RGB-D Inertial Odometry for a Resource-restricted Robot in Dynamic Environments

Jianheng Liu, Xuanfu Li, Yueqian Liu, and Haoyao Chen

***Abstract*— Current simultaneous localization and mapping (SLAM) algorithms perform well in static environments but easily fail in dynamic environments. Recent works introduce deep learning-based semantic information to SLAM systems to reduce the influence of dynamic objects. However, it is still challenging to apply a robust localization in dynamic environments for resource-restricted robots. This paper proposes a real-time RGB-D inertial odometry system for resource-restricted robots in dynamic environments named Dynamic-VINS. Three main threads run in parallel: object detection, feature tracking, and state optimization. The proposed Dynamic-VINS combines object detection and depth information for dynamic feature recognition and achieves performance comparable to semantic segmentation. Dynamic-VINS adopts grid-based feature detection and proposes a fast and efficient method to extract high-quality FAST feature points. IMU is applied to predict motion for feature tracking and moving consistency check. The proposed method is evaluated on both public datasets and real-world applications and shows competitive localization accuracy and robustness in dynamic environments. Yet, to the best of our knowledge, it is the best-performance real-time RGB-D inertial odometry for resource-restricted platforms in dynamic environments for now. The proposed system is open source at: https://github.com/HITSZ-NRSL/Dynamic-VINS.git**

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) is a foundational capability for many emerging applications, such as autonomous mobile robots and augmented reality. Cameras as portable sensors are commonly equipped on mobile robots and devices. Therefore, visual SLAM (vSLAM) has received tremendous attention over the past decades. Lots of works [1]–[4] are proposed to improve visual SLAM systems' performance. Most of the existing vSLAM systems depend on a static world assumption. Stable features in the environment are used to form a solid constraint for Bundle Adjustment [5]. However, in real-world scenarios like shopping malls and subways, dynamic objects such as moving people, vehicles, and unknown objects, have an adverse impact on pose optimization. Although some approaches like RANSAC [6] can suppress the influence of dynamic features to a certain extent, it will become overwhelmed when a vast number of dynamic objects appear in the scene.

This work was supported in part by the National Natural Science Foundation of China (Grant No.U21A20119 and No.U1713206) and in part by the Shenzhen Science and Innovation Committee (Grant No.JCYJ20200109113412326; No.JCYJ20210324120400003; No.JCYJ20180507183837726; No.JCYJ20180507183456108).

J.H. Liu, Y.Q. Liu and H.Y. Chen* are with the School of Mechanical Engineering and Automation, Harbin Institute of Technology Shenzhen, P.R. China (e-mail: hychen5@hit.edu.cn).

X.F. Li is with the Department of HiSilicon Research, Huawei Technology Co., Ltd, P.R. China (e-mail: lixuanfu@huawei.com).

Therefore, it is necessary for the system to reduce dynamic objects' influence on the estimation results consciously. The pure geometric methods [7]–[9] are widely used to handle dynamic objects, but it is unable to cope with latent or slightly moving objects. With the development of deep learning, many researchers have tried combining multi-view geometric methods with semantic information [10]–[13] to implement a robust SLAM system in dynamic environments. To avoid the accidental deletion of stable features through object detection [14], recent dynamic SLAM systems [15], [16] exploit the advantages of pixel-wise semantic segmentation for a better recognition of dynamic features. Due to the expensive computing resource consumption of semantic segmentation, it is difficult for a semantic-segmentation-based SLAM system to run in real-time. Therefore, some researchers have tried to perform semantic segmentation only on keyframes and track moving objects via moving probability propagation [17], [18] or direct method [19] on each frame. In the cases of missed detections or object tracking failures, the pose optimization is imprecise. Moreover, since semantic segmentation is performed after keyframe selection, real-time precise pose estimation is inaccessible, and unstable dynamic features in the original frame may also cause redundant keyframe creation and unnecessary computational burdens.

The above systems still require too many computing resources to perform robust real-time localization in dynamic environments for Size, Weight, and Power (SWaP) restricted mobile robots or devices. Some researchers [20]–[22] try to run visual odometry in real-time on embedded computing devices, yet the keyframe-based visual odometry is not performed [23], which makes their accuracy unsatisfactory. At the same time, increasingly embedded computing platforms are equipped with NPU/GPU computing units, such as HUAWEI Atlas200, NVIDIA Jetson, etc. It enables lightweight deep learning networks to run on the embedded computing platform in real-time. Some studies [14], [24] implemented a keyframe-based dynamic SLAM system running on embedded computing platforms. However, these works are still difficult to balance efficiency and accuracy for mobile robot applications.

To address all these issues, this paper proposes a real-time RGB-D inertial odometry for resource-restricted robots in dynamic environments named Dynamic-VINS. It enables edge computing devices to provide instant robust state feedback for mobile platforms with little computation burden. An efficient dynamic feature recognition module that does not require a high-precision depth camera can be used in

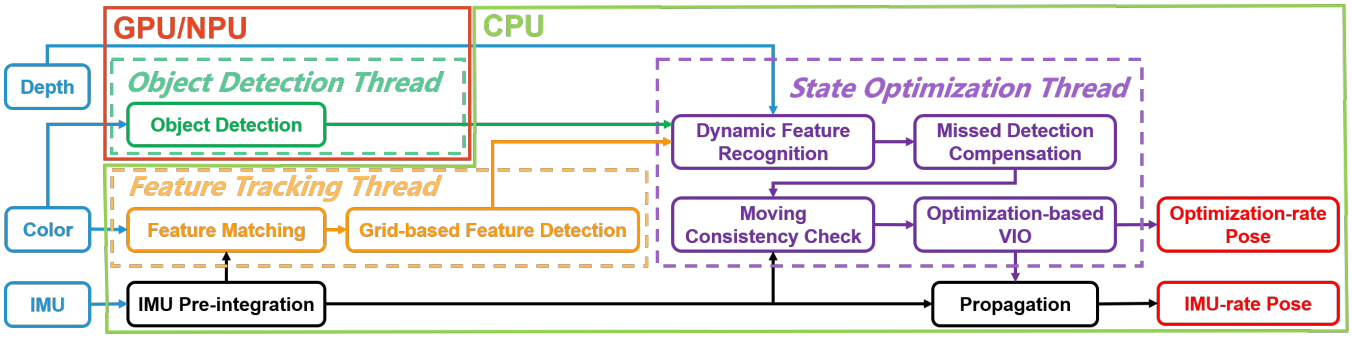


Fig. 1. The framework of Dynamic-VINS. The contributing modules are highlighted and surrounded by dash lines with different colors. Three main threads run in parallel in Dynamic-VINS. Features are tracked and detected in the feature tracking thread. The object detection thread detects dynamic objects in each frame in real-time. The state optimization thread summarizes the features information, object detection results, and depth image to recognize the dynamic features. Finally, stable features and IMU preintegration results are used for pose estimation.

mobile devices equipped with depth-measure modules. The main contributions of this paper are as follows:

1) An efficient optimization-based RGB-D inertial odometry is proposed to provide real-time state estimation results for resource-restricted robots in dynamic and complex environments.
2) Lightweight feature detection and tracking are proposed to cut the computing burden. In addition, dynamic feature recognition modules combining object detection and depth information are proposed to provide robust dynamic feature recognition in complex and outdoor environments.
3) Validation experiments are performed to show the proposed system's competitive accuracy, robustness, and efficiency on resource-restricted platforms in dynamic environments.

II. SYSTEM OVERVIEW

The proposed SLAM system in this paper is extended based on VINS-Mono [2] and VINS-RGBD [25]; our framework is shown in Fig. 1, and the contributing modules are highlighted with different colors. For efficiency, three main threads (surrounded by dash lines) run parallel in Dynamic-VINS: object detection, feature tracking, and state optimization. Color images are passed to both the object detection thread and the feature tracking thread. IMU measurements between two consecutive frames are preintegrated [26] for feature tracking, moving consistency check, and state optimization.

In the feature tracking thread, features are tracked with the help of IMU preintegration and detected by grid-based feature detection. The object detection thread detects dynamic objects in each frame in real-time. Then, the state optimization thread will summarize the features information, object detection results, and depth image to recognize the dynamic features. A missed detection compensation module is conducted in case of missed detection. The moving consistency check procedure combines the IMU preintegration and historical pose estimation results to identify potential dynamic features. Finally, stable features and IMU preintegration results are used for the pose estimation. And the propagation of the IMU is responsible for an IMU-rate pose estimation result. Loop closure is also supported in this system, but this paper pays more attention to the localization independent of loop closure.

III. METHODOLOGY

This study proposes lightweight, high-quality feature tracking and detection methods to accelerate the system. Semantic and geometry information from the input RGB-D images and IMU preintegration are applied for dynamic feature recognition and moving consistency check. The missed detection compensation module plays a subsidiary role to object detection in case of missed detection. Dynamic features on unknown objects are further identified by moving consistency check. The proposed methods are divided into five parts for a detailed description.

A. Feature Matching

For each incoming image, the feature points are tracked using the KLT sparse optical flow method [27]. In this paper, the IMU measurements between frames are used to predict the motion of features. Better initial position estimation of features is provided to improve the efficiency of feature tracking by reducing optical flow pyramid layers. It can effectively discard unstable features such as noise and dynamic features with inconsistent motion. The basic idea is illustrated in Fig. 2.

In the previous frame, stable features are colored red, and newly detected features are colored blue. When the current frame arrives, the IMU measurements between the current and previous frames are used to predict the feature position (green) in the current frame. Optical flow uses the predicted feature position as the initial position to look for a match feature in the current frame. The successfully tracked features are turned red, while those that failed to be tracked are marked as unstable features (purple). In order to avoid the repetition and aggregation of feature detection, an orange circular mask centered on the stable feature is set; the region where the unstable features are located is considered an

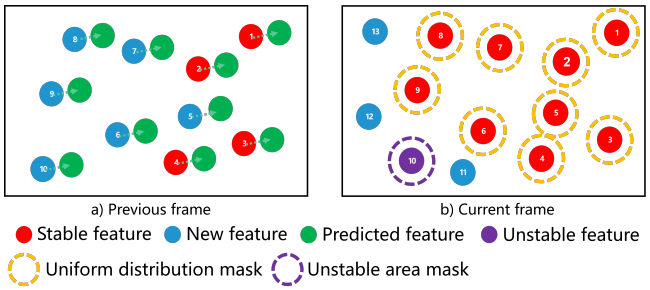


Fig. 2. Illustration of feature tracking and detection. Stable features and new features are colored red and blue, respectively. The green circles denote the prediction for optical flow. The successfully tracked features turn red; otherwise, the features turn purple. The orange and purple dash-line circles as masks are set for a uniform feature distribution and reliable feature detection. New feature points are detected from unmasked areas in the current frame.

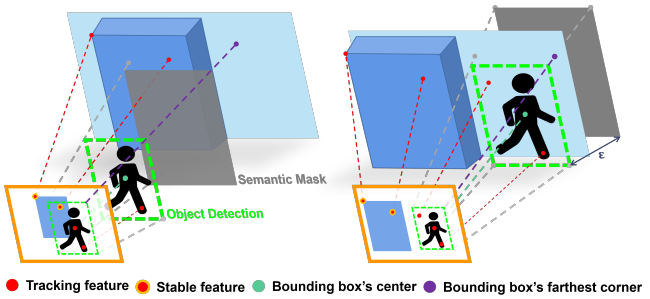


Fig. 3. Illustration of semantic mask setting for dynamic feature recognition when all pixel's depth is available ($d > 0$). The left scene represents when an objected bounding box's farthest corner's depth is bigger than the center to a threshold ε and a semantic mask with weighted depth is set between them to separate features on dynamic objects from the background. Otherwise, the semantic mask is set behind the bounding box's center with the distance of ε, shown on the right.

unstable feature detection region and masked with a purple circular to avoid unstable feature detection. According to the mask, new features are detected from unmasked areas in the current frame and colored blue.

The above means can obtain uniformly distributed features to capture comprehensive constraints and avoid repeatedly extracting unstable features on the area with blurs or weak textures. Long-term feature tracking can reduce the time consumption with the help of grid-based feature detection in the following.

B. Grid-based Feature Detection

The system maintains a minimum number of features for stability. Therefore, feature points need to be extracted from the frame constantly. This study adopts grid-based feature detection. Image is divided into grids, and the boundary of each grid is padded to prevent the features at the edge of the grid from being ignored; the padding enables the current grid to obtain adjacent pixel information for feature detection. Unlike traversing the whole image to detect features, only the grid with insufficient matched features will conduct feature detection. The grid cell that fails to detect features due to weak texture or is covered by the mask will be skipped in the next detection frame to avoid repeated useless detection. The thread pool technique is used to exploit the parallel performance of grid-based feature detection. Thus, the time consumption of feature detection is significantly reduced without loss.

The FAST feature detector [28] can efficiently extract feature points but easily treats noise as features and extracts similar clustered features. Therefore, the ideas of mask in Sec. III-A and Non-Maximum-Suppression are combined to select high-quality and uniformly distributed FAST features.

C. Dynamic Feature Recognition

Most feature points can be stably tracked through the above improvement. However, long-term tracking features on dynamic objects always come with abnormal motion and introduce wrong constraints to the system. For the sake of efficiency and computational cost, a real-time single-stage object detection method, YOLOv3 [11], is used to detect many kinds of dynamic scene elements like people and vehicles. If a detected bounding box covers a large region of the image, blindly deleting feature points in the bounding box might result in no available features to provide constraints. Therefore, semantic-segmentation-like masks are helpful to maintain the system's running by tracking features not occluded by dynamic objects.

This paper combines object detection and depth information for highly efficient dynamic feature recognition to achieve performance comparable to semantic segmentation. As the farther the depth camera measures, the worse the accuracy is. This problem makes some methods, such as Seed Filling, DBSCAN, and K-Means, which make full use of the depth information, exhibit poor performance with a low accuracy depth camera, as shown in Fig. 5(a). Therefore, a set of points in the detected bounding box and depth information are integrated to obtain comparable performance to the semantic segmentation, as illustrated in Fig. 3.

A pixel's depth d is available, if $d > 0$, otherwise, $d = 0$. Considering that the bounding box corners of most dynamic objects correspond to the background points, and the dynamic objects commonly have a relatively large depth gap with the background. The K-th dynamic object's largest background depth ${}^{K}d_{max}$ is obtained as follow

$$ {}^{K}d_{max} = \max\left({}^{K}d_{tl} + {}^{K}d_{tr}{}^{K} + {}^{K}d_{bl} + {}^{K}d_{br}\right), \tag{1} $$

where ${}^{K}d_{tl}$, ${}^{K}d_{tr}$, ${}^{K}d_{bl}$, ${}^{K}d_{br}$ are the depth values of the Kth object detection bounding box's corners, respectively. Next, the Kth bounding box's depth threshold ${}^{K}\bar{d}$ is defined as

$$ {}^{K}\bar{d} = \begin{cases} \frac{1}{2}\left({}^{K}d_{\max} + {}^{K}d_{c}\right), & \text{if } {}^{K}d_{\max} - {}^{K}d_{c} > \varepsilon,\ {}^{K}d_{c} > 0, \\ {}^{K}d_{c} + \varepsilon, & \text{if } {}^{K}d_{\max} - {}^{K}d_{c} < \varepsilon,\ {}^{K}d_{c} > 0, \\ {}^{K}d_{\max}, & \text{if } {}^{K}d_{\max} > 0,\ {}^{K}d_{c} = 0, \\ +\infty, & \text{otherwise}, \end{cases} \tag{2} $$

where ${}^{K}d_{c}$ is the depth value of the bounding box's center; $\varepsilon > 0$ is a predefined distance according to the most common dynamic objects' size in scenes. The depth threshold ${}^{K}\bar{d}$ is defined in the middle of the center's depth ${}^{K}d_{c}$ and the

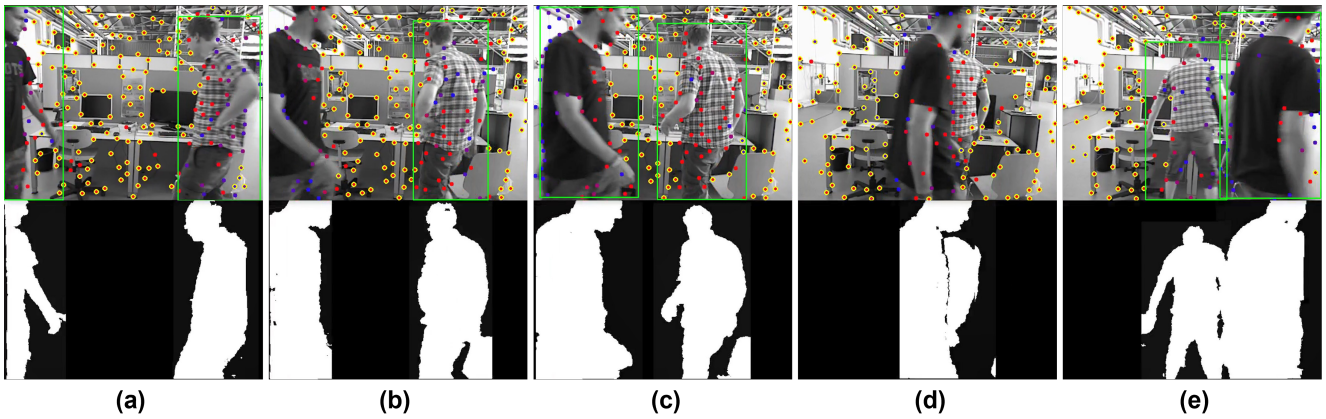
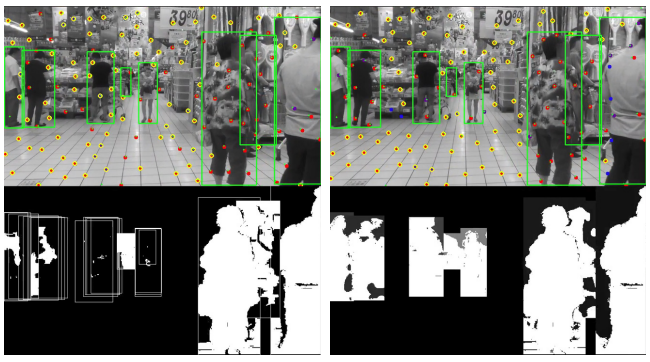


Fig. 4. Results of missed detection compensation. The dynamic feature recognition results are shown in the first row. The green box shows the dynamic object's position from the object detection results. The second row shows the generated semantic mask. With the help of missed detection compensation, even if object detection failed in (b) and (d), a semantic mask including all dynamic objects could be built.

deepest background depth ${}^{K}d_{\max}$. When the dynamic object has a close connection with the background or is behind an object ${}^{K}d_{\max} - {}^{K}d_c < \varepsilon$, the depth threshold is defined at ε distance from the dynamic object. If the depth is unavailable, a conservative strategy is adopted to choose an infinite depth as the threshold.

On the semantic mask, the area covered by the K-th dynamic object bounding box is set to the weighted depth ${}^{K}\bar{d}$; the area without dynamic objects is set to 0. Each incoming feature's depth d is compared with the corresponding pixel's depth threshold $\bar{d}$ on the semantic mask. If $d < \bar{d}$, the feature is considered as a dynamic one. Otherwise, the feature is considered as a stable one. Therefore, the region where the depth value is smaller than the weighted depth $\bar{d}$ constitutes the generalized semantic mask, as shown in Fig. 4 and Fig. 5(b).



(a) Seed Filling (b) Proposed Method

Fig. 5. Results of dynamic feature recognition. The stable features are circled by yellow. The dynamic feature recognition results generated by Seed Filling and the proposed method are shown in (a) and (b), respectively. The weighted depth $\bar{d}$ is colored gray; the brighter means a bigger value. The feature point on the white area will be marked as a dynamic feature.

Considering that dynamic objects may exist in the field of view for a long time, the dynamic features are tracked but not used for pose estimation, different from directly deleting dynamic features. According to its recorded information, each incoming feature point from the feature tracking thread will be judged whether it is a historical dynamic feature or not. The above methods can avoid blindly deleting feature points while ensuring efficiency. It can save time from detecting features on dynamic objects, has the robustness to the missed detection of object detection, and recycle false-positive dynamic features, as illustrated in Sec. III-E.

D. Missed Detection Compensation

Since object detection might sometimes fail, the proposed Dynamic-VINS utilizes the previous detection results to predict the following detection result to compensate for missed detections. It is assumed that the dynamic objects in adjacent frames have a consistent motion. Once a dynamic object is detected, its pixel velocity and bounding box will be updated. Assumed that j is the current detected frame and $j-1$ is the previous detected frame, the pixel velocity ${}^{K}\mathbf{v}^{c_j}$ (pixel/frame) of the Kth dynamic object between frames is defined as

$$ {}^{K}\mathbf{v}^{c_j} = {}^{K}\mathbf{u}_c^{c_j} - {}^{K}\mathbf{u}_c^{c_{j-1}}, \tag{3} $$

where ${}^{K}\mathbf{u}_c^{c_j}$, ${}^{K}\mathbf{u}_c^{c_{j-1}}$ represent the pixel location of the Kth object detection bounding box's center in jth frame and $j-1$th frame, respectively. A weighted predicted velocity ${}^{K}\hat{\mathbf{v}}$ is defined as

$$ {}^{K}\hat{\mathbf{v}}^{c_{j+1}} = \frac{1}{2}({}^{K}\mathbf{v}^{c_j} + {}^{K}\hat{\mathbf{v}}^{c_j}), \tag{4} $$

With the update going on, the velocities of older frames will have a lower weight in ${}^{K}\hat{\mathbf{v}}$. If the object fail to be detected in the next frame, the bounding box ${}^{K}\mathbf{Box}$ containing the corners' pixel locations ${}^{K}\mathbf{u}_{tl}, {}^{K}\mathbf{u}_{tr}, {}^{K}\mathbf{u}_{bl}$ and ${}^{K}\mathbf{u}_{br}$, will be updated based on the predicted velocity ${}^{K}\hat{\mathbf{v}}$ as follow

$$ {}^{K}\hat{\mathbf{Box}}^{c_{j+1}} = {}^{K}\mathbf{Box}^{c_j} + {}^{K}\hat{\mathbf{v}}^{c_{j+1}}, \tag{5} $$

When the missed detection time is over a threshold, this dynamic object's compensation will be abandoned. The result is shown in Fig. 4. It improves the recall rate of object detection and is helpful for a more consistent dynamic feature recognition.

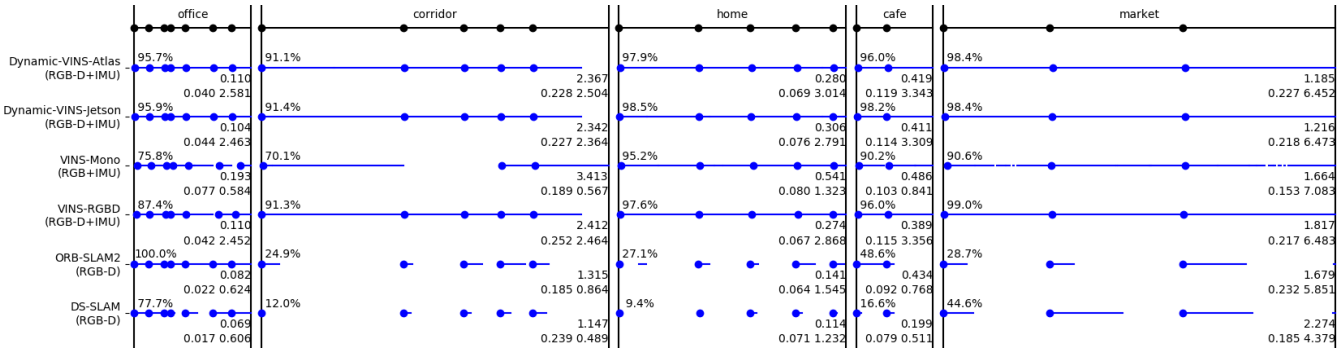


Fig. 6. Per-sequence testing results with the OpenLORIS-Scene datasets. Each black dot on the top line represents the start of one data sequence. For each algorithm, blue dots indicate successful initialization moments, and blue lines indicate successful tracking span. The percentage value on the top left of each scene is the average correct rate; the higher the correct rate of an algorithm, the more robust it is. The float value on the first line below is average ATE RMSE and the values on the second line below are T.RPE and R.RPE from left to right, and smaller means more accurate.

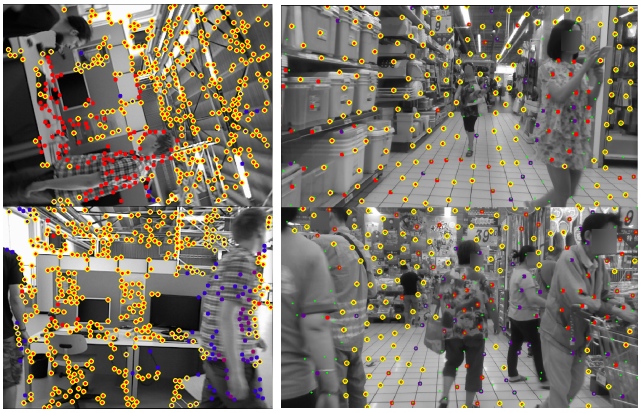


Fig. 7. Results of Moving Consistency Check. Features without yellow circular are the outliers marked by the Moving Consistency Check module.

E. Moving Consistency Check

Since object detection can only recognize artificially defined dynamic objects and has a missed detection problem, the state optimization will still be affected by unknown moving objects like books moved by people. Dynamic-VINS combines the pose predicted by IMU and the optimized pose in the sliding windows to recognize dynamic features.

Consider the kth feature is first observed in the ith image and is observed by other m images in sliding windows. The average reprojection residual r_k of the feature observation in the sliding windows is defined as

$$r_k = \frac{1}{m}\sum_{j\neq i}\left\|\mathbf{u}_k^{c_i} - \pi\left(\mathbf{T}_b^c\mathbf{T}_w^{b_i}\mathbf{T}_{b_j}^w\mathbf{T}_c^b\mathbf{P}_k^{c_j}\right)\right\|, \tag{6}$$

where $\mathbf{u}_k^{c_i}$ is the observation of kth feature in the ith frame; $\mathbf{P}_k^{c_j}$ is the 3D location of kth feature in the jth frame; $\mathbf{T}_c^b$ and $\mathbf{T}_{b_j}^w$ are the transforms from camera frame to body frame and from jth body frame to world frame, respecvtively; π represents the camera projection model. When the r_k is over a preset threshold, the kth feature is considered as a dynamic feature.

As shown in Fig. 7, the moving consistency check (MCC) module can find out unstable features. However, some stable features are misidentified (top left image), and features on standing people are not recognized (bottom right image). A low threshold holds a high recall rate of unstable features. Further, a misidentified unstable feature with more observations will be recycled if its reprojection error is lower than the threshold.

IV. EXPERIMENTAL RESULTS

Quantitative experiments[1] are performed to evaluate the proposed system's accuracy, robustness, and efficiency. Public SLAM evaluation datasets, OpenLORIS-Scene [29] and TUM RGB-D [30], provide sensor data and ground truth to evaluate SLAM system in complex dynamic environments. Since our system is built on VINS-Mono [2] and VINS-RGBD [25], they are used as the baselines to demonstrate our improvement. VINS-Mono [2] provides robust and accurate visual-inertial odometry by fusing IMU preintegration and feature observations. VINS-RGBD [25] integrates RGB-D camera based on VINS-Mono for better performance. Furthermore, DS-SLAM [15] and Ji *et al*. [24], state-of-the-art semantic algorithms based on ORB-SLAM2 [4], are also included for comparison.

The accuracy is evaluated by Root-Mean-Square-Error (RMSE) of Absolute Trajectory Error (ATE), Translational Relative Pose Error (T.RPE), and Rotational Relative Pose Error (R.RPE). Correct Rate (CR) [29] measuring the correct rate over the whole period of data is used to evaluate the robustness. The RMSE of an algorithm is calculated only for its successful tracking outputs. Therefore, the longer an algorithm tracks successfully, the more error is likely to accumulate. It implies that evaluating algorithms purely by ATE could be misleading. On the other hand, considering only CR could also be misleading.

In order to demonstrate the efficiency of the proposed system, all experiments of Dynamic-VINS are performed on the embedded edge computing devices, HUAWEI Atlas200 DK and NVIDIA Jetson AGX Xavier. And the compared algorithms' results are included from their original papers.

[1]The experimental video is available at https://youtu.be/y0U1IVtFBwY

TABLE I

RESULTS OF RMSE OF ATE $[m]$, T.RPE $[m/s]$, AND R.RPE $[^\circ/s]$ ON TUM RGB-D *fr3_walking* DATASETS.

Sequence	ORB-SLAM2 [4]			DS-SLAM [15]			Ji *et al*. [24]			Dynamic-VINS		
	ATE	T.RPE	R.RPE	ATE	T.RPE	R.RPE	ATE	T.RPE	R.RPE	ATE	T.RPE	R.RPE
fr3_walking_xyz	0.7521	0.4124	7.7432	0.0247	0.0333	0.8266	**0.0194**	**0.0234**	**0.6368**	0.0486	0.0578	1.6932
fr3_walking_static	0.3900	0.2162	3.8958	0.0081	0.0102	**0.2690**	0.0111	0.0117	0.2872	**0.0077**	**0.0095**	0.4581
fr3_walking_rpy	0.8705	0.4249	8.0802	0.4442	0.1503	3.0042	**0.0371**	**0.0471**	**1.0587**	0.0629	0.0595	5.0839
fr3_walking_half	0.4863	0.3550	7.3744	0.0303	**0.0297**	**0.8142**	**0.0290**	0.0423	0.9650	0.0608	0.0665	5.2116

TABLE II

ABLATION EXPERIMENT RESULTS OF RMSE OF ATE $[m]$, T.RPE $[m/s]$, AND R.RPE $[^\circ/s]$ ON TUM RGB-D *fr3_walking* DATASETS.

Sequence	W/O CIRCULAR MASK			W/O OBJECT DETECTION			W/O SEG-LIKE MASK			W/O MCC		
	ATE	T.RPE	R.RPE	ATE	T.RPE	R.RPE	ATE	T.RPE	R.RPE	ATE	T.RPE	R.RPE
fr3_walking_xyz	0.9795	0.6156	6.2692	0.0592	0.0575	1.7181	0.0523	0.0608	1.7474	0.0676	0.0604	1.8020
fr3_walking_static	0.4111	0.4052	9.8985	0.3458	0.3136	9.2520	0.0305	0.0194	0.5463	0.0454	0.0229	0.5676
fr3_walking_rpy	0.4111	0.4052	9.8985	0.2138	0.1191	5.4847	0.1174	0.0729	5.5470	0.1236	0.0996	5.4196
fr3_walking_half	1.1218	0.6779	11.521	0.0988	0.0651	5.1839	0.0754	0.0672	5.1952	0.1748	0.1169	5.8525

Atlas200 DK has an 8-core A55 Arm CPU (1.6GHz), 8 GB of RAM, and a 2-core HUAWEI DaVinci NPU. Jetson AGX Xavier has an 8-core ARMv8.2 64-bit CPU (2.25GHz), 16 GB of RAM, and a 512-core Nvidia Volta GPU. And the results tested on both devices are named Dynamic-VINS-Atlas and Dynamic-VINS-Jetson, respectively. Yet, to the best of our knowledge, the proposed method is the best-performance real-time RGB-D inertial odometry for dynamic environments on resource-restricted embedded platforms.

A. OpenLORIS-Scene Dataset

OpenLORIS-Scene [3] is a real-world indoor dataset with a large variety of challenging scenarios like dynamic scenes, featureless frames, and dim illumination. The results on the OpenLORIS-Scene dataset are shown in Fig. 6, including the results of VINS-Mono, ORB-SLAM2, and DS-SLAM from [3] as baselines.

The OpenLORIS dataset includes five scenes and 22 sequences in total. The proposed Dynamic-VINS shows the best robustness among the tested algorithms. In *office* scenes that are primarily static environments, all the algorithms can track successfully and achieve a decent accuracy. It is challenging for the pure visual SLAM systems to track stable features in *home* and *corridor* scenes that contain a large area of textureless walls and dim lighting. Thanks to the IMU sensor, the VINS systems show robustness superiority when the camera is unreliable. The scenarios of *home* and *cafe* contain a number of sitting people with a bit of motion, and *market* exists lots of moving pedestrians and objects with unpredictable motion. And the *market* scenes cover the largest area and contain highly dynamic objects, as shown in Fig. 5. Although DS-SLAM is able to filter out some dynamic features, its performance is still unsatisfactory. VINS-RGBD has a similar performance with Dynamic-VINS in relative static scenes, while VINS-RGBD's accuracy drops in highly dynamic *market* scenes. The proposed Dynamic-VINS can effectively deal with complex dynamic environments and improve robustness and accuracy.

B. TUM RGB-D Dataset

The TUM RGB-D dataset [30] offers several sequences containing dynamic objects in indoor environments. The highly dynamic *fr3_walking* sequences are chosen for evaluation where two people walk around a desk and change chairs' positions while the camera moves in different motions. As the VINS system does not support VO mode and the TUM RGB-D dataset does not provide IMU measurements, a VO mode is implemented by simply disabling modules relevant to IMU in Dynamic-VINS for experiments. The results are shown in Table I. The compared methods' results are included from their original published papers. The algorithms based on ORB-SLAM2 and semantic segmentation perform better. Although Dynamic-VINS is not designed for pure visual odometry, it still shows competitive performance and has a significant improvement over ORB-SLAM2.

To validate the effectiveness of each module in Dynamic-VINS, ablation experiments are conducted as shown in Table II. The system without applying circular masks (W/O CIRCULAR MASK) from the Sec. III-A and Sec. III-B fails to extract evenly distributed stable features, which seriously degrades the accuracy performance. Without the object detection (W/O OBJECT DETECTION), dynamic features introduce wrong constraints to impair the system's accuracy. Dynamic-VINS-W/O-SEG-LIKE-MASK shows the results that mask all features in the bounding boxes. The background features help the system maintain as many stable features as possible to provide more visual constraints. The moving consistency check plays an important role when object detection fails, as shown in the column W/O-MCC.

C. Runtime Analysis

This part compares VINS-Mono, VINS-RGBD, and Dynamic-VINS for runtime analysis. These methods are expected to track and detect 130 feature points, and the frames in Dynamic-VINS are divided into 7x8 grids. The object detection runs on the NPU/GPU parallel to the CPU. The average computation times of each module and thread

TABLE III

AVERAGE COMPUTATION TIME [*ms*] OF EACH MODULE AND THREAD ON OPENLORIS *market* SCENES.

Platforms	Mehods	Feature Tracking	Feature Detection	Tracking Thread*	Dynamic Feature Recognition Modules†	State Optimization	Optimization Thread*	Object Detection*
HUAWEI Atlas200 DK	VINS-Mono [2]	18.6226	58.2712	57.6301	-	76.5047	85.0247	-
	VINS-RGBD [25]	20.6066	58.9413	81.4598	-	75.2211	83.3476	-
	Dynamic-VINS	15.5350	1.7645	19.8980	1.3424	74.9509	82.4916	17.5850
NVIDIA Jetson AGX Xavier	VINS-Mono [2]	4.4990	14.3691	10.9123	-	49.5326	52.4842	-
	VINS-RGBD [25]	4.1099	15.4521	11.9251	-	49.0472	52.3388	-
	Dynamic-VINS	3.3649	0.9396	5.5416	0.4707	43.0424	47.5377	21.9211

* Tracking Thread, Optimization Thread and Object Detection correspond to the three different threads shown in Fig. 1, respectively.

† Dynamic Feature Recognition Modules sum up the Dynamic Feature Recognition, Missed Detection Compensation, and Moving Consistency Check modules.

are calculated on OpenLORIS *market* scenes; the results run on both embedded platforms are shown in Table III. It should be noted that the average computation time is only to be updated when the module is used. Specifically, in VINS architecture, the feature detection is executed at a consistent frequency with the state optimization thread, which means the frequency of feature detection is lower than that of Feature Tracking Thread.

On edge computing devices with AI accelerator modules, the single-stage object detection method is computed by an NPU or GPU without costing the CPU resources and can output inference results in real-time. With the same parameters, Dynamic-VINS shows significant improvement in feature detection efficiency in both embedded platforms and is the one able to achieve instant feature tracking and detection in HUAWEI Atlas200 DK. The dynamic feature recognition modules (Dynamic Feature Recognition, Missed Detection Compensation, Moving Consistency Check) to recognize dynamic features only take a tiny part of the consuming time. For real-time application, the system is able to output a faster frame-to-frame pose and a higher-frequency imu-propagated pose rather than waiting for the complete optimization result.

D. Real-World Experiments

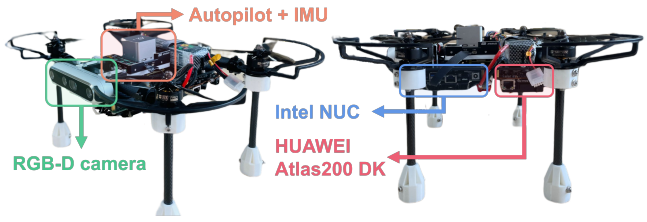
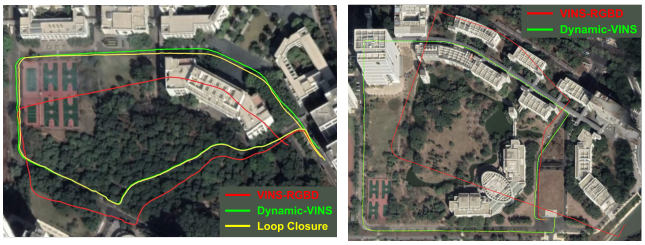


Fig. 8. A compact aerial robot equipped with an RGB-D camera, an autopilot with IMUs, an onboard computer, and an embedded edge computing device. The whole size is about 255x165mm.

A compact aerial robot is shown in Fig. 8. An RGB-D camera (Intel Realsense D455) provides 30Hz color and aligned depth images. An autopilot (CUAV X7pro) with an onboard IMU (ADIS16470, 200Hz) is used to provide IMU measurements. The aerial robot is equipped with an onboard computer (Intel NUC, i7-5557U CPU) and an embedded edge computing device (HUAWEI Atlas200 DK). These two computation resource providers play different roles in the aerial robot. The onboard computer charges for peripheral management and other core functions requiring more CPU resources, such as planning and mapping. The edge computing device as auxiliary equipment offers instant state feedback and object detection results to the onboard computer.



(a) HITSZ campus (b) THUSZ campus

Fig. 9. The estimated trajectories in the outdoor environment aligned with the Google map. The green line is the estimated trajectory from Dynamic-VINS, the red line is from VINS-RGBD, and the yellow line represents the loop closure that happened at the end of the dataset.

Large-scale outdoor datasets with moving people and vehicles on the HITSZ and THUSZ campus are recorded by the handheld aerial robot above for safety. The total path lengths are approximately 800*m* and 1220*m*, respectively. The dataset has a similar scene at the beginning and the end for loop closure, while loop closure fails in the THUSZ campus dataset. VINS-RGBD and Dynamic-VINS run the dataset on NVIDIA Jetson AGX Xavier. The estimated trajectories and loop closure trajectory aligned with the Google map are shown in Fig. 9. In outdoor environments, the depth camera is limited in range and affected by the sunlight. The dynamic feature recognition modules can still segment dynamic objects but with a larger mask region, as shown in Fig. 10. Compared with loop closure results, Dynamic-VINS could provide a robust and stable pose estimation with little drift.

V. CONCLUSIONS

This paper presents a real-time RGB-D inertial odometry for resource-restricted robots in dynamic environments. Cost-efficient feature tracking and detection methods are proposed to cut down the computing burden. A lightweight

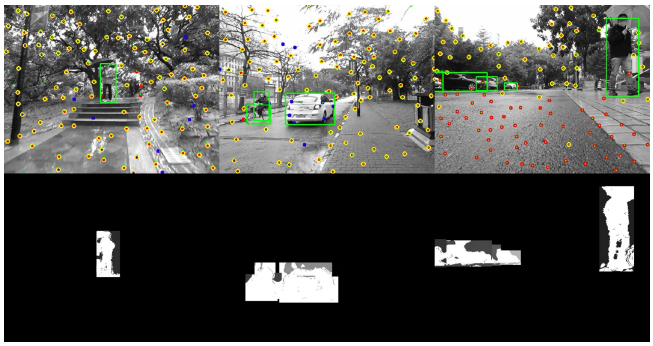


Fig. 10. Results of dynamic feature recognition in outdoor environments. The dynamic feature recognition modules are still able to segment dynamic objects but with a larger mask region.

object-detection-based method is introduced to deal with dynamic features in real-time. Validation experiments show the proposed system's competitive accuracy, robustness, and efficiency in dynamic environments. Furthermore, Dynamic-VINS is able to run on resource-restricted platforms to output an instant pose estimation. In the future, the proposed approaches are expected to be validated on the existing popular SLAM frameworks. The missed detection compensation module is expected to develop into a moving object tracking module, and semantic information will be further introduced for high-level guidance on mobile robots or mobile devices in complex dynamic environments.

REFERENCES

[1] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[2] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[3] P. Geneva, K. Eckenhoff, *et al.*, "OpenVINS: A Research Platform for Visual-Inertial Estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4666–4672.

[4] R. Mur-Artal and J. D. Tardos, "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.

[5] B. Triggs, P. F. McLauchlan, *et al.*, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[7] 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, pp. 110–122, 2017.

[8] E. Palazzolo, J. Behley, *et al.*, "ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7855–7862.

[9] W. Dai, Y. Zhang, *et al.*, "RGB-D SLAM in Dynamic Environments Using Point Correlations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 373–389, 2022.

[10] 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 2016*, ser. Lecture Notes in Computer Science, B. Leibe, J. Matas, *et al.*, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

[11] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767 [cs]*, 2018.

[12] V. Badrinarayanan, 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.

[13] K. He, G. Gkioxari, *et al.*, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[14] L. Xiao, J. Wang, *et al.*, "Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robotics and Autonomous Systems*, vol. 117, pp. 1–16, 2019.

[15] C. Yu, Z. Liu, *et al.*, "DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, 2018, pp. 1168–1174.

[16] B. Bescos, J. M. Facil, *et al.*, "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[17] F. Zhong, S. Wang, *et al.*, "Detect-SLAM: Making Object Detection and SLAM Mutually Beneficial," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1001–1010.

[18] Y. Liu and J. Miura, "RDS-SLAM: Real-Time Dynamic SLAM Using Semantic Segmentation Methods," *IEEE Access*, vol. 9, pp. 23 772–23 785, 2021.

[19] I. Ballester, A. Fontán, *et al.*, "DOT: Dynamic Object Tracking for Visual SLAM," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 705–11 711.

[20] K. Schauwecker, N. R. Ke, S. A. Scherer, and A. Zell, "Markerless Visual Control of a Quad-Rotor Micro Aerial Vehicle by Means of On-Board Stereo Processing," in *Autonomous Mobile Systems 2012*, ser. Informatik Aktuell, P. Levi, O. Zweigle, *et al.*, Eds. Berlin, Heidelberg: Springer, 2012, pp. 11–20.

[21] Z. Zakaryaie Nejad and A. Hosseininaveh Ahmadabadian, "ARM-VO: An efficient monocular visual odometry for ground vehicles on ARM CPUs," *Machine Vision and Applications*, vol. 30, no. 6, pp. 1061–1070, 2019.

[22] S. Bahnam, S. Pfeiffer, and G. C. de Croon, "Stereo Visual Inertial Odometry for Robots with Limited Computational Resources," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, 2021, pp. 9154–9159.

[23] G. Younes, D. Asmar, *et al.*, "Keyframe-based monocular SLAM: Design, survey, and future directions," *Robotics and Autonomous Systems*, vol. 98, pp. 67–88, 2017.

[24] T. Ji, C. Wang, and L. Xie, "Towards Real-time Semantic RGB-D SLAM in Dynamic Environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 175–11 181.

[25] Z. Shan, R. Li, and S. Schwertfeger, "Rgbd-inertial trajectory estimation and mapping for ground robots," *Sensors*, vol. 19, no. 10, p. 2251, 2019.

[26] C. Forster, L. Carlone, *et al.*, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation." Georgia Institute of Technology, 2015.

[27] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," in *Proc. DARPA Image Understanding Workshop, 1981*. Vancouver, British Columbia, 1981.

[28] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[29] X. Shi, D. Li, *et al.*, "Are We Ready for Service Robots? The OpenLORIS-Scene Datasets for Lifelong SLAM," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3139–3145.

[30] J. Sturm, N. Engelhard, *et al.*, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura-Algarve, Portugal: IEEE, 2012, pp. 573–580.