s,w} \tag{15}$$

$$\sigma_{b,c}^{k} = \sqrt{\frac{1}{|R_{b,c}^{k}|} \sum_{x_{b,c,s,w} \in R_{b,c}^{k}} (x_{b,c,s,w} - u_{b,c}^{k})^2 + \varepsilon} \tag{16}$$

Finally, we merge all subregions:

$$\hat{X}_{b,c} = \hat{R}_{b,c}^{1} \cup \hat{R}_{b,c}^{2} \cup \ldots \hat{R}_{b,c}^{K} \tag{17}$$

There are certain particularities in using the GAN to repair images in SLAM. General image repair only needs to consider the rationality of the restored image, but SLAM considers the image's rationality and the reality's relevance. For example, the semantic segmentation network will divide a car into moving instances, and its feature points will be removed. If it is repaired with a general generation confrontation network, the empty area may be filled as a blank road, and for pedestrians or others, the correct one should be filled as a road. When the hole is filled as other instances, the semantic information of the scene will become confused. At this moment, the auxiliary descriptor can show its talents. When we input an image after removing dynamic points, the semantic segmentation mask covers the image to simulate the missing image. And then, the image is sent to the encoder of the corresponding attribute according to the information of the auxiliary descriptor. When performing region segmentation, the segmentation is performed according to the location information of the auxiliary descriptor:

$$x_{c,s,w} = \begin{cases} R_c^1(Dynamic - object - class) \\ R_c^2(no - Dynamic - object - class) \end{cases} \tag{18}$$

In Eq. (18), the size of R_c^1 is $(l,\ l)$. We use Eq. (13) to conduct local normalization, merge A and B, and then get the repaired image through the decoder.

4 Experimental Results and Discussion

4.1 Pose Estimation Precision

The TUM dataset [27] comprises 39 indoor scenes captured in image sequences. Every sequence contained in the dataset includes 8-bit RGB images of size 640 × 480, along with their corresponding 16-bit depth images and timestamps. Additionally, the dataset provides accurate real camera trajectories. We used six sequences of dynamic scenes in the TUM dataset to verify the proposed method. The sequences are as follows:

① fr3/walking_halfsphere sequence; ② fr3/sitting_halfsphere sequence; ③ fr3/walking_xyz sequence; ④ fr3/sitting_xyz sequence; ⑤ fr3/walking_rpy sequence; ⑥ fr3/walking_static sequence.

To evaluate the performance of our proposed methods, we performed comprehensive experiments that included a comparison of our test results against those of ORB-SLAM2 and other dynamic SLAM systems. The W_xyz sequence duration is 27 s, with a total of 2884 frames of images. The absolute trajectory errors of fr3/walking_xyz are shown in Fig. 3.

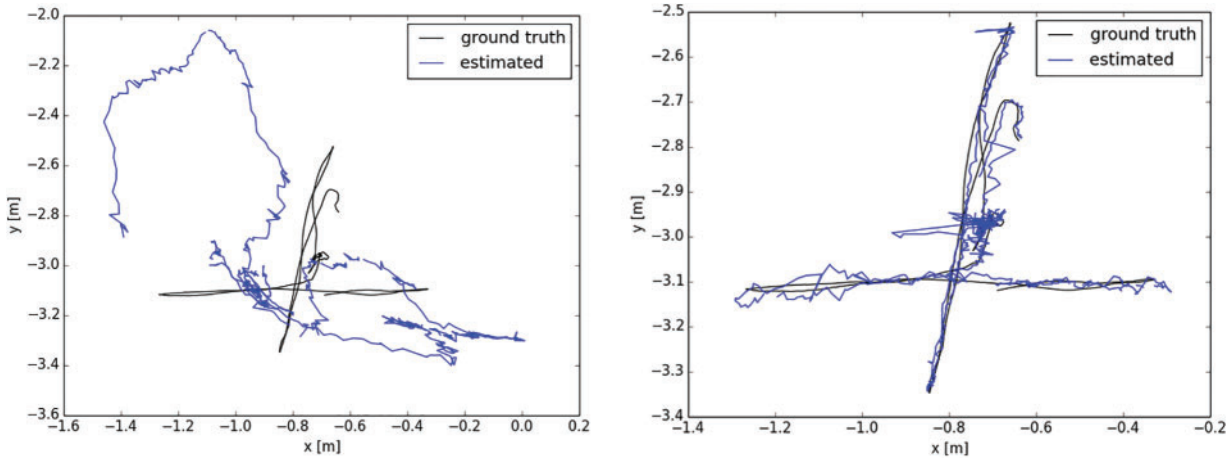


Figure 3: Absolute trajectory errors of ORB-SLAM2 (left) and RO-SLAM (right) on the fr3/walking_xyz sequence

We also calculated the relative trajectory errors shown in Fig. 4.

Through Figs. 3 and 4, RO-SLAM has a more significant improvement in camera pose estimation than ORB-SLAM2 in a dynamic environment.

We also compared the performance with DynaSLAM and DS-SLAM. To ensure objective and accurate results, we ran each experimental sequence ten times and took the median value. The detailed results can be found in Table 3.

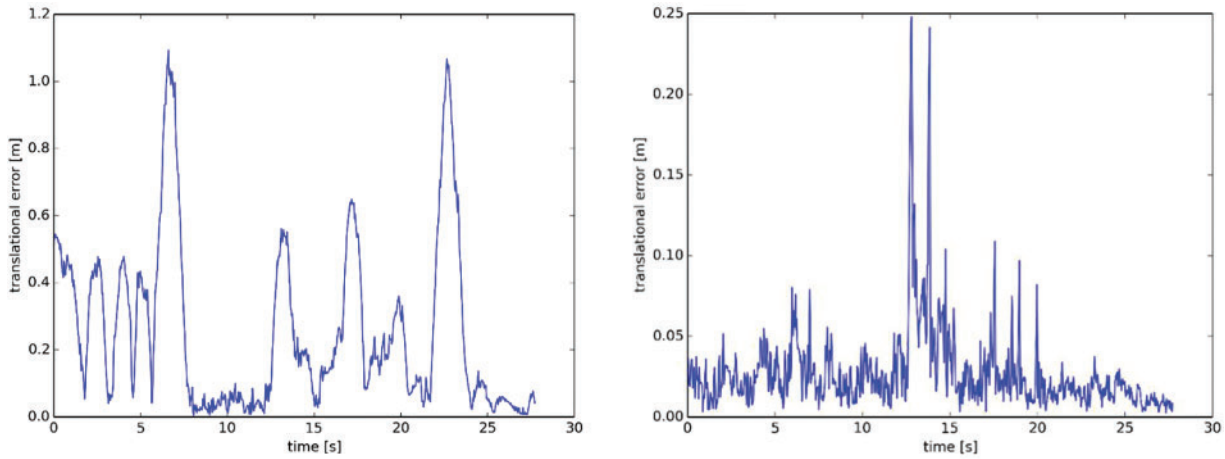


Figure 4: Relative trajectory errors of ORB-SLAM2 (left) and RO-SLAM (right) on the fr3/walking_xyz sequence

Table 3: The experiment results on the TUM dataset (RMSE of ATE) (unit: m)

Sequence	DynaSLAM	DS-SLAM	ORB-SLAM2	RO-SLAM
fr3/walking_halfsphere	0.0250	0.0222	0.3510	**0.0210**
fr3/walking_xyz	0.0150	0.0151	0.4590	**0.0146**
fr3/walking_rpy	**0.0400**	0.2835	0.6620	0.1300
fr3/walking_static	0.0090	**0.0067**	0.0900	0.0090
fr3/sitting_halfsphere	0.0170	-	0.0200	**0.0160**
fr3/sitting_xyz	0.0140	**-**	**0.0090**	**0.0090**

It can be concluded from Table 3 that RO-SLAM has a better performance improvement than DynaSLAM and DS-SLAM on the fr3/walking_halfsphere and fr3/walking_xyz sequences, and the absolute trajectory errors are 0.0210 and 0.0146, respectively. However, the precision of the fr3/walking_rpy sequence is lower than that of DynaSLAM. The reason may be that the method in this paper is not sensitive to low-angle rotation.

4.2 Semantic Segmentation Accuracy

The improved PSPNet is built using the Pytorch1.5.1 framework, and the network is randomly initialized under the default settings of PyTorch. The model training in this paper is performed on an NVIDIA Tesla V100-FHHL-16 GB. The improved PSPNet is trained using the standard Cityscapes dataset [28], the dataset is divided into 2957:500:1525 (training: verification: testing), the optimization function uses Adam, the learning rate is 0.01, and the training is 100 epochs. The visualization result of the model training convergence process is shown in Fig. 5.

And we compared the proposed method with other classic methods on the Cityscapes dataset. These classic methods are often used in SLAM. Table 4 displays the results obtained from our experiments.

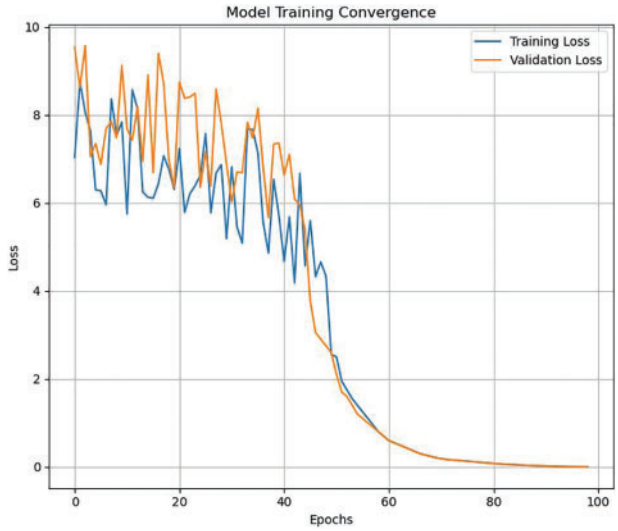


Figure 5: Model training convergence

Table 4: Semantic segmentation on the cityscapes dataset

Method	IoU
SegNet [15]	0.570
FCN [29]	0.653
DPN [30]	0.668
DeepLab [13]	0.704
PSPNet [7]	0.784
Improved PSPNet	**0.812**

Our experimental results indicate that the proposed method outperforms other classic methods. The proposed method performed better than PSPNet and improved the IoU by 2.8 points. In the actual scene, the segmentation outcomes of PSPNet and our proposed method are presented in Fig. 6.

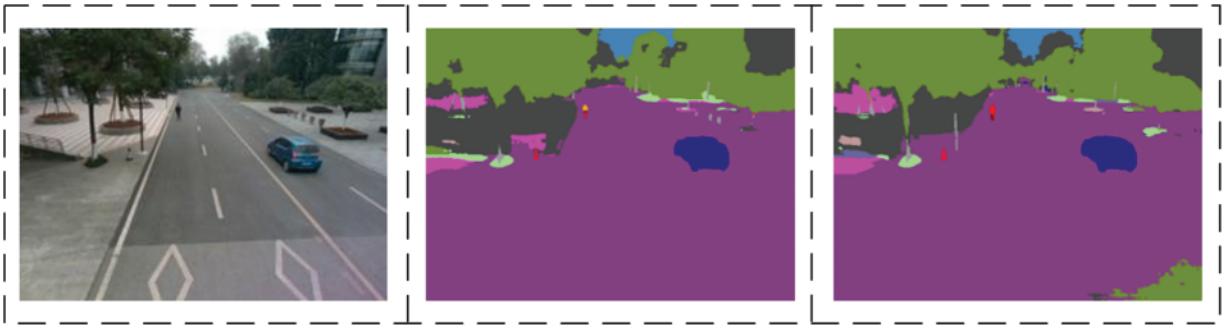


Figure 6: The results of semantic segmentation in actual scene

It can be seen that PSPNet has made false detections on the upper part of the person and has missed detections on the left tree trunk and electric lights. The proposed method is relatively regular in scene segmentation, with no false or missed detections.

4.3 Map Building in Real Environment and Efficiency Analysis

We experimented in a real environment, and the test scene is a straight road with pedestrians and cars. The DJI Matrices 600, a heavily loaded hexacopter UAV, is chosen as the aerial platform of this study. The flying height of the UVA is 3.5 meters, and the flying distance is 50 meters. The UAV's flying trajectory is a straight line of the south to north. And the UAV has a flying speed of 1 meter per second. Suppose the UAV follows a path that is too fast or too irregular. In that case, it may result in blurry or distorted images, making it difficult to accurately detect and classify objects or features in the scene. We implemented our system using C++, which was executed on a computer running the Ubuntu 16.04 operating system. The computer was equipped with Intel i7-8750H@2.20 GHz and 16 GB of memory. The graphic card is NVIDIA GTX1060-6 GB. And we used Intel Realsense D455 as the experiment depth camera to capture a video. The The videos comprise of components for RGB and depth. Each frame has a size of 640 pixels width by 480 pixels height. The camera and computer employ a USB Type-C data cable to communicate data. Fig. 7 presents the system diagram of the software and hardware configuration in this experiment.

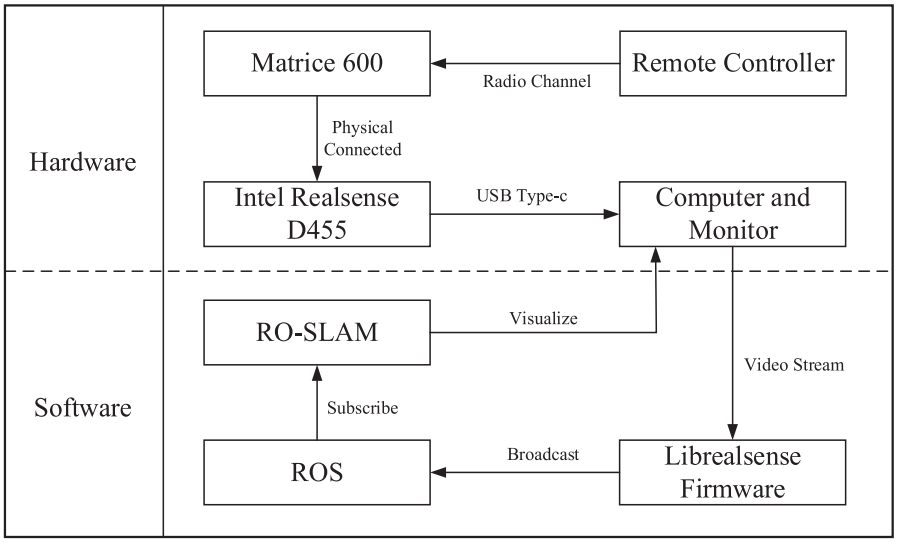
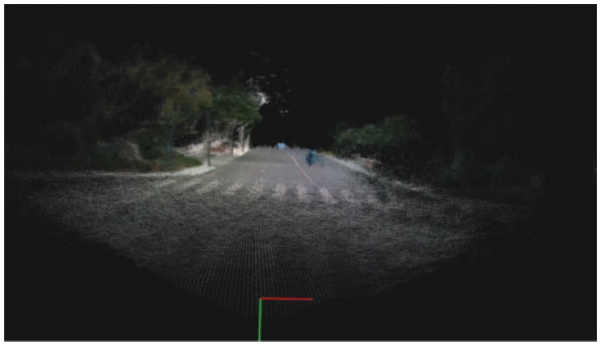


Figure 7: System diagram of the experiment

The point cloud map constructed in the real environment is shown in Fig. 8. It can be seen from Fig. 8B that dynamic objects will be retained in the map without dynamic object removal. Fig. 8C shows the holes caused by dynamic object removal. It can be seen from Figs. 8D and 8E that the point cloud map constructed by the proposed system is clear and not affected by dynamic objects. Through the static scene restore module, the map holes caused by dynamic removal are filled, almost consistent with the actual situation.

To evaluate the computation time, we compare the proposed method with DynaSLAM. We conducted a total of ten trials for the experiments and computed the average time taken per frame, which is presented in Table 5.

(A) The original RGB image and Depth image

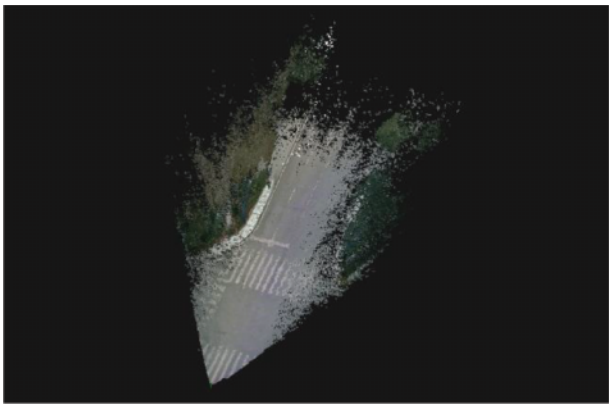


(B) Map without dynamic removal

(C) Map without static scene restore

(D) Front view of cloud point map constructed by RO-SLAM

Figure 8: (Continued)



(E) Top view of cloud point map constructed by RO-SLAM

Figure 8: Point cloud map of a real environment

Table 5: Average time expense (unit: ms)

Method	Time
Dyna-SLAM with Mask RCNN	235.98
Dyna-SLAM with PSPNet	198.42
RO-SLAM with improved PSPNet	137.96

The results show that RO-SLAM is faster than Dyna-SLAM.

5 Conclusion

In this paper, we proposed an innovative algorithm for visual SLAM called RO-SLAM. And the algorithm is designed for UAVs operating in dynamic outdoor environments. We have implemented this algorithm using the ORB-SLAM2 framework. And the proposed algorithm can handle intricate and dynamic environments. Our algorithm leveraged the improved PSPNet's semantic segmentation results to eliminate dynamic feature points from original feature points and construct an auxiliary descriptor. By eliminating the feature points of dynamic objects, the proposed algorithm improved the precision of camera pose estimation. Additionally, we proposed a new static sense restore method based on GANs, which addresses the shortcomings of traditional geometry-based methods. We performed extensive experiments on the TUM dataset to validate the effectiveness of our algorithm in dealing with highly dynamic environments and compared it with other existing algorithms. Our results showed significant improvements in performance. Additionally, we collected data using UAV in a real outdoor environment to verify RO-SLAM's ability of map construction. Through the static scene restore module, we were able to fill the map hole caused by dynamic removal, and successfully construct precision maps finally. In future work, we aim to investigate how machine learning-based schemes can be leveraged for data communication in SLAM.

Acknowledgement: The author wishes to express gratitude to the entire staff of the CIS lab at the Shanghai Advanced Research Institute for their valuable technical discussions, helpful suggestions, and provision of hardware resources.

Funding Statement: The author received no funding for this study.

Conflicts of Interest: The author declares that he has no conflicts of interest to report regarding the present study.

References

[1] X. Yihan, L. Jinhui, W. Zhou and C. Chen, "Learning-empowered resource allocation for air slicing in UAV-assisted cellular V2X communications," *IEEE Systems Journal*, vol. 17, no. 1, pp. 1008–1011, 2022.

[2] B. Bescos, J. M. Fácil, J. Civera and J. Neira, "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[3] C. Yu, Z. Liu, X. J. Liu, F. Xie, Y. Yang *et al.,* "DS-SLAM: A semantic visual SLAM towards dynamic environments," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Madrid, Comunidadde-Madrid, Spain, pp. 1168–1174, 2018.

[4] W. G. Aguilar, G. A. Rodríguez, L. Álvarez, S. Sandoval, F. Quisaguano *et al.,* "Visual SLAM with a RGB-D camera on a quadrotor UAV using on-board processing," in *Proc. of Int. Work-Conf. on Artificial Neural Networks*, Alghero, Sardinia, Italy, pp. 596–606, 2017.

[5] S. Bu, Y. Zhao, G. Wan and Z. Liu, "Map2DFusion: Real-time incremental UAV image mosaicing based on monocular SLAM," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Daejeon, Korea, pp. 4564–4571, 2016.

[6] M. A. Raúl and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[7] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid scene parsing network," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, USA, pp. 2881–2890, 2017.

[8] A. Flint, C. Mei, D. Murray and I. Reid, "Growing semantically meaningful models for visual slam," in *Proc. of IEEE Computer Vision and Pattern Recognition*, San Francisco, California, USA, pp. 467–474, 2010.

[9] A. Kundu, Y. Li, F. Dellaert, F. Li and J. M. Rehg, "Joint semantic segmentation and 3D reconstruction from monocular video," in *Proc. of European Conf. on Computer Vision*, Zurich, Switzerland, pp. 703–718, 2014.

[10] A. Hermans, G. Floros and B. Leibe, "Dense 3D semantic mapping of indoor scenes from RGB-D images," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Miami, Florida, USA, pp. 2631–2638, 2014.

[11] J. Stuckler, B. Waldvogel, H. Schulz and S. Behnke, "Dense real-time mapping of object-class semantics from RGB-D video," *Journal of Real-Time Image Processing*, vol. 10, no. 4, pp. 599–609, 2015.

[12] K. Masaya, K. Iwami, T. Ogawa, T. Yamasaki and K. Aizawa, "Mask-slam: Robust feature-based monocular slam by masking using semantic segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, Salt Lake City, UT, USA, pp. 258–266, 2018.

[13] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[14] L. Riazuelo, L. Montano and J. M. M. Montiel, "Semantic visual slam in populated environments," in *Proc. of European Conf. on Mobile Robots*, Paris, France, pp. 1–7, 2017.

[15] B. Vijay, A. Kendall and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[16] D. H. Kim and J. H. Kim, "Effective background model-based RGB-D dense visual odometry in a dynamic environment," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1565–1573, 2016.

[17] Y. Sun, M. Liu and M. Q. H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robotics and Autonomous Systems*, vol. 89, no. 4, pp. 110–122, 2017.

[18] S. Raluca, "Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Brisbane, Queensland, Australia, pp. 67–79, 2018.

[19] P. Emanuele, J. Behley, P. Lottes, P. Giguere and C. Stachniss, "Refusion: 3D reconstruction in dynamic environments for rgb-d cameras exploiting residuals," in *Proc. of Int. Conf. on Intelligent Robots and Systems*, Macao, China, pp. 7855–7862, 2019.

[20] S. Yang, G. Fan, L. Bai, R. Li and D. Li, "MGC-VSLAM: A meshing-based and geometric constraint VSLAM for dynamic indoor environments," *IEEE Access*, vol. 8, no. 3, pp. 81007–81021, 2020.

[21] Y. Sun, M. Liu and M. Q. H. Meng, "December. Invisibility: A moving-object removal approach for dynamic scene modelling using RGB-D camera," in *Proc. of the IEEE Int. Conf. on Robotics and Biomimetics*, Macao, China, pp. 50–55, 2017.

[22] 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, pp. 3714, 2019.

[23] S. Song, H. Lim, A. J. Lee and H. Myung, "DynaVINS: A visual-inertial SLAM for dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11523–11530, 2022.

[24] C. François, "Xception: Deep learning with depthwise separable convolutions," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp. 1251–1258, 2017.

[25] C. Pauline and S. Henikoff, "SIFT: Predicting amino acid changes that affect protein function," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3812–3814, 2013.

[26] B. Herbert, T. Tuytelaars and L. V. Gool, "Surf: Speeded up robust features," in *Proc. of European Conf. on Computer Vision*, Graz, Austria, pp. 404–417, 2006.

[27] S. Jürgen, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas *et al.,* "Towards a benchmark for RGB-D SLAM evaluation," in *Proc. of RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf.*, New York City, NY, USA, pp. 231–236, 2011.

[28] C. Marius, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler *et al.,* "The cityscapes dataset," in *Proc. of CVPR Workshop on the Future of Datasets in Vision*, Boston, MA, USA, pp. 23–41, 2015.

[29] L. Jonathan, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, pp. 3431–3440, 2015.

[30] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan *et al.,* "Dual path networks," *Advances in Neural Information Processing Systems*, vol. 30, no. 4, pp. 184–199, 2017.