2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Object Covisibility Graph for Change Detection and 3D Object-Oriented Map Revision in Semi-Static Scenes

Ziquan Wang[1], Masahiko Mikawa[2] and Makoto Fujisawa[2]

***Abstract*—Recently, Object-oriented SLAM(Object SLAM) has attracted extensive research due to its ability to perceive the environment at a 3d object level. Existing object SLAM methods mostly focus on constructing 3d object map for static objects or mitigating the impact of currently dynamic objects on localization and mapping. However, detection of semi-static objects whose position change while unobserved still pose a significant challenge, resulting in outdated maps, which could lead to localization and robot application failures. In this paper, we propose a method to compare current observation with the existing map, enabling the continuous detection and updating of outdated sections within the map caused by position-changing semi-static objects. First, we introduce Object Covisibility Graph(OCG) to maintain the historically observed co-visibility relationships between objects. Building on this, we design an algorithm that uses the OCG to determine whether the current camera is within the observable region of each object, and subsequently implement an object state updating algorithm to detect and update outdated sections continuously. We conduct experiments on our self-make dataset with changing objects and a dataset with only static objects. The experimental results show that our method updates the outdated parts of the map more effectively compared to previous studies.**

I. Introduction

Visual SLAM is a technology for robots to build maps of the surrounding environment and localize themselves, and has been widely utilized in fields such as robotic navigation [1]. However, although the maps constructed by traditional Visual SLAM are rich in geometric information, they lack object-level information about the environment, which restricts the robot's performance in more advanced applications such as intelligent obstacle avoidance, human-robot interaction, and object recognition.

Building upon this, recent researches have focused on developing Object SLAM methods capable of object-oriented localization and map construction approaches, in which objects are typically represented by lightweight models such as cubes and quadrics [2], [3] or more resource-intensive representations like truncated signed distance functions(TSDFs) [4]. Some methods [5], [6] concentrate on improving the accuracy of object map generation or leveraging the generated object maps to enhance system relocalization performance [7], [8] both in static scene, while some other researchers aim to handle dynamic scenarios [9]–[11], focusing on tracking currently moving objects or using semantic segmentation to filter out potentially dynamic objects, thereby mitigating their impact on localization and mapping. However, these methods overlook those semi-static objects whose positions change while unobserved, such as chairs in a room or cars in a parking lot, leading to parts of the constructed object map becoming outdated. Since these methods lack a process for comparing current frame observations with the existing map, accurately identifying these objects with changing positions still remains a challenge for current object-oriented mapping approaches, causing the decrease of the localization and mapping accuracy in scenarios with semi-static objects.

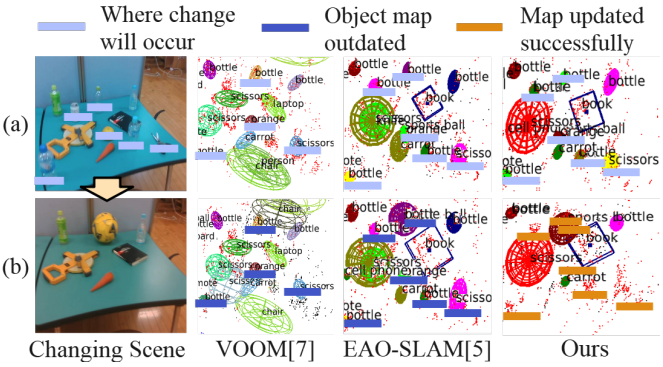


Fig. 1. Comparison between VOOM, EAO-SLAM and our method on our self-make dataset with a changing scene. (a) Before change. (b) After change. The first column illustrates a semi-static scene where object positions change while unobserved, highlighted in gray. The subsequent columns present the results of each method. When the changes is observed, EAO and VOOM fail to mitigate the impact of semi-static objects, as marked in blue, while our method effectively eliminates the influence, marked in orange.

In this paper, we propose a method to address the problem of updating outdated object maps in scenarios with semi-static objects, as shown in Fig. 1. Inspired by the use of the Covisibility Graph for loop detection in [12], we propose a new graph structure called the Object Covisibility Graph (OCG) to store the covisibility relationships between objects observed in past frames. Note that although similar concepts are proposed in [7], their essence and objectives are entirely different. The OCG is then employed to filter out all the objects in the map for which the current camera pose lies within their observable regions. i.e., the objects that should be potentially visible from the current camera pose. Furthermore, we design an algorithm that gradually identify and filter out outdated objects by comparing current frame observation with the objects in the existing map that should be potentially visible from the current camera pose. Finally, we conducted experiments on the TUM-RGB-D dataset [13] and a self-make semi-static dataset, demonstrating that our

This work was supported by JSPS KAKENHI Grant Numbers 18K04041 and 21K12071.

[1]Graduate School of Comprehensive Human Sciences, University of Tsukuba, Japan s2430515@u.tsukuba.ac.jp

[2]Institute of Library, Information and Media Science, University of Tsukuba, Japan

method is able to effectively update redundant or outdated objects while preserving the non-expired parts of the map in both semi-static and static environments.

In summary, our contributions are as follows:

- We propose a new data structure called the Object Covisibility Graph, which stores the covisibility relationship between objects in the scene and can be used to judge whether the current camera position is in the object's observable region.
- Based on Object Covisibility Graph, we propose an algorithm to solve the problem of updating outdated map in scenarios with semi-static objects.
- Experiment results on both TUM-RGB-D dataset and our self-make semi-static dataset demonstrate our method's ability to effectively update redundant or outdated objects while preserving the non-expired parts of the map in both semi-static and static environments.

II. Related Work

A. Object-oriented SLAM

With the advancement of robotics technology, there has been an increasing demand for mapping methods capable of constructing object-level maps. Some methods [14] employed semantic segmentation to detect 2D objects, attaching the semantic labels to 3d map points to create maps enriched with semantic information. However, since semantic information is only attached to map points, these methods struggle to achieve object-level understanding of the scene.

To represent the map at the object level, SLAM++ [15] firstly proposed the use of CAD models to build a 3D object-oriented map. However, this method relies on a pre-prepared database of object models, leading to limited generalization and constrained object variety. To address this issue, general object representation models have been proposed, including lightweight models using cubes [2] and quadrics [3], as well as more resource-intensive models like TSDFs [4]. Among these approaches, some focus on integrating object residuals into the bundle adjustment to boost localization precision [16], others aim at using object information to improve loop closure and relocalization performance [7], [8], while some prioritize constructing high-precision, low-redundancy object maps [5] [6]. However, these methods are all based on the assumption of a static environment, which makes them incapable of handling dynamic world challenges. On the other hand, some methods concentrate on eliminating the influence of current dynamic objects on localization [9]–[11]. Nevertheless, due to the lack of a process for comparing current frame observations with the existing object map, the aforementioned methods are unable to address the issue of outdated map caused by semi-static objects. In this work, we propose a method that is able to detect semi-static objects and update the map accordingly while also maintaining the accuracy of static object mapping.

B. Semantic Mapping in Semi-Static Environments

Semi-static objects, which are typically stationary but may change position over time, could lead to the outdated parts of existing object maps, thereby affecting localization accuracy and robotic applications. As some studies have made effort on mitigating the impact of semi-static objects on localization accuracy [17], [18]. these methods lack the ability to reflect the effects of semi-static objects onto object-level maps, leaving tasks such as navigation and decision making still being disrupted. [19] proposed a method that uses instance segmentation to classify and layer static backgrounds, dynamic objects, and semi-static objects into a dense point map, and successfully mitigated the impact of semi-static objects within the map. However, its reliance on dense semantic maps results in significant real-time performance limitations. Similarly, voxel-based methods [20], [21] have been introduced, using Bayesian estimation or weighted voxel differences to identify and remove outdated parts of the map, thereby addressing the semi-static issue. Nevertheless, voxel-based approaches continue to struggle with excessive processing time and overly large map sizes. [22] proposed a lightweight mapping method that employs an outlier detection algorithm based on the distance between object centroids. However, in real-world scenarios, determining the appropriate threshold for centroid distance to identify outliers remains challenging due to the uncertainty in scale. In this paper, we use a lightweight object map presentation and propose a generalizable algorithm to address the changing detection problem in semi-static environments.

III. Method

A. System Overview

Fig. 2 illustrates the overview of our system. Overall, our system takes each frame's RGB image and the object detection results as input. First, we use EAO-SLAM [5] to build an object map, with outdated semi-static objects undetected. Then, we store the covisibility relationship of objects in the scene up to the current frame using object covisibility graph(OCG), introduced in Sec. III-B. Subsequently, we design a change detection algorithm that performs observable region judgement and object state updating based on the OCG and current camera pose, as described in Sec. III-C, D, E. Finally, an outdated-parts-updated object map is obtained.

B. Object Covisiblility Graph

Fig. 3 shows an example of an OCG in operation. The OCG is an undirected graph where each node represents an object in the object map. An edge between two nodes indicates that the objects corresponding to these nodes can be observed simultaneously in the same frame, and the value of the edge represents the number of times they have been observed together.

The OCG is updated at each keyframe based on the objects observed in the current frame. This process involves adding newly observed nodes and edges to the graph, updating the value of edges between currently observable nodes, and removing those nodes that have been confirmed as outliers. Note that since this work does not involve improvements in visual odometry, the keyframe selection strategy is adopted from previous works [5], [12].

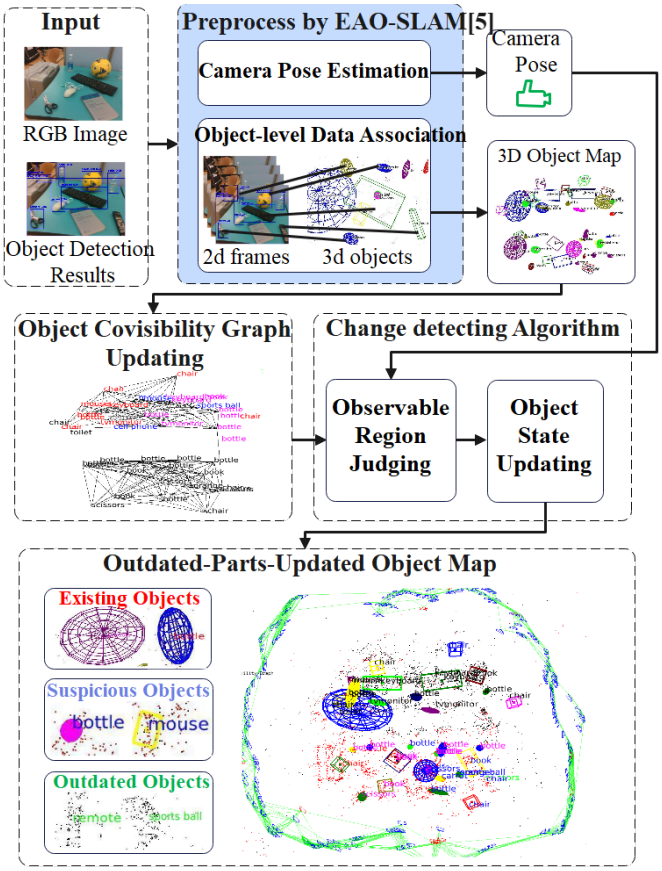


Fig. 2. Overview of the system.

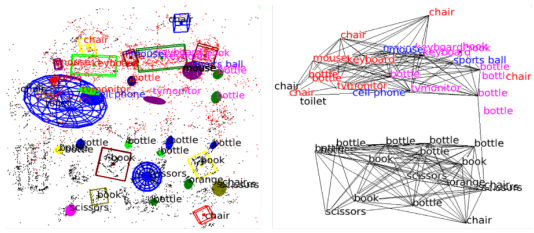
C. Object's Observable Region

We posit that for each object O_i, there exists a set of camera poses such that the field of view of these camera poses encompasses O_i or the region surrounding O_i. We refer to this camera poses set as **the object's observable region**. An example of the observable region is illustrated in Fig. 4(a).

As discussed in Section I, a key aspect of our idea for determining whether an object is outdated is to assess whether the current camera pose falls within the observable region of a certain object. In practical scenarios, this judgement cannot solely rely on the camera's orientation. Even if the camera is directly facing the object, factors such as blocking or viewing distance could cause an object to be undetectable by the object detection algorithms, resulting the object to be unobservable in the current frame. In this paper, rather than directly computing the observable region, we leverage the concept of first-level adjacent nodes in the OCG to determine whether the current frame has entered or exited the observable region, thus inferring the object's state.

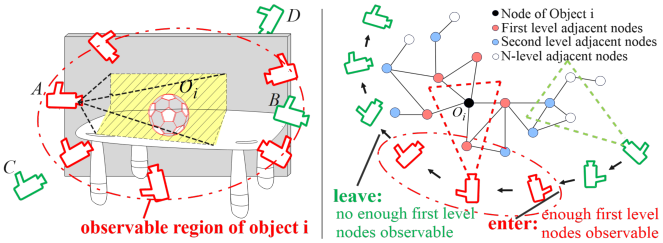
D. Judging Observable Region by first-level adjacent in OCG

In an undirected graph, "two nodes are n-level adjacent" means that the shortest path between the two nodes requires passing through n edges. In the OCG, edges represent co-visibility relationships between nodes. For a pair of n-level adjacent nodes, the larger the value of n, the less likely it is that the two nodes can be seen in the same frame, indicating that the nodes are relatively farther apart in the real world.



(a) 3D Object Map (b) Object Covisibility Graph

Fig. 3. Example of 3d object map and and Object Covisibility Graph(OCG). Each node in OCG represents an object within the object map.



(a) (b)

Fig. 4. (a) Example of the observable region of an object O_i. Camera pose A is in the observable region, while B, C, D are outside. B: Near enough to the O_i, but the orientation is excessively misaligned. C: Too far for the object recognition algorithm to identify the O_i. D: Blocked by an obstacle. (b) The relationship between the current observable condition of first-level adjacent nodes in the OCG with judging whether the current camera pose has entered of left the observable region of O_i.

Specifically, first-level adjacent nodes in the OCG have significant meaning. For two nodes V_i and V_j, if V_j is a first-level adjacent node to V_i (i.e., there exists an edge $E(V_i, V_j)$ between them), it indicates that these two nodes can be observed in the same frame, that compared to other nodes, V_j is relatively closer to V_i in the actual environment.

Based on this feature, for each object i in the map, we determine whether the current frame lies within its observable region by assessing whether a sufficient number of its first level adjacent nodes are currently observable, as illustrated in Fig 4(b). Specifically, we use the $ActiveEdgeRate$ to represent the sufficiency of the currently observable first-level adjacent nodes. Its calculation is shown in the following (1), where $E(V_i, V_j)$ represents the edge value between node V_i and V_j, $\mathcal{F}(V_i)$ denotes the set of currently observable first-level adjacent nodes of V_i, and $\mathcal{N}(V_i)$ represents the set of all first-level adjacent nodes of V_i.

$$ActiveEdgeRate(V_i) = \frac{\sum_{V_j \in \mathcal{F}(V_i)} E(V_i, V_j)}{\sum_{V_j \in \mathcal{N}(V_i)} E(V_i, V_j)} \quad (1)$$

Additionally, $\tau_{observable}$ is defined as the threshold for the $ActiveEdgeRate$. When the object is currently observed, the current frame pose is considered to be within the object's observable region. If the $ActiveEdgeRate$ is less than $\tau_{observable}$, we infer that the current frame fails to observe a sufficient number of first-level adjacent nodes, and the camera has left the object's observable region. Conversely, if the $ActiveEdgeRate$ exceeds $\tau_{observable}$, we infer that the camera has entered the observable region.

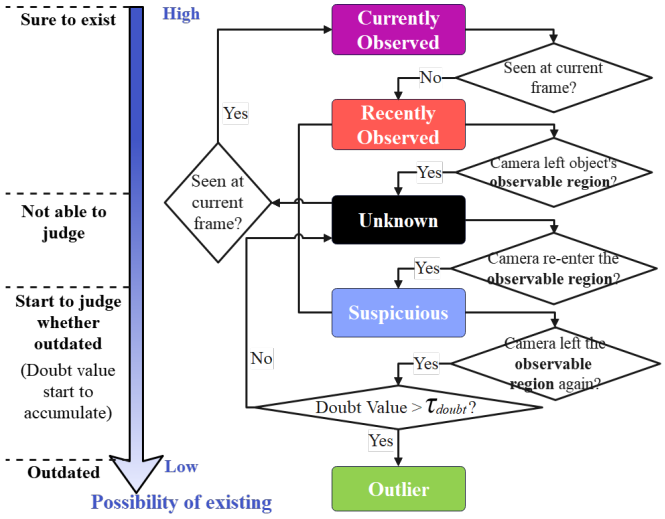


Fig. 5. State updating algorithm for each object in the map.

E. Object States and Object State Updating Algorithm

From sure to exist to sure to disappear, we classify the states of all objects in the map into five levels based on the likelihood that the objects still remain in their original positions within the scene:

- State 1. **Observed in the current frame**.
- State 2. **Recently observed**: Not observed in the current frame, but the camera has not yet left the object's observable region.
- State 3. **Unknown**: The camera has left the object's observable region.
- State 4. **Suspicious**: Recently and currently not observed, and the camera has re-entered the object's observable region. Begin to accumulate $DoubtValue$.
- State 5. **Outlier**: After sufficient observation, the object is confirmed to no longer be at its original position and is deleted from the map.

Based on the judgement on whether the current camera has left or entered an object's observable region, we design an algorithm for object state updating as shown in Fig. 5. Specifically, from the moment an object is observed in the current frame until the camera moves away from the object's observable region, we define the object's state as recently observed, i.e., states 1 and 2. When the camera has left the object's observable region, the object is in state 3, meaning the object is currently impossible to be observable. When the camera re-enters the potential observable region of the object, it is in state 4, Suspicious, indicating that the process of checking whether the object still exists has begun. In this state, We define $DoubtValue$ to represent the reliability of the inference that an object has disappeared from its original position. Specifically, the accumulation process of $DoubtValue$ is described in Equ. (2), where P is defined as a condition variable. $P = 1$ denotes that the projection of the object's centroid on the pixel plane falls within the image boundaries, and $P = 0$ denotes that it is not.

$$DoubtValue_{\text{new}} = \begin{cases} DoubtValue_{\text{old}} + 30, & ActiveEdgeRate = 1 \text{ and } P = 1 \\ DoubtValue_{\text{old}} + 20, & ActiveEdgeRate \in [0.5, 1) \text{ and } P = 1 \\ DoubtValue_{\text{old}} + 10, & ActiveEdgeRate \in (0.2, 0.5) \text{ and } P = 1 \\ DoubtValue_{\text{old}}, & P = 0 \end{cases} \quad (2)$$

Fig. 6. Self-make dataset with changing scenes. The camera moves around a table with a central partition while progressively capturing the scene that (a) Before changing and (b) After changing. All changing processes are unobserved.

The $DoubtValue$ is ultimately assessed when the camera exits the object's observable region again. τ_{doubt} is defined as the threshold for the $DoubtValue$. If the $DoubtValue$ exceeds the τ_{doubt}, it indicates that after sufficient observation, the object is still not observed, then it is removed as an outlier, i.e., state 5. Conversely, If the $DoubtValue$ fails to exceed the τ_{doubt}, the object reverts to state 3, Unknown. In above states, if the object is observed in the current frame, its state is directed transitioned to state 1.

Occasional failures in object detection could impact the robustness of the object state updating algorithm, as our method relies on object detection to determine whether an object is seen in the current frame. The thresholds $\tau_{observable}$ and τ_{doubt} are pivotal in regulating the method's tolerance to such failures. These thresholds can be adjusted, with higher values providing greater tolerance for detection inaccuracies.

IV. Experiment

A. Overview

In this chapter, we present the experimental results of our method. We implemented our system based on EAO-SLAM [5], a robust object data association and camera pose estimation system. The system input consists of an RGB image stream and offline object detection results for each frame by YOLOv8 [23]. The thresholds $\tau_{observable}$ and τ_{doubt} are heuristically set to 0.2, 50.

To demonstrate our method's ability to update outdated maps caused by semi-static objects and due to the lack of available semi-static scene datasets, we conducted experiments on a self-make semi-static dataset, which primarily focuses on capturing objects placed on tables similar to the TUM RGB-D dataset, but also includes semi-static scene changes. The dataset contains four sequences, all captured in the same scene but with varying object distributions and camera trajectories. Fig. 6 shows examples of our self-make dataset, where the camera moves around to capture objects on the table, and the positions of several objects change over time. All changing processes are unobserved.

Besides, to demonstrate that our method can also perform well in static environments, we also conduct experiments on TUM RGB-D dataset [13].

B. Metrics

In this paper, we primarily evaluate our method through two aspects: localization accuracy and object-level data as-

TABLE I

OBJECT-LEVEL DATA ASSOCIATION COMPARISON RESULTS ON SELF-MAKE SEMI-STATIC DATASET AND TUM RGB-D DATASET [13]. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**.

Dataset	Method	Corr. S	Unident. S	Redun. S	Corr. M	Delet. S	Unident. M	*Acc.* ↑	*Prec.* ↑	*Recall* ↑
Self-Make Dataset with Semi-Static Objects										
Self_straight_1	EAO-SLAM [5]	21	0	1	0	0	7	0.724	0.724	1
	VOOM [7]	19	2	3	0	0	7	0.613	0.655	0.905
	Ours	**21**	**0**	**1**	**7**	**0**	**0**	**0.955**	**0.933**	**1**
	GT	21	0	0	7	0	0	–	–	–
Self_straight_2	EAO-SLAM [5]	23	0	2	0	**0**	10	0.657	0.657	**1**
	VOOM [7]	20	3	2	0	0	10	0.571	0.625	0.870
	Ours	**23**	**0**	**1**	**9**	1	**1**	**0.885**	**0.875**	0.933
	GT	23	0	0	10	0	0	–	–	–
Self_round_1	EAO-SLAM [5]	34	0	15	0	**0**	6	0.618	0.618	**1**
	VOOM [7]	30	4	21	0	0	6	0.492	0.526	0.882
	Ours	**34**	**0**	**7**	**5**	1	**1**	**0.810**	**0.784**	0.967
	GT	34	0	0	6	0	0	–	–	–
Self_round_2	EAO-SLAM [5]	34	0	3	0	0	10	0.723	0.723	1
	VOOM [7]	32	2	18	0	0	10	0.516	0.533	0.941
	Ours	**34**	**0**	**2**	**9**	**0**	**1**	**0.919**	**0.893**	**1**
	GT	34	0	0	10	0	0	–	–	–
TUM RGB-D Dataset [13] with Only Static Objects										
Tum_fr1_desk_1	EAO-SLAM [5]	15	2	2	0	0	0	0.789	0.882	0.882
	Ours	**15**	**2**	**2**	**0**	**0**	**0**	**0.789**	**0.882**	**0.882**
	GT	17	0	0	0	0	0	–	–	–
Tum_fr2_desk	EAO-SLAM [5]	22	3	10	0	**0**	0	0.629	0.688	**0.880**
	Ours	**22**	**3**	**2**	**0**	5	**0**	**0.688**	**0.917**	0.733
	GT	25	0	0	0	0	0	–	–	–

TABLE II

COMPARISON OF ATE RMSE(M) ON TUM RGB-D DATASET. EACH SEQUENCE IS RUN 9 TIMES, AND THE MEDIAN RESULT IS RECORDED.

Dataset	ORB-SLAM2-mono [12]	EAO-SLAM [5]	Ours
Tum_fr1_desk1	0.157	0.074	0.065
Tum_fr1_desk2	0.667	0.735	0.578
Tum_fr2_desk	0.880	0.898	0.876
Tum_fr3_office	1.230	1.254	1.251

sociation performance. Object-level data association performance refers to the precision of associating objects in 2D frames with objects in the 3D object map, and is typically used to assess the ability of object SLAM systems to correctly generate object maps.

The localization accuracy is evaluated by the metric of the root mean square error (RMSE) of the absolute translation error (ATE), (i.e. ATE RMSE(m)). For evaluating object-level association performance, we use the metric of the number of constructed objects. Unlike many previous works like EAO-SLAM [5] and VOOM [7], we not only evaluate the number of generated static objects but also assess the ability to detect and remove semi-static objects. Specifically, we use the following six metrics to evaluate our method. Including correctly generated static objects (**Corr. S**), unidentified static objects (**Unident. S**), redundant static objects (**Redun. S**), correct deleted semi-static objects (**Corr. M**), incorrectly deleted static objects (**Delet. S**), unidentified semi-static objects (**Unident. M**). All of those metrics represent the number of corresponding objects, where the first three metrics represent the ability of generating static objects, and the last three metrics reflects the ability of detecting semi-static objects. All of these metrics are manually recorded. The closer these values are to the ground truth (GT), the higher the system's accuracy in object data association.

Additionally, we use accuracy (**Acc.**), precision (**Prec.**), and recall (**Recall**) to intuitively evaluate the object data association performance, where **Acc.** represents overall accuracy, **Prec.** indicates the ability to remove redundant objects, and **Recall** reflects the ability of avoiding incorrect deletions.

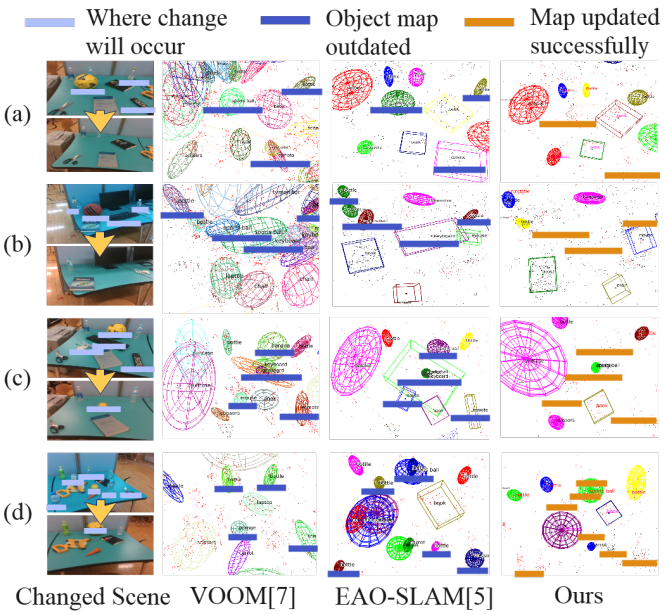


Fig. 7. Visual comparison between VOOM, EAO-SLAM and our method on our self-make dataset with changing scenes. The changing parts have been marked, same as Fig. 1.

C. Object-level Data Association Results

Table I shows the object-level data association comparison results of our method with EAO-SLAM [5] and VOOM [7] on our self-make semi-static dataset and the TUM RGB-D dataset. Experimental results on the self-make semi-static dataset demonstrate that our method maintains the same performance in generating static objects as [5] while significantly outperforming [5] and [7] in correctly identifying and removing semi-static objects that cause outdated parts of the

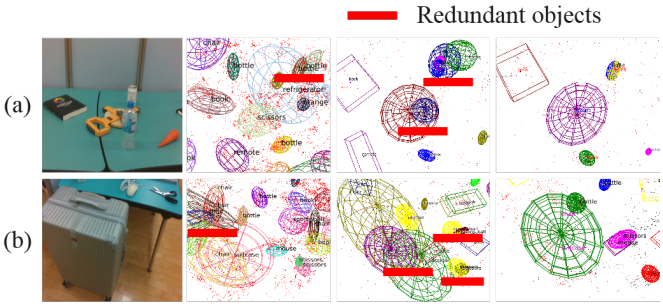


Fig. 8. Example of redundant maps. The redundant parts are marked in red.

map. Fig. 7 illustrates the visual comparison between our method and previous researches on the self-make semi-static dataset. Previous methods struggle to detect and eliminate changing parts in semi-static scenes, while our method effectively updates outdated object maps while preserving the non-expired portions of the map.

Additionally, due to object detection occasionally producing different recognition results for the same object, previous research sometimes generates redundant objects for a single object, which impacts the accuracy of data association. Since these redundant objects are usually not updated after being generated, our method can identify them as outdated semi-static objects and remove them. Experimental results in both self-make semi-static dataset and TUM RGB-D dataset show that compared to previous research, our method is capable of removing redundant objects while minimizing the risk of mistakenly deleting static objects. Fig. 8 demonstrates the effectiveness of our method in removing redundant objects from the map. For the redundant objects that appear in [5] and [7], our method can delete them while retaining the actual existing 3D objects.

However, results on both datasets indicate that our method could carry a potential risk of incorrectly deleting actual existing static objects. One possible reason is that although the camera enters the observable range of the object and and actually sees the object, the object detection method fails to recognize the object in the current frame occasionally, leading the system to incorrectly assume that the object is not observable within its observable region, and thus deletes the object mistakenly.

D. Localization Results

Although this paper does not focus on improving localization accuracy, we conducted localization experiments on the TUM dataset to demonstrate that our method does not negatively impact localization performance. Table II presents our localization results. Since both the EAO-SLAM [5] and our approach are based on the mono mode of ORB-SLAM2, we compare the localization performance of ORB-SLAM2-mono, EAO-SLAM, and our method. Although the results on fr1-desk1 show slightly larger deviations, which may be attributed to the small size of the scene and the scale uncertainty in monocular initialization, the overall experimental result indicate that our approach achieves a similar localization performance compared to ORB-SLAM2-mono.

V. Conclusion

In this paper, we present a method for updating outdated parts of object maps due to semi-static objects. First, we developed Object Covisibility Graph(OCG) to store the covisibility relationship between objects in the map. Next, based on OCG, we propose algorithms to judge whether the current camera pose is within the observable region of an object and detect outdated objects in the map. Experiment results show that our method effectively update redundant or out-dated objects in the map while preserving the non-expired parts of the map in both semi-static and static environments. The future work of this paper includes: 1. Tracking and mapping for dynamic objects. 2. Loop closing and relocalization in semi-static scenarios.

References

[1] S. Yang, S. A. Scherer, X. Yi, and A. Zell. Multi-camera visual slam for autonomous navigation of micro aerial vehicles. *RAS*, 2017.
[2] S. Yang and S. Scherer. Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, 35(4):925–938, 2019.
[3] L. Nicholson, M. Milford, and N. Sünderhauf. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *RAL*, 4(1):1–8, 2018.
[4] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric object-level slam. In *3DV*, 2018.
[5] Y. Wu, Y. Zhang, D. Zhu, Y. Feng, S. Coleman, and D. Kerr. Eao-slam: Monocular semi-dense object slam based on ensemble data association. In *IROS*, 2020.
[6] 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. *RAL*, 7(2), 2022.
[7] Y. Wang, C. Jiang, and X. Chen. Voom: Robust visual object odometry and mapping using hierarchical landmarks. In *ICRA*, 2024.
[8] M. Zins, G. Simon, and M. Berger. Oa-slam: Leveraging objects for camera relocalization in visual slam. In *ISMAR*, 2022.
[9] B. Bescos, C. Campos, J. D. Tardós, and J. Neira. Dynaslam ii: Tightly-coupled multi-object tracking and slam. *RAL*, 2021.
[10] M. Strecke and J. Stuckler. Em-fusion: Dynamic object-level slam with probabilistic data association. In *ICCV*, 2019.
[11] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic slam. In *ICRA*, 2019.
[12] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *TR*, 2017.
[13] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, 2012.
[14] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *IROS*, 2017.
[15] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *CVPR*, 2013.
[16] Z. Liao, W. Wang, X. Qi, and X. Zhang. Rgb-d object slam using quadrics for indoor environments. *Sensors*, 2020.
[17] D. M. Rosen, J. Mason, and J. J. Leonard. Towards lifelong feature-based mapping in semi-static environments. In *ICRA*, 2016.
[18] S. Zhu, X. Zhang, S. Guo, J. Li, and H. Liu. Lifelong localization in semi-dynamic environment. In *ICRA*, 2021.
[19] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger. Robust dense mapping for large-scale dynamic environments. In *ICRA*, 2018.
[20] J. Qian, V. Chatrath, J. Yang, J. Servos, A. P. Schoellig, and S. L. Waslander. Pocd: Probabilistic object-level change detection and volumetric mapping in semi-static scenes. In *RSS*, 2022.
[21] L. Schmid, J. Delmerico, J. L. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena. Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. In *ICRA*, 2022.
[22] C. Gomez, A. C. Hernandez, E. Derner, R. Barber, and R. Babuška. Object-based pose graph for dynamic indoor environments. *RAL*, 2020.
[23] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics YOLO, January 2023.