2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Semantic SLAM Fusing Moving Constraint for Dynamic Objects under Indoor Environments

Zhenyuan Yang, W. K. R. Sachinthana, S. M. Bhagya P. Samarakoon, and Mohan Rajesh Elara

Abstract— **Simultaneous Localization and Mapping (SLAM) technology is a rapidly developing field in robotics. Most existing SLAM algorithms lack robustness in dynamic environments because moving objects can influence mapping and localization accuracy, making it challenging for robots to identify moving objects and understand their surroundings. Though some works proposed semantic SLAM methods, they rely on point-based feature extraction and matching algorithms with semantic information and simply exclude dynamic objects. In this work, we proposed real-time RGB-D SLAM to combine point, line, and plane features with object detection to increase the robustness in dynamic environments. The proposed method combines object detection, feature points, and lines to identify moving objects and uses feature planes and semantic information to identify constrained moving objects. Thus, the localization accuracy can be improved under dynamic environments by excluding or using dynamic objects. Experiments were conducted on a public TUM dataset and in a real-world environment. The result shows that the proposed SLAM algorithm can increase the dynamic object detection speed and the robustness of SLAM performance compared to state-of-the-art in dynamic environments.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a significant topic in robotics, especially in autonomous navigation, robot inspection, and path planning. SLAM can be classified into two categories: laser-based SLAM and visual-based SLAM. Visual-based SLAM has a strong scene recognition ability with the advantage of low cost and easy installation compared to laser-based SLAM [1]. Visual SLAM has developed fast in recent years. For instance, traditional SLAM approaches like ORB-SLAM2 [2] using feature-based tracking, LSD-SLAM [3] using direct tracking, and VINS [4] using optical flow tracking exhibit good performance across typical scenes.

However, as the requirements of real life and industry needs increase, there are more demands for SLAM in different environments. For texture-less indoor environments, the pure point-based SLAM system has difficulty keeping its robustness because of fewer feature points to track. Lines and planes-based SLAM systems are working in texture-less environments such as Pop-up SLAM [5], Structure-PLP-SLAM [6], Planar-SLAM [7] could be found in the literature. These SLAM systems combine multiple features to keep their robustness and increase localization accuracy under indoor texture-less environments. However, the environment being static has been one of the main assumptions. When it comes to real-world scenarios, especially in indoor environments, active objects such as moving people, chairs, bottles, or some small movable objects are regular, which must be considered.

Dynamic objects can largely influence the accuracy of robot localization. To address this issue, SLAM in dynamic environments based on semantic segmentation or object detection such as DS-SLAM [8], Dyna-SLAM [9] has been introduced. However, most methods are based on deep learning techniques, so the computation time must be considered for real-time applications. Moreover, these methods detect dynamic objects and exclude all their feature points. However, when considering large moving obstacles, excluding all features on them will greatly decrease the localization accuracy, even resulting in lost tracking. Therefore, making use of dynamic objects rather than excluding them is crucial.

This paper proposes a new RGB-D SLAM system to increase robustness under dynamic indoor environments using structural and moving constraints. Further, the system uses dynamic objects constrained by indoor structures to further improve the localization accuracy. An example of dynamic object detection and map building is shown in Fig. 1. The main contributions are as follows:

- A proposed SLAM system fusing point, line, plane features with object detection to exclude dynamic objects.
- The proposed system can exclude dynamic objects faster than the state-of-the-art method.
- Proposed system has a new way of nonlinear optimization method to make use of dynamic objects.

The structure of this paper is as follows: Section II mentions the related work. Section III introduces the overview methodology of the proposed SLAM system. Section IV shows the experiment results and Section V is the conclusion and future work.

II. Related Work

A. Semantic SLAM

Semantic SLAM combines object identification with traditional SLAM techniques. While SLAM focuses solely on the geometric aspects of the environment, semantic analysis involves recognizing objects within the surroundings. By integrating these techniques, it becomes possible to generate maps that include geometric data and semantic information about the objects present in the environment. The concept

This research is supported by the National Robotics Programme under its National Robotics Programme (NRP) BAU, Ermine III: Deployable Reconfigurable Robots, Award No. M22NBK0054, A*STAR under its "RIE2025 IAF-PP Advanced ROS2-native Platform Technologies for Cross-sectorial Robotics Adoption (M21K1a0104)" programme, and also supported by SUTD Growth Plan (SGP) Grant, Grant Ref. No. PIE-SGP-DZ-2023-01.

The authors are with the Engineering Product Development Pillar, Singapore University of Technology and Design, 8 Somapah Rd, Singapore 487372.

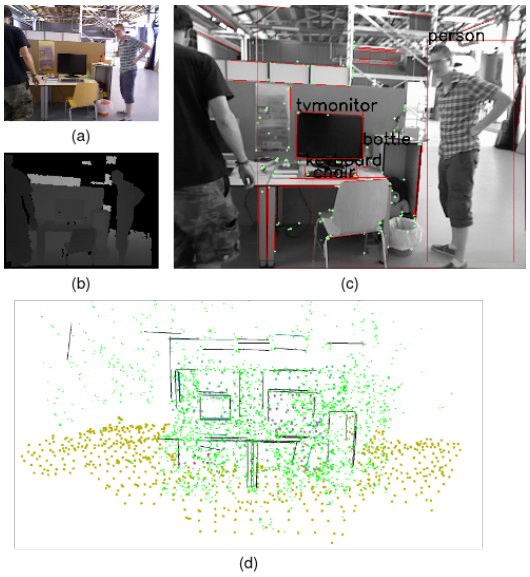


Fig. 1. An example of mapping and dynamic points exclusion results in the TUM RGB-D dataset (fr3_walking_xyz). (a) and (b) show the original RGB and depth images. (c) is the result after object detection and dynamic points exclusion. (d) is the map with points, lines and planes.

of semantic fusion has been introduced by McCormac [10]. The system consisted of three main components, SLAM ElasticFusion, a CNN, and a Bayesian update scheme. Most semantic SLAM systems use semantic information to increase their robustness under dynamic environments. SO-SLAM [11] is a semantic object SLAM method using object detection that applies spatial constraints on the object level. Yanmin Wu et al. [12] introduced an extensive object SLAM framework, emphasizing object-centric perception and object-oriented tasks for robots.

B. SLAM in Dynamic Environments

Bescos et al. proposed the Dyna-SLAM [9], built over the ORB-SLAM2 with added capabilities of handling dynamic objects. Identification of dynamic obstacles has become possible with the help of multi-view geometry and deep learning approaches. DS-SLAM [8] is a pixel-level SLAM system combined with SegNet [13]. Compared to Dyna-SLAM, DS-SLAM considers the feature points on stationary dynamic objects in the process of camera pose estimation. Therefore, DS-SLAM works much better in highly dynamic scenarios and performs faster compared to Dyna-SLAM. FD-SLAM [14] utilizes a multi-thread process and a segmentation model to manage dynamic objects with increased overall processing speed. However, having a frame-by-frame segmentation approach still hinders the speed of the FD-SLAM. RDS-SLAM [15] relies solely on a lightweight SSD detection model to identify dynamic feature points without using any geometric approach. CFP-SLAM [16] is another semantic SLAM method that calculates the static probability of objects and takes them as weights to estimate camera poses.

Semantic SLAM approaches have addressed dynamic scenarios by removing the effect of dynamic objects up to a certain level. However, there is still a gap that exists for further improvement.

C. SLAM Using Multiple Features

Feature extraction in SLAM refers to identifying distinctive and relevant visual landmarks or features from sensor data, typically obtained from cameras. These features are used to estimate the robot's position and orientation in its environment while simultaneously building a map of that environment. Combination feature extraction and mapping, such as point-line, point-plane, and line-plane, have been utilized in recent research. Xingxing et al. [17] presented a work on a Visual SLAM that utilizes heterogeneous line and point features. It has used unique orthonormal representation as the minimal parameterization to construct line and point features. Albert et al. [18] proposed a SLAM framework that processes points and lines targeting low-texture environments. Raza Yunus et al. [19] proposed a SLAM framework partly based on Manhattan World Assumption combining lines and planes features. Mehdi et al. [20] have explored the effects of accommodating plane features in a sparse point-based SLAM system and validated performance improvement. The work [21] presented a structure-aware SLAM method targeting indoor environments. It utilizes both planes and lines as essential elements for mapping and localization. Fangwen Shu et al. proposed Structure-PLP-SLAM [6], which combines point, line, and plane features for monocular, stereo, and RGB-D cameras. By combining planes and lines alongside point-based features, the method aims to better understand and map indoor environments compared to traditional point-based SLAM approaches.

III. System Introduction

A. Overview Framework

ORB-SLAM2 [2] is an outstanding SLAM system with high performance and wide usability. Therefore, the proposed work is developed based on ORB-SLAM2 to detect multiple features and dynamic objects. The overview framework of the proposed SLAM system is shown in Fig. 2. The following subsections introduce feature extraction and matching, the object detection network, the proposed dynamic object detection and exclusion algorithm, the moving constraint check, and the whole pose estimation process.

B. Feature Extraction and Matching

1) Point: For points, Oriented FAST and Rotated BRIEF (ORB) [22] feature is used here. ORB feature can be used for point feature extraction and matching process and it is widely used for point-based SLAM systems for its efficiency and accuracy.

2) Line: For line extraction, Line Segment Detector (LSD) [23] is used here. It is a linear-time line segment detector that gives subpixel accurate results. For line matching, feature line descriptors need to be calculated. Line Band Descriptor(LBD) [24] is chosen here to calculate line descriptors. LBD is a binary descriptor that performs efficient and robust feature matching between different feature lines.

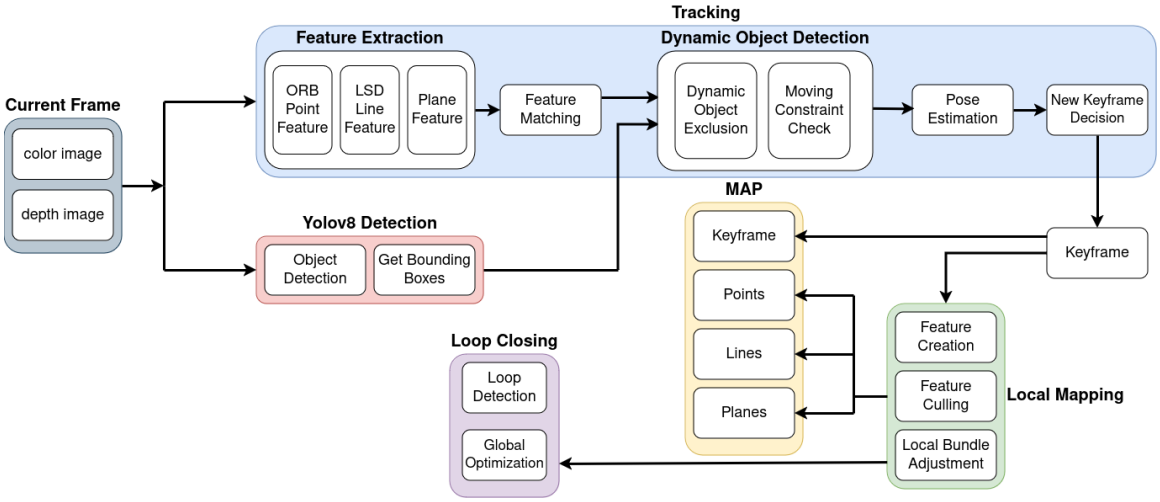


Fig. 2. Overview framework

For the line representation, the startpoints and endpoints of lines are used to represent lines.

3) Plane: The depth images captured by the RGB-D camera will be used here to extract planes. The method in [19] is used here for plane feature extraction and matching. For plane representation, the Hessian form of representing planes is used to express infinite planes. The representation is $\boldsymbol{\pi}^T = (\boldsymbol{n}^T, d^T)$ [25], in which **n** is the normal vector, d is the distance between the plane and the original point. The hessian form of representing planes is an over-parameterized representation, which will influence the nonlinear optimization step. A 3D plane has a degrees of freedom of 3, but the Hessian form has four parameters. Here, a spherical coordinate is introduced to represent planes. The representation of planes can be transformed from Hessian form to (1). n_x, n_y and n_z are components of the normal vector in x, y, z directions, ϕ and θ are the azimuth and elevation angles of the plane normal.

$$q(\boldsymbol{\pi}) = (\phi = arctan(n_y/n_x), \theta = arcsin(n_z), d) \quad (1)$$

C. Object Detection

In the proposed SLAM system, an AI-based computer vision model is used to detect dynamic objects. YOLO (You Only Look Once) [26] is a simple and widely-used object detection algorithm. Object detection is chosen here rather than semantic segmentation because object detection generally has a faster speed of processing RGB images. Semantic segmentation takes a lot more computation source than object detection if they detect the same amount of objects. The simplicity of robot application is essential because of the limited computation resources of a robot. Furthermore, the model training difficulty of semantic segmentation is much larger than object detection, making it very hard to implement in different kinds of complex scenes.

YOLO-v8 [27] is used here because it has an extremely fast detection speed suitable for real-time object detection. The YOLOv8s model trained on the COCO2017 dataset can identify 80 classes of objects, in which most of the common objects in indoor environments are included. To exclude the points which are not on this object, the distance of each point and all detected planes will be calculated. If the distance is smaller than a threshold, then the point will be identified as falling on a plane(possibly the floor or a wall nearby) and not belong to this object.

D. Dynamic Object Detection

Dynamic object detection is based on both object detection and feature-matching results. The main idea is that if most of the feature points and feature lines are detected as dynamic points and lines in the bounding box of this object, then the object is determined as a dynamic object. All detected objects with bounding boxes with enough features will be considered. Instead of calculating optical flow in [8], the matched feature points and lines in the previous feature-matching process will be used here. Firstly, the fundamental matrix of each object will be calculated using two sets of matched feature points and lines in the object bounding box between the current frame and the last frame. For feature lines, two sets of feature points are considered with their startpoints and endpoints. There are two steps to decide whether an object is dynamic. The first step is to calculate the fundamental matrix using RANSAC [28]. When calculating the fundamental matrix, epipolar constraint is used to identify whether a point is an outlier. The epipolar constraint is as (2) and (3).

$$\boldsymbol{P}_1 = [u_1, v_1, 1], \boldsymbol{P}_2 = [u_2, v_2, 1] \quad (2)$$

$$\boldsymbol{P}_2 \boldsymbol{F} \boldsymbol{P}_1^T = 0 \quad (3)$$

With the fundamental matrix, the distance between the points in the current frame and the epipolar lines can be calculated. The calculation method is in [8] and the equations are shown in (4) and (5). In these two equations, P_1 and P_2 are two matched points between the last frame and the

current frame in homogeneous coordinate form, Dis is the distance between the points in the current frame and the epipolar lines, F is the fundamental matrix. If the distance is bigger than a threshold ϵ_d, then the point is considered a dynamic point and an outlier. The second step is to calculate the outlier ratio. If the outlier ratio of the features is bigger than a threshold ϵ_r, the object is determined to be a dynamic object, and the bounding box is considered as a dynamic area. If the outlier ratio is less than ϵ_r, the object is considered static. The algorithm for detecting dynamic objects is shown in Algorithm 1.

In our algorithm, the ϵ_d is set to 0.1 meters and ϵ_r is set to 70%. These two parameters can be adjusted based on the real scenes or applications.

$$\boldsymbol{I}_1 = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \boldsymbol{F}\boldsymbol{P}_1^T \tag{4}$$

$$Dis = \frac{\boldsymbol{P}_2\boldsymbol{F}\boldsymbol{P}_1^T}{\sqrt{||X||^2 + ||Y||^2}} \tag{5}$$

If the number of feature points is fewer than 8, then the RANSAC or eight-point algorithm [29] cannot be used to calculate the fundamental matrix. Then, a second check will be applied to these objects using the fundamental matrix calculated from the static object with the most feature points.

E. Moving Constraint Check

To make full use of the semantic information, the moving constraint check will be applied to all detected objects. For moving rigid bodies constrained by planes(such as movable chairs, tables, etc.), the constraint relationship can be used to increase pose estimation accuracy. To check whether an object is a constrained moving object, the features in the bounding box of this object will be used. For feature points, the distances between all feature points and all detected planes will be calculated for the last frame and the current frame. If the difference in average distance of all points in these two frames is smaller than a threshold ϵ_m^p, then this moving object is considered constrained by this plane. For feature lines, the difference in the average angle of all feature lines in these two frames will be calculated. If the difference is smaller than a threshold ϵ_m^l, then it is considered a constrained moving object by this plane. Chairs and tables are common moving objects(constrained by the floor) in indoor environments. In this work, chairs and tables are assumed as possible constrained moving objects, and they will be verified during the moving constraint check process. The algorithm for detecting constrained moving objects is shown in Algorithm 2.

For constrained moving objects, the constraints between these objects and related planes will be considered in the pose estimation process.

Algorithm 1 Dynamic Object Detection

Input: *bounding boxes* B, *matched features* $f1(last\ frame)$, $f2(current\ frame)$
Output: *dynamic areas*

for i *in range of* $B.size()$ **do**
  if $f_2.size() > 8$ **then**
    $outliers,\ F = GetFundaMat(f_1[i],\ f_2[i])$
    $outliersNum = outliers.size()$
    $f_2[i]Num = f_2[i].size()$
    if $outliersNum\ /\ f2Num > \epsilon_r$ **then**
      $dynamic\ areas += B[i]$
    end if
  else
    $lessfeatureareas += B[i]$
  end if
end for
for i *in range of* $lessfeatureareas.size()$ **do**
  if $GetEpipolarLineDis(f_1[i],\ f_2[i],\ F) > \epsilon_r$ **then**
    $dynamic\ areas += B[i]$
  end if
end for
return *dynamic areas*

F. Pose Estimation

For pose estimation, the bundle adjustment process optimizes the camera orientation $\boldsymbol{R} \in SO(3)$ and position $t \in \mathbb{R}^3$. The factor graph of the proposed SLAM system is shown in Fig. 3 to illustrate the pose estimation process. The reprojection errors are divided into four factors: points, lines, planes, and features(points and lines) in constrained dynamic objects. For points, the reprojection errors are defined in (6), where $\boldsymbol{p}$ is the observed 2D pixel coordinates, $\boldsymbol{P}$ is the 3D matched points in world frame, Π is the project function from the world coordinate to camera coordinate.

$$e_p = \boldsymbol{p} - \Pi(\boldsymbol{R}_{cw}\boldsymbol{P} + \boldsymbol{t}_{cw}) \tag{6}$$

For lines, the reprojection errors are defined using the same method in [19], as shown in (7), where $\boldsymbol{l}$ is the observed 2D line, $\boldsymbol{P}_m$ is an end point of the matched 3D line.

$$e_l = \boldsymbol{l}^T\Pi(\boldsymbol{R}_{cw}\boldsymbol{P}_m^l + \boldsymbol{t}_{cw}) \tag{7}$$

Algorithm 2 Constrained Moving Object Detection

Input: *bounding boxes* B, *matched features* $f1(last\ frame)$, $f2(current\ frame)$, *planes* P
Output: *constrained dynamic areas*

for i *in range of* $B.size()$ **do**
  for j *in range of* $P.size()$ **do**
    $Diff = GetAvgDisDiff(f_1[i],\ f_2[i],\ P[j])$
    if $Diff < \epsilon_m$ **then**
      $constrained\ dynamic\ areas += B[i]$
    end if
  end for
end for
return *constrained dynamic areas*

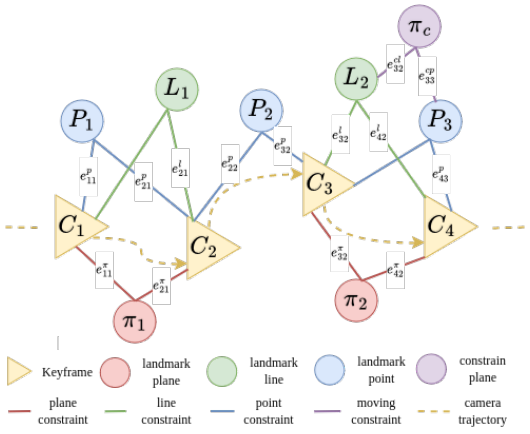


Fig. 3. The factor graph for the proposed SLAM system.

For planes, the reprojection errors are defined in (8), where $\boldsymbol{\pi}_c$ is the detected planes in the current frame, $\boldsymbol{\pi}_w$ is the matched planes in the world frame, $\boldsymbol{T} \in SE(3)$ is the transformation matrix.

$$e_\pi = q(\boldsymbol{\pi}_c) - q(\boldsymbol{T}_{cw}^{-T}\boldsymbol{\pi}_w) \tag{8}$$

For constrained dynamic objects, the errors are defined using the point and line features inside the bounding boxes of the objects. For points, the constraint is that the distance between the feature points and the constraining plane should remain the same, so the moving direction vectors for points should be perpendicular to the plane's normal vector. For lines, the constraint is that the angle between lines and the constraining planes should remain the same. The errors are shown in (9) and (10).

$$e_{cp} = \boldsymbol{n}_w^T\boldsymbol{P}_l - \boldsymbol{n}_w^T(\boldsymbol{R}_{cw}^{-1}\boldsymbol{P}_c - \boldsymbol{R}_{cw}^{-1}\boldsymbol{t}_{cw}) \tag{9}$$

In (9), $\boldsymbol{P}_l$ is the matched 3D points in the last frame under world coordinate, $\boldsymbol{n}_w$ is the normal vector of the plane under world coordinate, $\boldsymbol{P}_c$ is the matched 3D points in the current frame under camera coordinate.

$$e_{cl} = \frac{\boldsymbol{s}_l\boldsymbol{n}_w}{||\boldsymbol{s}_l||} - \frac{\boldsymbol{R}_{cw}^{-1}\boldsymbol{s}_c\boldsymbol{n}_w}{||\boldsymbol{R}_{cw}^{-1}\boldsymbol{s}_c||} \tag{10}$$

In (10), $\boldsymbol{s}_l$ is the matched 3D line direction vector in the last frame under world coordinate, $\boldsymbol{s}_c$ is the matched 3D line direction vector in the current frame under camera coordinate, $\boldsymbol{n}_w$ is the normal vector of the plane under world coordinate.

With all the error terms, the overall cost function is shown in (11), where ρ_h is the robust Huber cost function and $\boldsymbol{\Omega}$ is the information matrix, which is the inverse of the covariance matrix.

$$\begin{aligned} C = &\sum \rho_h(e_p^T\boldsymbol{\Omega}_p e_p) + \sum \rho_h(e_l^T\boldsymbol{\Omega}_l e_l) \\ &+ \sum \rho_h(e_\pi^T\boldsymbol{\Omega}_\pi e_\pi) + \sum \rho_h(e_{cp}^T\boldsymbol{\Omega}_{cp} e_{cp}) \\ &+ \sum \rho_h(e_{cl}^T\boldsymbol{\Omega}_{cl} e_{cl}) \end{aligned} \tag{11}$$

This cost function is solved using the Levenberg-Marquardt algorithm to get the optimal $\boldsymbol{T}_{cw}$, which is $\boldsymbol{T}_{cw} = \arg\min_{\boldsymbol{T}_{cw}}(C)$. The optimization process is implemented using g2o [30] library.

IV. Experiments and Results

For the experiments, the performance of the proposed SLAM system is evaluated both on TUM RGB-D dataset [31] and a real-world scene.

A. Experiments on TUM dataset

The experiments are conducted on a computer. The computer has an Intel Core i7-10700 CPU@2.90GHz x 16 processor, a NVIDIA GeForce GTX 1050 Mobile graphic. The operating system is Ubuntu 20.04. The TUM RGB-D dataset uses nine sequences containing dynamic environments and different camera motions. These sequences have environments in which people walk around or sit on chairs. Some representative state-of-the-art SLAM systems are chosen here to compare the performance of our SLAM system. ORB-SLAM2 is chosen as our system is built based on this. Planar-SLAM and Manhattan-SLAM are chosen because they are state-of-the-art methods of fusing point, line, and plane features. DS-SLAM and Dyna-SLAM were chosen because they are state-of-the-art SLAM systems of SLAM in dynamic environments. Absolute Trajectory Error(ATE) and Relative Pose Error(RPE) are used here to evaluate their performances. The ATE is used to evaluate the global performance of a SLAM system, and the RPE is used to evaluate the drift of a visual odometry system [31]. ATE and RPE are calculated by the evo tool [32]. The Root Mean Squared Error (RMSE) values of ATE and RPE are shown in Table I, Table II, and Table III.

The results of the experiments show that the proposed SLAM system performs much better than ORB-SLAM2, Planar-SLAM, and Manhattan-SLAM in highly dynamic environments(walking) as the dynamic objects and features are successfully excluded. Compared with DS-SLAM and Dyna-SLAM, the proposed system is slightly better as it combines multiple features and uses the structural constraint for moving objects (such as the chair in sequence fr2_desk_with_person) to improve the localization accuracy. Although in some sequences, the performance of the proposed system has less accuracy, the results are very close. However, in sequence fr3_walking_rpy, the proposed system has a large error compared to other walking sequences. This is because when the camera rotates nearly 90 degrees, the "person" cannot be detected by the object detection model, so the performance of the proposed system is partly based on the quality of the model. It is also a limitation of all semantic SLAM systems.

The trajectories of DS-SLAM and the proposed SLAM system for sequence fr3_walking_xyz are shown in Fig. 4. From (a) in C1 and C2, the trajectory of the proposed system fits more compared to DS-SLAM. From (b) and (c) in C1 and C2, the proposed system has less fluctuation in XYZ directions and less error on roll, pitch, and yaw.

TABLE I

COMPARISON RESULTS OF THE RMSE OF ATE IN METERS. THE BEST RESULTS ARE HIGHLIGHTED.

Sequences	ORB-SLAM2	DS-SLAM	Dyna-SLAM	Planar-SLAM	Manhattan-SLAM	Proposed System
fr2_desk_w_person	0.0532	0.0063	0.0061	0.0811	0.0191	**0.0056**
fr3_s_static	0.0119	0.0071	0.0071	0.0237	0.0084	**0.0068**
fr3_s_xyz	0.0134	0.0119	0.0151	0.0197	0.0102	**0.0118**
fr3_s_half	0.0193	**0.0147**	0.0168	0.0502	0.0658	0.0155
fr3_s_rpy	0.0301	0.0228	**0.0199**	0.0715	0.0286	0.0258
fr3_w_static	0.2513	0.0077	0.0079	0.3053	0.3027	**0.0075**
fr3_w_xyz	0.4032	0.1702	0.0180	0.2050	1.0429	**0.0179**
fr3_w_half	0.3508	0.0351	0.0333	0.6156	0.6692	**0.0329**
fr3_w_rpy	0.7631	0.4488	**0.0261**	0.4494	0.8907	0.1945

TABLE II

COMPARISON RESULTS OF THE RMSE OF RPE IN TRANSLATION PART IN METERS. THE BEST RESULTS ARE HIGHLIGHTED.

Sequences	ORB-SLAM2	DS-SLAM	Dyna-SLAM	Planar-SLAM	Manhattan-SLAM	Proposed System
fr2_desk_w_person	0.0104	**0.0032**	0.0112	0.0250	0.0046	**0.0032**
fr3_s_static	0.0097	**0.0044**	0.0124	0.0093	0.0051	0.0051
fr3_s_xyz	0.0153	0.0085	0.0201	0.0114	0.0079	**0.0075**
fr3_s_half	0.0210	0.0105	0.0245	0.0397	0.0107	**0.0103**
fr3_s_rpy	0.0141	0.0139	0.0132	0.0199	**0.0128**	0.0173
fr3_w_static	0.0249	**0.0049**	0.0103	0.0184	0.0133	0.0064
fr3_w_xyz	0.0293	0.0134	0.0214	0.0327	0.0238	**0.0124**
fr3_w_half	0.0276	0.0151	0.0259	0.0765	0.0557	**0.0135**
fr3_w_rpy	0.0642	**0.0209**	0.0315	0.0562	0.0317	0.0211

TABLE III

COMPARISON RESULTS OF THE RMSE OF RPE IN ROTATION PART IN RADIANS. THE BEST RESULTS ARE HIGHLIGHTED.

Sequences	ORB-SLAM2	DS-SLAM	Dyna-SLAM	Planar-SLAM	Manhattan-SLAM	Proposed System
fr2_desk_w_person	0.0078	**0.0071**	0.0081	0.0098	0.0073	0.0072
fr3_s_static	0.0097	0.0037	0.0065	0.0063	0.0041	**0.0036**
fr3_s_xyz	0.0079	0.0078	0.0083	0.0091	**0.0077**	0.0080
fr3_s_half	0.0091	0.0095	**0.0074**	0.0173	0.0115	0.0090
fr3_s_rpy	0.0120	0.0106	0.0105	0.0142	0.0102	**0.0100**
fr3_w_static	0.0165	0.0041	**0.0032**	0.0101	0.0074	0.0045
fr3_w_xyz	0.0335	0.0102	0.0331	0.0169	0.0145	**0.0096**
fr3_w_half	0.0423	0.0109	0.0102	0.0442	0.0402	**0.0097**
fr3_w_rpy	0.0964	0.0123	**0.0092**	0.0632	0.0176	0.0129

Furthermore, the time of tracking thread is also compared between the proposed SLAM system and DS-SLAM. For DS-SLAM, the main time cost to exclude dynamic points is calculating the optical flow pyramid and doing the moving consistency check. In the proposed method, calculating optical flow is not required. The main time cost is in finding corresponding matched points between two frames for each detected bounding box and calculating the fundamental matrix for each object. Considering both points and lines are used in the proposed method to detect dynamic objects, and DS-SLAM only uses points, the situation of the proposed method is divided into two parts: only use points and use points and lines. The experiments are implemented in the TUM walking sequences and assume persons are the only dynamic objects to be excluded. The comparison of average time cost per frame to detect dynamic objects is shown in Table IV.

TABLE IV

COMPARISON OF THE AVERAGE TIME COST ON DETECTING DYNAMIC OBJECTS

Sequence	DS-SLAM (s)	Proposed Method (s)	Proposed Method (only points) (s)
fr3_w_static	0.0813	0.0999	0.0521
fr3_w_xyz	0.0871	0.1060	0.0577
fr3_w_half	0.0842	0.1166	0.0535
fr3_w_rpy	0.0854	0.0827	0.0565

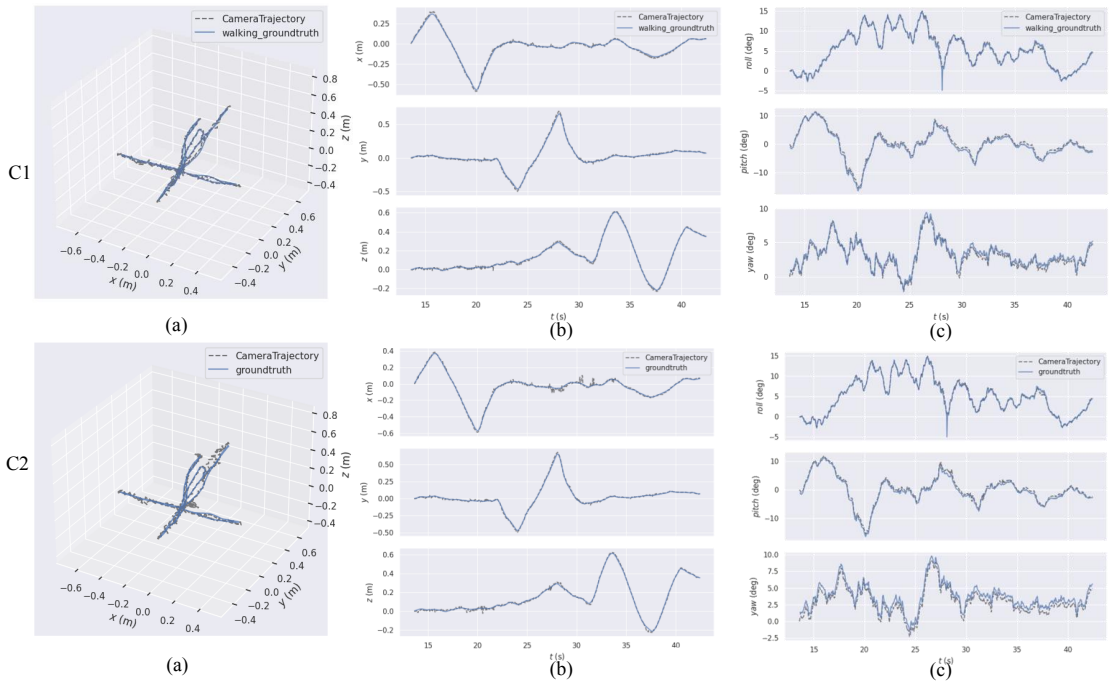


Fig. 4. C1 is the trajectory results of the proposed system. C2 is the trajectory results of DS-SLAM.

The results show that the proposed method improves about 35% on detecting dynamic objects time compared to DS-SLAM when both detect points. When the proposed method detect points, lines and planes, the time is slightly greater than DS-SLAM but it is still enough for real-time application. The proposed method even performs better under some circumstances (fr3_w_rpy). The limitation of the proposed method is that when it comes to multiple dynamic objects with large amount of features, the computation cost will increase a lot as it calculates the fundamental matrix for every detected object.

B. Experiments on a real-world scene

The real-world experiment conducted using variant of Smorphi robot [33]. It is equipped with a Jetson AGX Orin, which has an NVIDIA Ampere architecture GPU and an Arm Cortex-A78AE CPU. It also has a Realsense D435i camera to detect the environment. The robot and experiment environment are shown in Fig. 5. The map building is shown in Fig. 6. An example of dynamic object detection is shown in Fig. 7. In part of this experiment, a person was pushing a movable chair. The person was detected as a dynamic object and the chair was detected as a constrained dynamic object which was constrained by the detected floor.

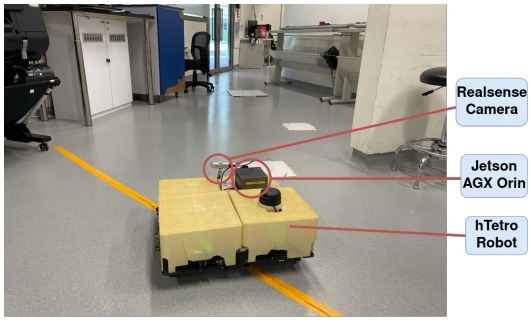


Fig. 5. The robot and the experiment environment.

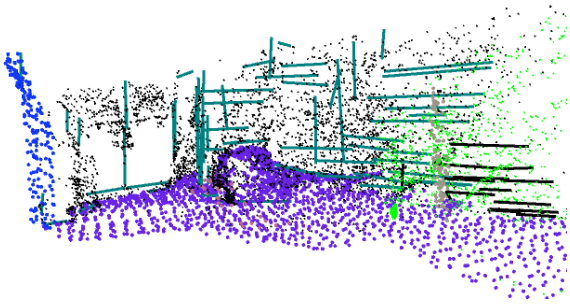
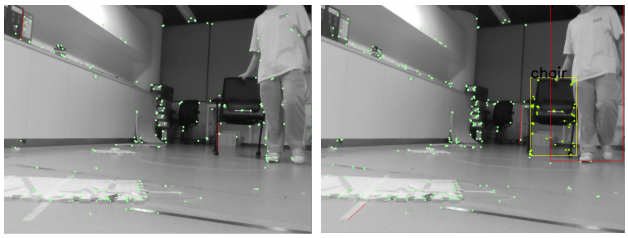


Fig. 6. Map building in the real-world environment.

V. Conclusion

This paper proposed a new semantic SLAM system that can perform robust localization in dynamic indoor environments. The proposed SLAM system can use object detection to identify objects with semantic information, quickly identify dynamic objects, exclude dynamic features, and combine points, lines, and plane features to create the map. It can identify constrained moving objects and use these constraints to improve localization accuracy. The experiment results on the TUM RGB-D dynamic dataset show that the proposed system can keep its robustness in dynamic environments and achieve an accurate localization accuracy compared with other state-of-the-art methods. The experiment in the real-world scene shows that the proposed system can be applied to real-time robot applications. While the proposed method



(a) (b)

Fig. 7. (a) is the image before excluding dynamic features. (b) is an example of dynamic object detection. The green dots represent feature points, red lines represent feature lines. The red bounding box means the detected object, the all feature points and lines on this object have been excluded. The yellow bounding box means a constrained moving object, yellow points represent constrained moving points.

provides new insight into Visual SLAM in dynamic environments, there are some limitations. Firstly, the proposed method assumes that all planes are flat and static, but in the real world, the planes are not entirely flat, and it may cause vibrations for constrained moving objects. Also, the motions of different parts vary for the non-rigid body, and the features cannot be constrained by planes. For future work, more structural features such as cube, quadric and moving constraints will be applied using line and plane features.

REFERENCES

[1] J. Cheng, L. Zhang, Q. Chen, X. Hu, and J. Cai, "A review of visual slam methods for autonomous driving vehicles," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 104992, 2022.

[2] R. Mur-Artal 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.

[3] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[4] 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.

[5] S. Yang, Y. Song, M. Kaess, and S. Scherer, "Pop-up slam: Semantic monocular plane slam for low-texture environments," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1222–1229.

[6] F. Shu, J. Wang, A. Pagani, and D. Stricker, "Structure plp-slam: Efficient sparse mapping and localization using point, line and plane for monocular, rgb-d and stereo cameras," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2105–2112.

[7] Y. Li, R. Yunus, N. Brasch, N. Navab, and F. Tombari, "Rgb-d slam with structural regularities," in *2021 IEEE international conference on Robotics and automation (ICRA)*. IEEE, 2021, pp. 11 581–11 587.

[8] 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 *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1168–1174.

[9] F. J. C. J. Bescos, Berta and J. Neira, "DynaSLAM: Tracking, mapping and inpainting in dynamic environments," *IEEE RA-L*, 2018.

[10] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and automation (ICRA)*. IEEE, 2017, pp. 4628–4635.

[11] Z. Liao, Y. Hu, J. Zhang, X. Qi, X. Zhang, and W. Wang, "So-slam: Semantic object slam with scale proportional and symmetrical texture constraints," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4008–4015, 2022.

[12] Y. Wu, Y. Zhang, D. Zhu, Z. Deng, W. Sun, X. Chen, and J. Zhang, "An object slam framework for association, mapping, and high-level tasks," *IEEE Transactions on Robotics*, 2023.

[13] 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.

[14] X. Yang, Y. Ming, Z. Cui, and A. Calway, "Fd-slam: 3-d reconstruction using features and dense matching," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8040–8046.

[15] 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.

[16] X. Hu, Y. Zhang, Z. Cao, R. Ma, Y. Wu, Z. Deng, and W. Sun, "Cfp-slam: A real-time visual slam based on coarse-to-fine probability in dynamic environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4399–4406.

[17] X. Zuo, X. Xie, Y. Liu, and G. Huang, "Robust visual slam with point and line features," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1775–1782.

[18] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "Pl-slam: Real-time monocular visual slam with points and lines," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 4503–4508.

[19] R. Yunus, Y. Li, and F. Tombari, "Manhattanslam: Robust planar tracking and mapping leveraging mixture of manhattan frames," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6687–6693.

[20] M. Hosseinzadeh, Y. Latif, and I. Reid, "Sparse point-plane slam," in *Australasian Conference on Robotics and Automation*, 2017.

[21] J. Zhang, G. Zeng, and H. Zha, "Structure-aware slam with planes and lines in man-made environment," *Pattern Recognition Letters*, vol. 127, pp. 181–190, 2019.

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[23] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 4, pp. 722–732, 2008.

[24] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency," *Journal of visual communication and image representation*, vol. 24, no. 7, pp. 794–805, 2013.

[25] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4605–4611.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[27] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[28] 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.

[29] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 6, pp. 580–593, 1997.

[30] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, "g2o: A general framework for (hyper) graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 9–13.

[31] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[32] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.

[33] I. D. Wijegunawardana, S. B. P. Samarakoon, M. V. J. Muthugala, and M. R. Elara, "Fmea-based coverage-path-planning strategy for floor-cleaning robots," *Advanced Intelligent Systems*, vol. 5, no. 11, p. 2300260, 2023.