RGB-D SLAM for Autonomous Mobile Robot Toward the Dynamic Environment

Hong-Yu Chen, Chin-Sheng Chen

***Abstract*—Simultaneous Localization and Mapping (SLAM) is widely used in Autonomous Mobile Robot (AMR) applications. Most SLAM methods are applied in a static environment, which significantly limits their application in real-world settings. This study proposes an RGB-D SLAM based on ORB-SLAM2 that is more robust to dynamic environments and is implemented in real-time. Object detection and multi-view geometry is used to detect and prevent moving objects impacting system, our SLAM system without the need for special hardware such as GPU. In order to verify the proposed method, this study conducts experiments using the public TUM dataset and real environment also compares the system time cost with that for other SLAM applications in dynamic environments.**

***Index Terms*—simultaneous localization and mapping (SLAM), dynamic environment, object detection, multi-view geometry.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a process by which a robot uses different sensors to construct a map of the environment and, calculate its position in the environment. The low cost of visual sensors, means that many SLAM is use visual sensors give good result, such as the LSD-SLAM [1], the ORB-SLAM2 [2], and the RTAB-MAP [3].

Depending on whether to visual odometry is used to extract the feature points [4], visual SLAM is classified as direct and indirect. The direct method is assumes that the gray level for a three-dimensional point that is measured at each angle of view does not change. The indirect method, which is also called feature-based method, is currently the most commonly used method. The feature-based method selects representative points from the image and uses these feature points to represent the description in the image. The change in the pose of the camera per unit time is calculated using the displacement of the feature points in the continuous image.

Most Visual SLAM apply to a static environment [5] [6]. For the localization process for Visual SLAM, feature points are landmarks, so landmarks must be static for SLAM. However, this assumption is not valid for a real environment. There are many moving objects in the world, such as walking people and moving cars, and moving landmarks in dynamic environments affect the accuracy of localization and mapping.

In order to increase the accuracy of SLAM in dynamic environments, dynamic objects in the scene must be detected, prevented from being tracked. Some dynamic SLAM relies on purely geometrical approaches to detect moving objects but these usually fail to detect the previous dynamic object, such as sitting people or parked car, if their initial state is stationary, so the odometry drifts or there are wrong loop closures. We can rely neural networks, such as object detection and semantic segmentation, to give information about the objects in the scene. This study proposes a RGB-D SLAM framework for dynamic environments and runs it in the robot operating system (ROS) [7]. This method selects reliable feature points to adapt to a dynamic environment. The main contributions of this paper are:

- The proposed SLAM method is more accurate than ORB-SLAM2 in highly dynamic environments.
- The proposed method runs in real-time and performs well in a dynamic environment.

In this paper, Section II reviews the advantages and disadvantages of related SLAM work. Section III introduces the proposed system in detail. Section IV gives the results of experiments using this method for a public dataset. The conclusion is present in section V.

II. The Advantages and Disadvantages of Related Visual SLAM

A. Visual SLAM

Davision et al. developed the first Visual SLAM system that used a monocular camera: MonoSLAM [8]. Klein et al. [9] developed the first Visual SLAM that that uses keyframe bundle adjustment in 2007, using two independent threads for mapping and tracking. This was also the first time that visual SLAM dissociated the concept of front and back ends. Mur-Artal et al. [10] proposed a system with three threads, which is called ORB-SLAM, and can be use with monocular, stereo, and RGB-D camera.

Visual SLAM uses either direct or feature-based methods. Feature-based methods use image feature extraction and matching to calculate the pose, loop closing and bundle adjustment. Examples of feature point Visual SLAM methods include RGB-D SLAM [11] and ORB-SLAM. Direct methods optimize the image pixel intensity. Kinect-Fusion [12], used only depth data to create a dense model and used ICP to track the camera pose. DSO [13] combines photometric error minimization with camera motion optimization to obtain visual odometry. There are also semi-direct systems, such as LSD-SLAM, which apply the direct method to a semi-direct

This paper was submitted on 2022.

This work was supported in part by the Ministry of Science and Technology, Taiwan under grant No. MOST 109-2221-E-027 -044 -MY3.

Hong-Yu Chen is with the Graduate Institute of Automation Technology, National Taiwan University of Technology, Taipei, Taiwan.

Corresponding author: Chin-Sheng Chen, he is currently a Professor with the Graduate Institute of Automation Technology at National Taiwan University of Technology. (e-mail: saint@mail.ntut.edu.tw)

monocular SLAM. Features need not be calculated and a semi-dense map is constructed.

B. Visual SLAM in Dynamic Environments

Most visual SLAM systems assumed that the environment is static, so they cannot be used in a dynamic environment. Processing dynamic objects in the scene usually involve treating them as noise and using direct or feature-based filtering methods. ReFusion and StaticFusion are two direct methods for RGB-D cameras [14, 15]. ReFusion combines the TSDF model representation of KineticFusion with pure geometric methods to filter dynamic content.

Dib et al. proposed a dense visual odometry system for RGB-D cameras in dynamic environments using RANSAC [16]. Kim et al. determined the static part of the scene by computing the difference between the continuous depth image that is projected over the same plane [17]. Feature-based methods include that of the study by Li and Lee, which used depth edge points. These have an associated weight representing the probability of belonging to dynamic objects [18].

However, this method cannot detect a priori dynamic objects in a scene if they are static, such as cars or people. DS-SLAM addresses the problem of detecting dynamic objects by combining optical flow and semantic segmentation [19]. SOF-SLAM is a feature-based method developing ORB-SLAM2, which combines semantic segmentation and epipolar geometry to filter outliers [20].

DynaSLAM uses Mask R-CNN instance segmentation to obtain pixel-wise information for prior dynamic objects, in order to remove outliers in the scene [21, 22]. DynaSLAM is highly accurate and robust, but it does not perform in real-time because it involves a high computational burden.

Object detection can be performed in real-time. A previous study compared instance segmentation and object detection to filter priori dynamic features, in order to determine which is fastesr and most accurate [23]. A Mask R-CNN and YOLO were respectively used for instance segmentation and object detection, which is much improved.

C. Octomap

A octomap is a flexible map that can be updated at any time. An octree is shown in Fig. 1. A large cube is continuously divided equally into eight pieces until it becomes the smallest square. The entire large square is regarded as the root node and the smallest block is considered the leaf node. The node in the octree stores information about whether it is occupied, and 0 indicates blank status, and 1 indicates occupation status.

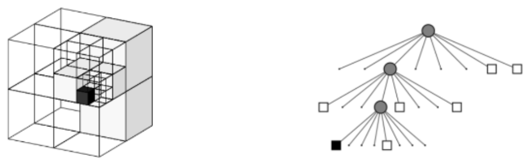


Fig. 1 Schematic diagram of an octree

III. SLAM SYSTEM AND ALGORITHM INTRODUCTION

This section details the proposed SLAM system. The framework for the method and the object detection network are described, and the implementation is detailed.

A. Overall workflow

The proposed method uses an object detector into the system and outputs include a bounding box and an object category. Our approach is based on ORB-SLAM2. ORB-SLAM2 is a 3D localization and mapping algorithm that uses ORB features. The advantage of ORB-SLAM2 is that it has good results in processing speed tracking and mapping accuracy. Compared to the original ORB-SLAM2, we have added three modules:

1). Object detection is added to determine the categories of objects in an image before tracking

2). A module to determine whether the feature points are dynamic is used to determine if these feature points can be used when tracking.

3). An octomap is constructed and used for more advanced tasks, such as navigation.

The framework for the proposed system is shown in Fig. 2, which mainly includes five threads. The object detection thread is used to calculate the object's category in the image. The object is detected first for each input frame. The tracking thread firstly extracts the ORB feature points, and feature points are associated with the category information. Using the category of the feature points and multi-view geometry, the feature points on dynamic objects are define and culled. When the tracking module determines whether the frame is a key frame, the system creates the a dense map.

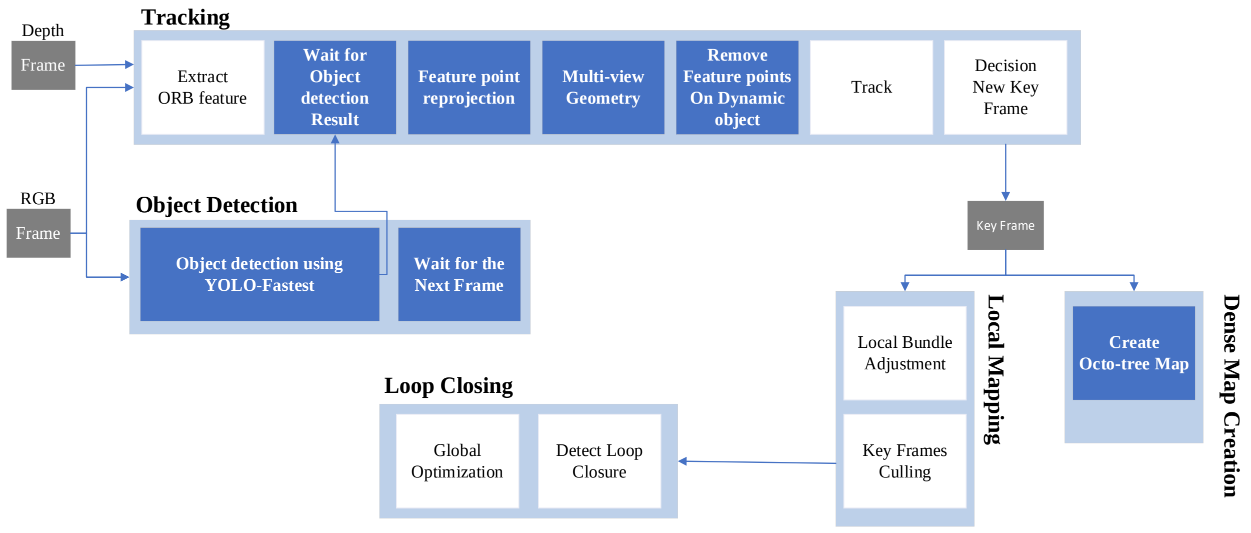


Fig. 2 The framework for the proposed method. Compared to the original ORB-SLAM2, a tracking thread, an object detection thread are added and a dense map thread is created (White words on blue background). The feature points considered dynamic will not be used for tracking.

B. Object detection

The most important task of object detection is to efficiently and accurately identify and locate objects in categories from images. Object detection is used in many fields, such as automatic driving, video surveillance, and intelligent and industrial defect detection in products. The proposed system uses the lightweight object detection YOLO-Fastest. The object detection network is trained on the MS COCO dataset and can detect 80 classes and acquire real-time, high quality, and persuasive object detection results.

YOLO-Fastest runs in real-time on low-cost devices. An RGB frame is the input and a corresponding bounding box is the output. These bounding boxes label the area in the frame with several predefined categories, such as persons, cars or clocks. These bounding boxes are used by visual SLAM to separate dynamic objects. A person, car, or bicycle are regarded as prior dynamic objects for the proposed system.

C. Feature point reprojection

After object detection, all prior dynamic objects are detected, and the position of feature points in the bounding box is determined. If all of the feature points in the bounding box are removed directly, two problems ensue:

1). Some feature points do not belong to dynamic objects, and reliable feature points are removed, show in Fig. (a).

2). The bounding box is huge if a dynamic object is too close to the camera. If all feature points are removed directly, too few feature points remain in the image, so the system fails to track, as shown in 3 (c).

This method is not appropriate, so feature point reprojection is used. Fig. 4 shows the scheme for this method.

The RGB frame is projected onto the depth image and the depth value for all feature points in the bounding box is calculated. Then we take the mode of the depth value after the statistics and treat the feature points with similar depth values as on the dynamic object, and the others are static. The result of feature point reprojection is shown in Fig. 3.

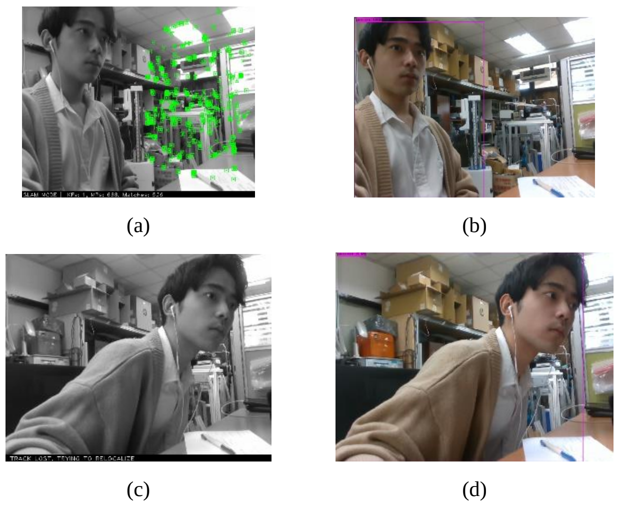


Fig. 3 All feature points in the bounding box are removed so: (a) Reliable points are removed. (b) The detection bounding box. (C) Subjects are too close to the camera so tracking is lost (d) The detection bounding box.

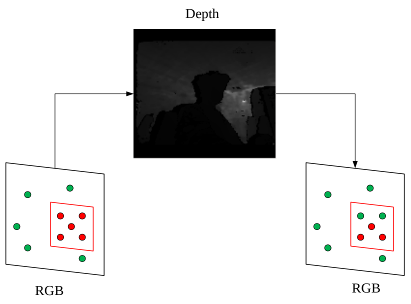


Fig. 4 Feature points reprojection. Green points are feature points that belong to a static object, and red points are feature points that belong to a dynamic object.

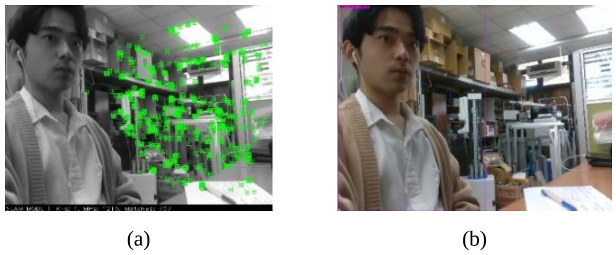


Fig. 3 (a) Result after feature points reprojection and (b) The detection bounding box.

D. Removal of dynamic feature points using multi-view geometry

Most dynamic objects are removed and are not used for tracking and mapping using the YOLO-fastest. However, some objects cannot be detected because they are not prior dynamic objects that are movable. Examples include a chair that someone is moving.

For each input frame, the previous keyframe is used. The method for selecting a keyframe is the same as that for ORB-SLAM2. The projection of each keypoint x from the previous keyframe onto the current frame is calculated to determine the keypoint x' and the depth projection z_{proj}. ORB-SLAM2 determines the keypoint x using ORB feature extraction. For each keypoint x, the corresponding 3D point is X. The angle α between $\overline{Xx}$ and $\overline{Xx'}$, which is the parallax angle, is then calculated. If this angle is greater than 15°, the point is obscured so it is ignored. For the TUM dataset, a static object with a parallax angle that is greater than 15° is mistaken for a dynamic object because there is a different viewpoint. The depth z' of the keypoint x' of the current frame is measured using the depth and compared with z_{proj}. If the difference Δz = zproj – z' exceeds a threshold τ_z, the keypoint x' is deemed to be a dynamic object. The architecture is shown in Fig. 4. A velue for τz = 0.6m is a reasonable choice, as shown by experiment.

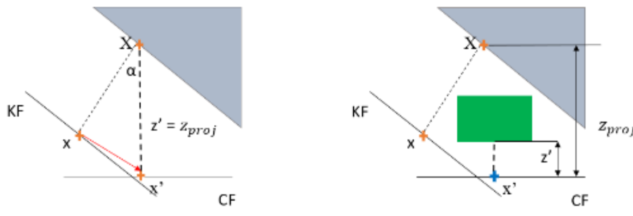


Fig. 4 Keypoint x from Key Frame (KF) is projected into Current Frame (CF) to give the result x' with depth z'. The project depth zproj is then calculated. If the difference Δz = zproj – z' is greater than threshold τz the feature points are deemed to be dynamic.

E. Removal of Outliers

Object detection and multi-view geometry address each other's disadvantages, so their combination allows accurate tracking and mapping. When the images pass through the object detection process and multi-view geometry, the keypoints for the dynamic object are removed from the image. The segmentation mask uses a geometric method if an object is detected using both methods. The result is shown in Fig. 5.

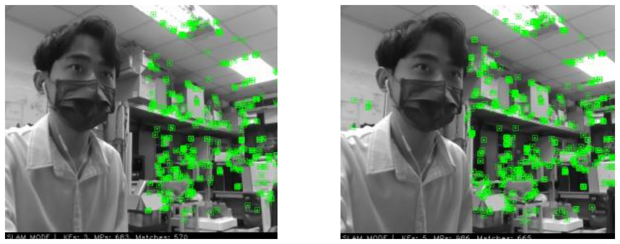


Fig. 5 Comparison of the results :(a) ORB-SLAM2 and (b) the proposed method.

F. Creating an Octomap

The mapping system has three nodes, a driver node, a pose estimation, and a mapping node, as shown in Fig. 6. The first node is the driving node, which collects data from the camera. The second node uses the proposed method for pose estimation. The last node is used to construct a map by receiving the image and pose data from the first and second nodes to perform point cloud splicing and construct an octomap.

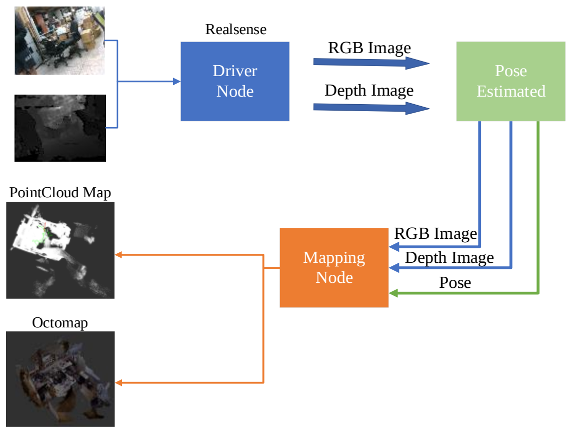


Fig. 6 The architecture for creating an octomap

IV. Experiment Results

This section details the experimental results to illustrate the accuracy of the proposed method. The system uses the public TUM dataset, and the results are compared with those for an open-source SLAM method that is also suited to dynamic environments: DynaSLAM. The system with the original ORB-SLAM2 is used to illustrate the improvement using the proposed method in a dynamic environment. Each sequence is run five times and the median result is shown. The experiments are performed on a PC with an Intel i5 CPU, 16GB memory, without a GPU.

A. TUM dataset

TUM Computer Vision Lab features the TUM dataset. It is composed of 39 sequences that are recorded using a Microsoft RGB-D Kinetic camera for different indoor scenes. The camera moves along the xyz axis. For the sequence that is named sitting, two people are sitting in front of a desk and chatting, so the degree of motion is low. For the walking sequence, two people walk around a desk. This dataset is highly dynamic, so it is challenging for SLAM systems. The dataset provides the ground truth for the camera trajectory, which is used to determine the performance of the proposed method.

The results for the TUM dynamic datasets are compared with those for ORB-SLAM2. The absolute translation error

(ATE) and relative pose error (RPE) are used to compare. The ATE measures the global consistency, and the RPE measures the translational and rotational drift.

The RMSE, values are calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P_{est,i} - P_{gt,i})^2}{n}} \quad (1)$$

where *n* represenst the number of sampling points; P_{est} is the calculated trajectory and P_{gt} is the trajectory ground truth. The RMSE value demonstrates the robustness of the system. This study also shows how better the proposed approach is than the original ORB-SLAM2.

The values are calculated as:

$$I = \left(\frac{e_{orb} - e_{propsed}}{e_{orb}}\right) \times 100\% \quad (2)$$

where I represents the improvements due to the proposed method, compared to the ORB-SLAM2 result, e_{orb} is the RMSE value for ORB-SLAM2 and $e_{propsed}$ is the RMSE value for the proposed method. Table I -

Table III show the experimental result for the experiment. Fig. 9 - Fig. 11 show the *ATE* and *RPE* results compared to ORB-SLAM2. The calculated trajectory is shown as a blue line in each figure, and the ground truth is shown as a black dashed line.

TABLE I THE ABSOLUTE TRAJECTORY ERROR (ATE) FOR ORB-SLAM2 AND THE PROPOSED METHOD FOR THE TUM DATASET (UNITS: M)

Sequences	ORB-SLAM2	Proposed method	Improvements
	RMSE	RMSE	RMSE
fr3_walking_xyz	0.6868	0.0181	97.36%
fr3_walking_rpy	3.1395	0.7704	75.46%
fr3_walking_static	0.5190	0.0153	97.05%
fr3_walking_half	0.6567	0.0578	91.19%
fr3_sitting_xyz	0.0078	0.0075	3.84%

TABLE II THE TRANSLATION DRIFT (RPE) FOR ORB-SLAM2 AND THE PROPOSED METHOD FOR THE TUM DATASET (UNITS: M/S)

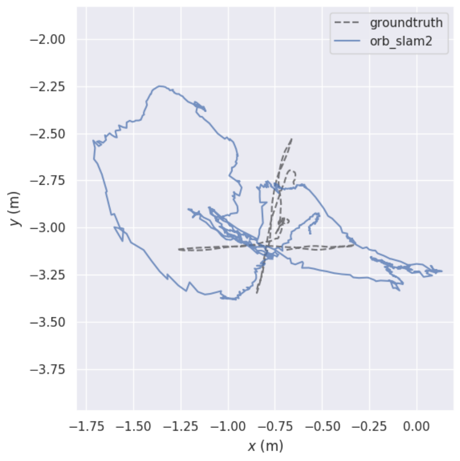
Sequences	ORB-SLAM2	Proposed method	Improvements
	RMSE	RMSE	RMSE
fr3_walking_xyz	1.0001	0.0261	97.39%
fr3_walking_rpy	0.9315	0.3312	64.44%
fr3_walking_static	0.5190	0.0273	94.74%
fr3_walking_half	1.0045	0.0844	91.60%
fr3_sitting_xyz	0.0122	0.0109	10.66%

TABLE III THE ROTATION DRIFT (RPE) FOR ORB-SLAM2 AND THE PROPOSED METHOD FOR THE TUM DATASET (UNITS: DEG/S)

Sequences	ORB-SLAM2	Proposed method	Improvements
	RMSE	RMSE	RMSE
fr3_walking_xyz	19.3829	0.6641	96.57%
fr3_walking_rpy	15.2400	5.3749	64.73%
fr3_walking_static	9.2761	0.5953	93.58%
fr3_walking_half	22.7469	1.3020	94.28%
fr3_sitting_xyz	0.3552	0.3319	6.56%

Table I -

Table III show that the proposed method gives significantly better results for highly dynamic scenes than ORB-SLAM2. However, the difference is less for a low dynamic scene (fr3_sitting_xyz). For low dynamic scenes, the feature points on the object are used, and they do not affect the accuracy of tracking, so the results for both methods are similar.


(a)

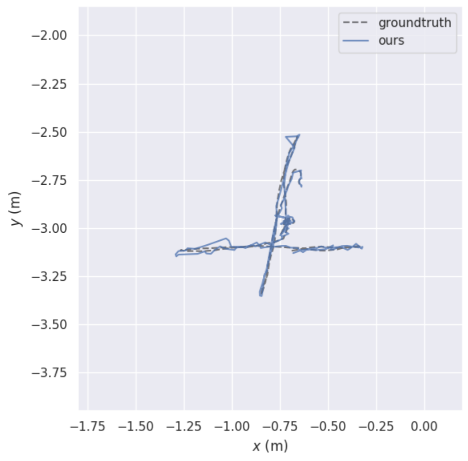
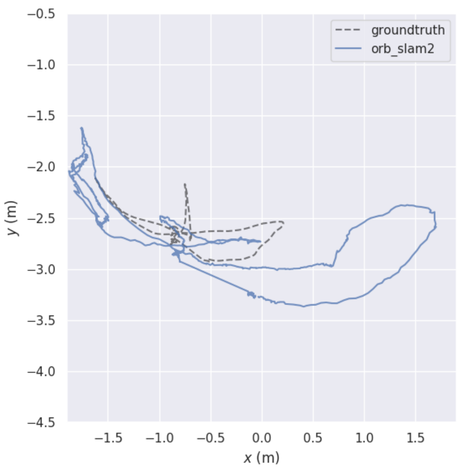

(b)

Fig. 7 Ground truth and estimated trajectory for the sequence fr3_walking_xyz for: (a) ORB-SLAM2 and (b) the proposed method


(a)

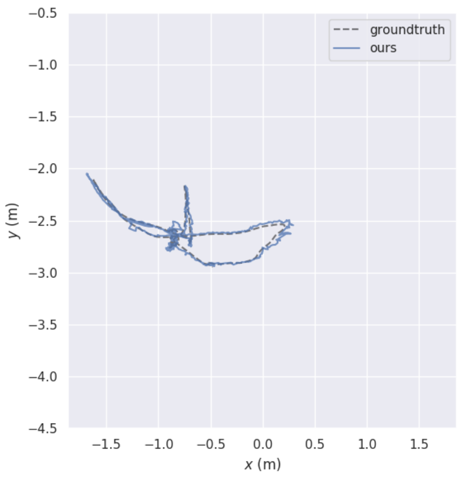
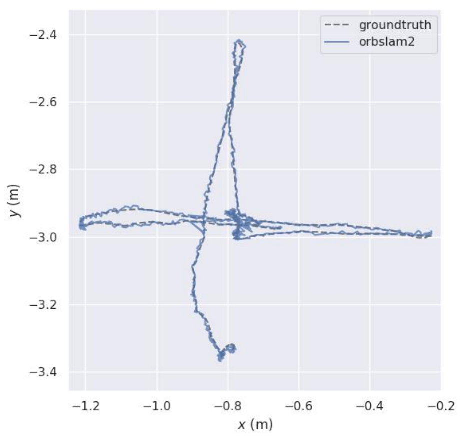

(b)

Fig. 8 Ground truth and estimated trajectory for the sequence fr3_walking_half for: (a) ORB-SLAM2 and (b) the proposed method


(a)

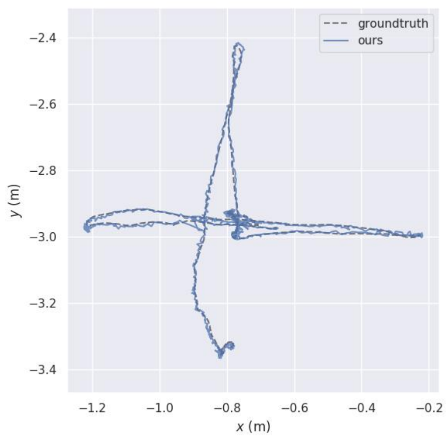

(b)

Fig. 9 Ground truth and estimated trajectory for the sequence fr3_sitting_xyz for: (a) ORB-SLAM2 and (b) the proposed method

B. Real environment

In Real Environment, We set two scenarios. One is straight another is rotated and use keyboard control the AMR for point-to-point movement. The AMR we used is developed by TECO are shown in Fig. 10. First, we set the camera parameters, as shown in Table IV and then substitute the camera parameters into the parameters of the SLAM proposed in this study, as shown in Table V.

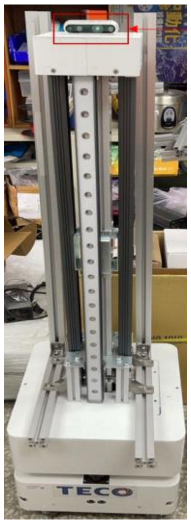


Fig. 10 AMR used in real environment

Table IV Camera parameters

depth width	1280
depth height	720
depth fps	30
color width	1280
color height	720
color fps	30

Table V SLAM parameters

scaleFactor	1.2
nFeatures	1500
nLevels	8
iniThFAST	20
minThFAST	7
camera_fps	30
color_fps	30
TheDepth	40
depth_map_factor	1000
camera_fx	920.79
camera_fy	921.10
camera_cx	663.68
camera_cy	374.82
camera_baseline	15.50

1) Scenario straight

In scenario straight we use keyboard control AMR to move from point A to B as shown in Fig. 11 and there is a person walking back and forth in front of the AMR. The AMR's perspective is shown in Fig. 12. Fig. 13 shows the comparison between the ORB-SLAM2(green line) and the poses of the proposed method (red line) and the groundtruth(blue line) in this scenario. The absolute trajectory error compared with ORB-SLAM2 as shown in Table VI.

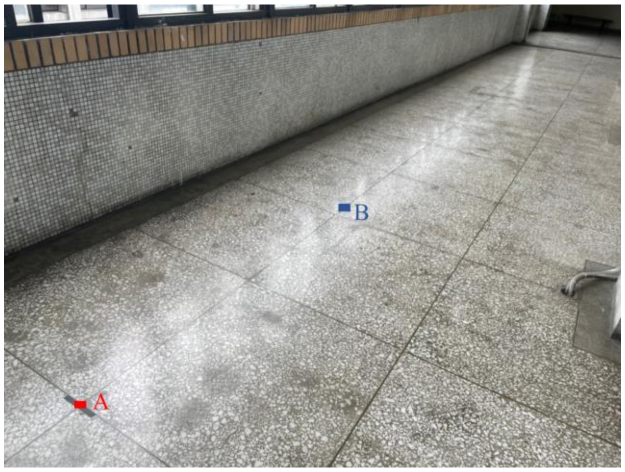
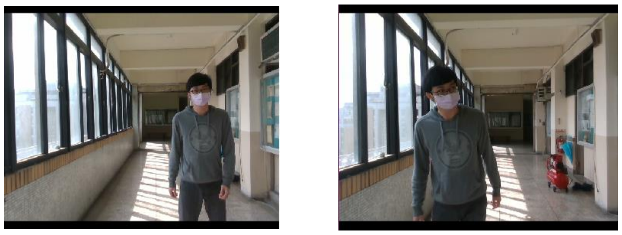


Fig. 11 AMR start point(A) and stop point(B)



(a) (b)

Fig. 12 AMR's perspective at different times in scenario straight; (a)t=6s, (b)t=8s

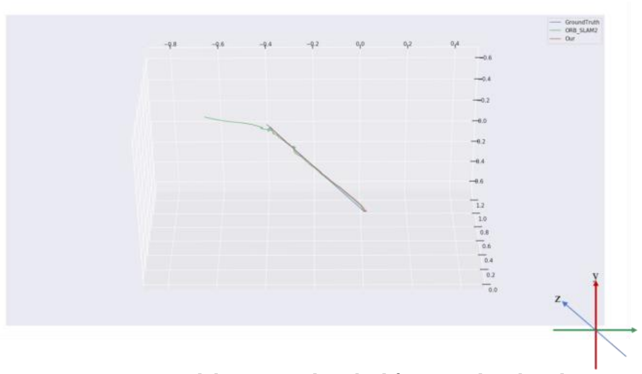


Fig. 13 ORB-SLAM2 and the proposed method for ground truth and estimated poses in scenario straight

TABLE VI THE ABSOLUTE TRAJECTORY ERROR (ATE) FOR ORB-SLAM2 AND THE PROPOSED METHOD FOR SCENARIO STRAIGHT

Sequences	ORB-SLAM2	Proposed method	Improvements
	RMSE	RMSE	RMSE
Straight	18.47	2.06	88.85%

2) Scenario rotate

In scenario rotate, AMR is control by keyboard from point A to B and then turn left and walk to point C as shown in Fig. 14. While AMR is walking, someone walk back and forth in front of the AMR. The AMR's perspective is shown in Fig. 15. Fig. 16 shows the comparison between the ORB-SLAM2(green line) and the poses of the proposed method (red line) and the groundtruth(blue line) in this scenario. The absolute trajectory error compared with ORB-SLAM2 as shown in Table VII.

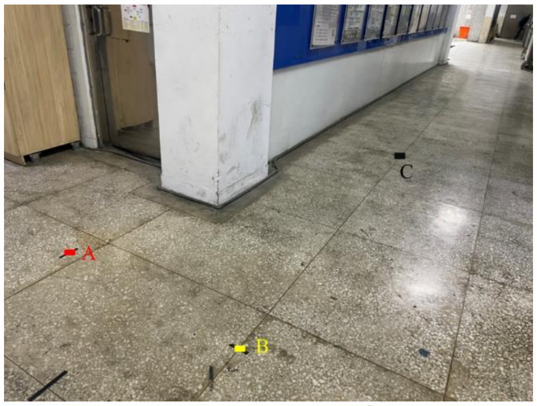


Fig. 14 AMR start point(A), rotation point(B) and stop point(C)

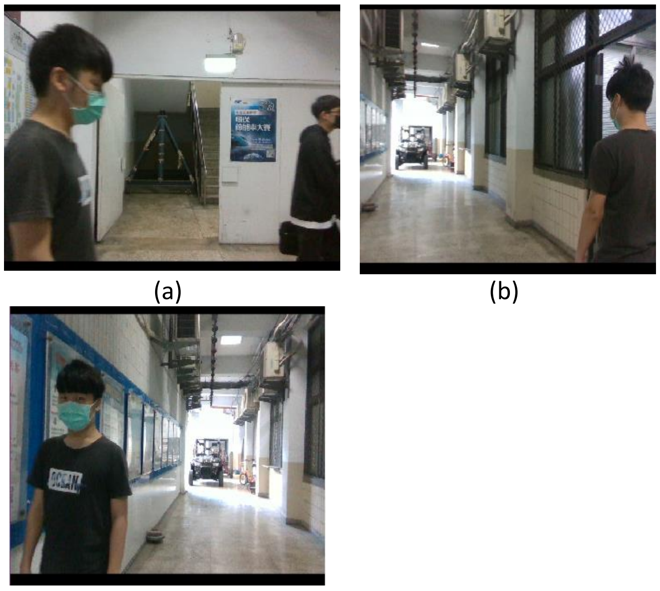


Fig. 15 AMR's perspective at different times in scenario rotate; (a)t=15s, (b)t=25s, (c)t=40s

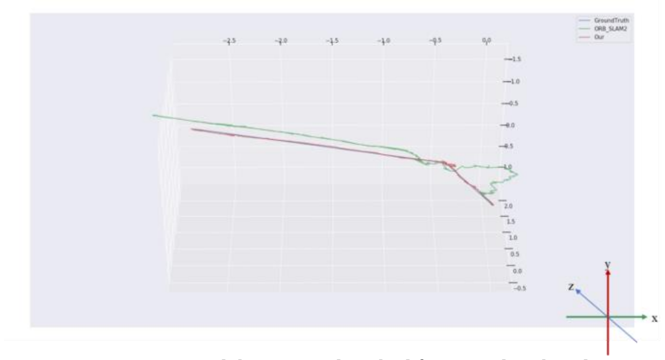


Fig. 16 ORB-SLAM2 and the proposed method for ground truth and estimated poses in scenario rotate

TABLE VII THE ABSOLUTE TRAJECTORY ERROR (ATE) FOR ORB-SLAM2 AND THE PROPOSED METHOD FOR SCENARIO ROTATE

Sequences	ORB-SLAM2	Proposed method	Improvements
	RMSE	RMSE	RMSE
Rotate	39.40 cm	4.04 cm	89.75%

C. The trade-off between real-time and accuracy

Most SLAM methods for a dynamic environment increase accuracy but cannot operate in real-time. This makes it challenging to perform on AMR. The proposed method maintains accuracy and operates in real-time. All tests are performed using a PC with an Intel Core i5 CPU and 16GB memory running on Ubuntu Linux 16.04LTS. The result for DynaSLAM are shown in Table VIII. The result in Table VIII., show that the proposed method gives a significantly better system tracking time without much sacrifice in accuracy.

D. Build an octomap

We used the method described in section III. And run on TUM dataset (fr1_room) to test our mapping effect. The result is shown in Fig. 17. The octomap, is used for more advanced tasks such as obstacle avoidance.

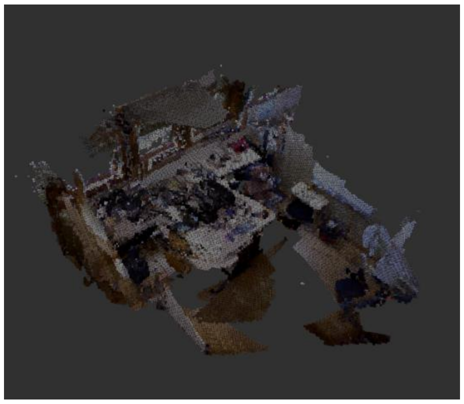


Fig. 17 The octomap result running in the fr1_room

V. CONCLUSIONS

This study proposes a SLAM method that is suited to a dynamic environment. The proposed system is based on ORB-SLAM2. Object detection, feature points reprojection and multi-view geometry is added. The object detection and multi-view geometry information is used to filter out and remove outliers. Only reliable feature points are used to calculate the trajectory. The result of experiments using a public dataset from TUM Computer Vision Lab and test in real environment demonstrate that the proposed system performs well in a highly dynamic environment. In the future, if we upgrade the hardware , we can try to use two-stage network to remove prior dynamic object. When building a map with two-stage networks, semantic information can be added to provide more information for the map to facilitate subsequent development and use.

TABLE VIII COMPARE TRACKING TIME AND ATE TO DYNASLAM.

Sequences	Proposed method		DynaSLAM	
	Track each frame time (s)	RMSE (cm)	Track each frame time (s)	RMSE (cm)
fr3_walking_xyz	0.1578	1.81	4.9089	1.51
fr3_walking_rpy	0.1736	21.60	7.7722	15.12
fr3_walking_static	0.2976	1.53	6.2772	0.71
fr3_walking_half	0.3150	5.78	6.8157	2.87
fr3_sitting_xyz	0.2223	0.75	5.4957	0.63

ACKNOWLEDGMENT

This study was financially sponsored by the Ministry of Science and Technology under project No. MOST 109-2221-E-027 -044 -MY3.

REFERENCES

[1] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in European conference on computer vision, 2014: Springer, pp. 834-849.
[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] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," Journal of Field Robotics, vol. 36, no. 2, pp. 416-446, 2019.
[4] A. Li, X. Ruan, J. Huang, X. Zhu, and F. Wang, "Review of vision-based simultaneous localization and mapping," in 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2019: IEEE, pp. 117-123.
[5] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in 2011 international conference on computer vision, 2011: IEEE, pp. 2320-2327.
[6] G. Graber, T. Pock, and H. Bischof, "Online 3d reconstruction using convex optimization," in 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 2011: IEEE, pp. 708-711.
[7] M. Quigley et al., "ROS: an open-source Robot Operating System," in ICRA workshop on open source software, 2009, vol. 3, no. 3.2: Kobe, Japan, p. 5.
[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," IEEE transactions on pattern analysis and machine intelligence, vol. 29, no. 6, pp. 1052-1067, 2007.
[9] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in 2007 6th IEEE and ACM international symposium on mixed and augmented reality, 2007: IEEE, pp. 225-234.
[10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," IEEE transactions on robotics, vol. 31, no. 5, pp. 1147-1163, 2015.
[11] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," IEEE transactions on robotics, vol. 30, no. 1, pp. 177-187, 2013.
[12] R. A. Newcombe et al., "Kinectfusion: Real-time dense surface mapping and tracking," in 2011 10th IEEE international symposium on mixed and augmented reality, 2011: IEEE, pp. 127-136.
[13] R. Wang, M. Schworer, and D. Cremers, "Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras," in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3903-3911.
[14] E. Palazzolo, J. Behley, P. Lottes, P. Giguere, and C. Stachniss, "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: IEEE, pp. 7855-7862.
[15] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers, "Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments," in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018: IEEE, pp. 3849-3856.
[16] A. Dib and F. Charpillet, "Robust dense visual odometry for RGB-D cameras in a dynamic environment," in 2015 International Conference on Advanced Robotics (ICAR), 2015: IEEE, pp. 1-7.
[17] 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.
[18] S. Li and D. Lee, "RGB-D SLAM in dynamic environments using static point weighting," IEEE Robotics and Automation Letters, vol. 2, no. 4, pp. 2263-2270, 2017.
[19] C. Yu et al., "DS-SLAM: A semantic visual SLAM towards dynamic environments," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018: IEEE, pp. 1168-1174.
[20] L. Cui and C. Ma, "SOF-SLAM: A semantic visual SLAM for dynamic environments," IEEE Access, vol. 7, pp. 166528-166539, 2019.
[21] 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.
[22] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961-2969.
[23] J. C. V. Soares, M. Gattass, and M. A. Meggiolaro, "Visual SLAM in human populated environments: exploring the trade-off between accuracy and speed of YOLO and Mask R-CNN," in 2019 19th International Conference on Advanced Robotics (ICAR), 2019: IEEE, pp. 135-140.