A Benchmark Synthetic Dataset for C-SLAM in Service Environments

Harin Park, Inha Lee, Minje Kim, Hyungyu Park and Kyungdon Joo*
Artificial Intelligence Graduate School, UNIST
{harinp33,epsilon8854,minje617,hyungyu,kyungdon}@unist.ac.kr

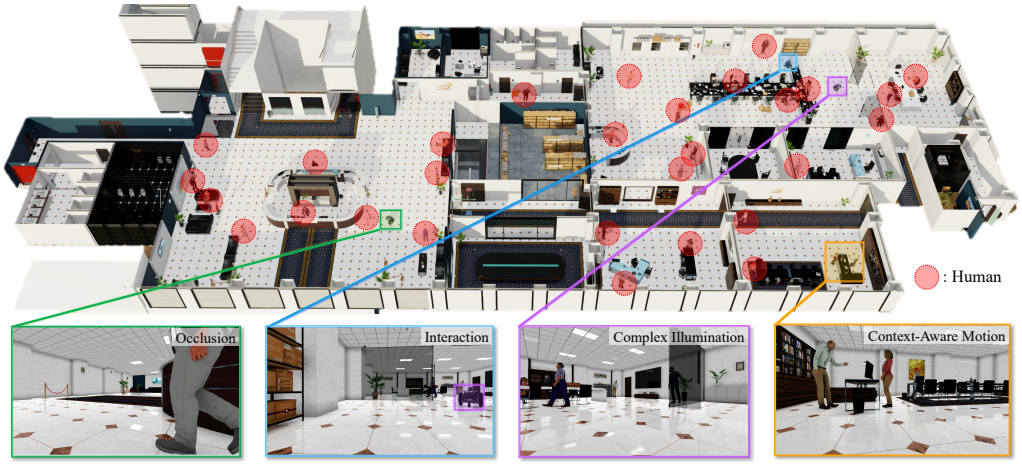


Figure 1. **Illustration of the proposed CSE dataset in *Office* environments**. Our CSE dataset contains realistic service robot environments with dynamic human objects, indicated by a red circle. In particular, our dataset includes challenging observations that are commonly encountered in service environments, such as occlusion and complex illumination. In addition, our dataset provides realistic and challenging scenarios, such as interaction between robots and context-aware human motions.

Abstract

*In this work, we introduce a new multi-modal C-SLAM dataset for multiple service robots in various indoor service environments, called C-SLAM dataset in Service Environments (CSE). We use the NVIDIA Isaac Sim to build data in various indoor service environments with the challenges that may occur in real-world service environments. By using simulation, we can provide accurate and precisely time-synchronized sensor data, such as stereo RGB, stereo depth, IMU, and GT poses. We configure three common indoor service environments (*Hospital, Office, *and* Warehouse*), each of which includes multiple dynamic objects that perform motions suitable to each environment. In addition, we navigates the robots to mimic the actions of real service robots. Through these factors, we generate a more realistic C-SLAM dataset for multiple service robots. We demonstrate our dataset by evaluating diverse state-of-the-art multi-robot SLAM methods.*

*Corresponding author.

1. Introduction

Recently, intelligent agents, such as personal robots and autonomous vehicles, have pervaded our lives and become indispensable. In particular, service robots have started to replace simple yet human resources-required tasks, such as serving, path guidance, cleaning, delivery, *etc*. [16, 17, 26]. To this end, service robots are required to understand unknown environments [17, 21], where simultaneous localization and mapping (SLAM), which estimates the pose of the robot itself and builds a map of an unknown environment simultaneously, is one of the most fundamental techniques for service robots [20]. Service robots are capable of perceiving their surroundings and performing their tasks using SLAM.

Specifically, service environments, where service robots operate and may interact with people [15], have become diverse (*e.g*., from static spaces to complex indoor or outdoor environments) [4] and have begun to require more complex tasks that are difficult for a single agent to handle [18]. This change in service environments naturally has led to an interest in multiple agents from single agents. Furthermore, SLAM algorithms, the

Table 1. **Comparison of existing C-SLAM datasets for multi-robot.** Time synchronization columns are referenced by [8]

Dataset	Sensors				Static (S) / Dynamic (D)	Platforms	Environment	Ground Truth Pose	Time sync	
	RGB	Depth	IMU	LiDAR					Intra	Inter
UTIAS [14]	✓†				S	UGV	Indoor	Motion capture	Sw	NTP
AirMuseum [7]	✓†		✓		S	UGV, UAV	Indoor	SfM	Sw	NTP
Ford-AV [5]	✓		✓	✓	D	Vehicle	Outdoor	GPS-IMU, SLAM corrected	–	GNSS
S3E [8]	✓		✓	✓	S	UGV	Outdoor, Indoor	RTK, Motion capture	Hw	GNSS, PTPv2
GRACO [27]	✓		✓	✓	D	UGV, UAV	Outdoor	GNSS / INS	Hw	GNSS
Tian *et al.* [23]	✓	✓	✓	✓	D	UGV, UAV	Outdoor, Indoor	Point-cloud by LiDAR, GPS	–	NTP
Ours	✓	✓	✓	✓*	S + D	UGV	Service Env. (Indoor)	Simulator (NVIDIA Isaac Sim)	Simulator	Simulator

† Only monocular RGB modality.
* We provide a function that converts depth images to pseudo-LiDAR data.

basis of robot perception, have also begun promoting performance improvement through collaboration between multiple agents. Accordingly, a new SLAM task, *Collaborative SLAM* (*C-SLAM* in short) for multiple agents, has been developed in the robotics community [9, 10, 22, 24] and aims to improve the robustness, efficiency, and accuracy of localization and mapping by leveraging the interchange of spatial information across multiple agents [16, 28].

Despite this progress, C-SLAM remains limited in terms of benchmarking [13]. While standardized benchmarks for single robot SLAM have emerged extensively, systematic evaluation techniques and datasets for C-SLAM are still lacking [13]. For example, the OpenLORIS-Scene dataset [20], as a SLAM dataset for a single service robot, encompasses various challenges encountered in service environments, such as textureless scenes, dynamic objects, and viewpoint changes. However, early datasets for C-SLAM [7, 14] are acquired only in static and indoor experimental environments, excluding dynamic objects. To alleviate these limitations, a few datasets that include dynamic objects, such as humans or vehicles, have recently been proposed [5, 8, 23, 27], but they are mainly acquired from urban outdoor scenes or limited indoor environments (*e.g.*, only a small corridor or a room-size laboratory). In other words, there is still a lack of diversity in service environments for multi-robot in terms of benchmark datasets (see Table 1).

In the case of service robots, they operate for long periods of time in various indoor environments, where they have diverse interactions. Concretely, each service robot can move through complex indoor spaces, collaborate with others, or interact with people in service environments, such as hospital, restaurant and office [11, 16, 17]. Notably, while multiple service robots are in operation, challenging scenarios arise for performing C-SLAM. For example, they may encounter homogeneous scenes or severe occlusions/sudden large rotations caused by dynamic objects. Motivated by this fact, in this work, we introduce a new multi-modal C-SLAM dataset for multiple service robots in indoor service environments, called `CSE` dataset; especially, our `CSE` dataset includes various indoor service environments, such as hospital, office and warehouse (see *Office* in Fig. 1), and mimics diverse and challenging scenarios for service robots.

In constructing our dataset, we consider several essential factors that must be satisfied as a C-SLAM dataset for service robots. 1) Each robot must provide time-synchronized and abundant sensor modalities that can allow us to demonstrate SLAM for different sensor combinations. 2) Multi-robot should explore diverse paths in various service environments, and precise and time-synchronized ground truth (GT) poses for each robot are essential for evaluating C-SLAM. It should be noted that acquiring accurate time-synchronized sensor data and GT poses among multi-robot is non-trivial in the real world. 3) Multi-robot must reproduce various scenarios that can occur in real service environments, such as avoiding dynamic objects during path planning and robot-human/robot-robot interactions at close distances. To satisfy the above factors, we propose a new synthetic C-SLAM dataset for multiple service robots using a simulator, NVIDIA Isaac Sim [2], which provides photo-realistic sensor data. Based on NVIDIA Isaac Sim, we acquire time-synchronized sensor data and accurate GT poses for multi-robot. We also configure various service environments with the challenges that arise in real-world service environments. Each environment is separated into static and dynamic, based on the on/off of dynamic objects. In particular, in the case of dynamic objects like humans, anonymity can be guaranteed by using simulation. In addition, we also build a realistic dataset by simulating the actions that occur when real service robots drive. The main characteristics of our dataset are:

- We propose the first synthetic C-SLAM dataset for multiple service robots, `CSE` dataset. Each robot includes stereo RGB, stereo depth, inertial measurement unit (IMU), and GT pose, and all data are precisely time-synchronized.
- We acquire data in diverse service environments (*hospital*, *warehouse*, *office*) where real service robots could operate. For each environment, we place the dynamic objects with suitable actions and include diverse challenging cases, such as homogeneous walls/floors and

redundant objects.

- We construct the scenarios considering intra/inter-robot loop closures to facilitate the appropriate evaluation of C-SLAM. In addition, we separate each environment into static and dynamic to acquire data in the same scenario. This allows us to evaluate the efficiencies of SLAM algorithms dealing with dynamic objects.
- We validate our dataset by evaluating diverse state-of-the-art multi-robot SLAM. We also analyze the impact and limitations of the various challenges in our dataset on SLAM algorithms.

2. Related work

The goal of C-SLAM algorithms is to enhance the efficiency and accuracy that surpass the capabilities of single-robot SLAM by integrating data from individual robots to form globally consistent maps and state estimates [13]. Due to their advantages, such as enabling mapping over large areas and facilitating efficient exploration and task execution, C-SLAM has gained significant attention in research. Nevertheless, there is still a shortage of benchmark datasets for evaluating and developing C-SLAM.

UTIAS [14] is the first dataset for multi-robot SLAM. UTIAS presents a 2D multi-robot SLAM dataset based on 15 distinct landmarks in a 15m $\times$ 8m indoor environment, using five robots equipped with a monocular camera. However, it is limited to a constrained indoor experimental space and offers only 2D GT poses. AirMuseum [7] and GRACO [27] are multi-robot SLAM datasets utilizing heterogeneous agent platforms. AirMuseum was acquired in an indoor environment with multiple ground robots and drones equipped with apriltag markers, providing GT trajectories through Structure from Motion (SfM). But it only assumes a static indoor environment. GRACO involves the use of ground robots and drones in outdoor urban scenes. However, they do not include challenging cases, such as occlusions due to various dynamic objects, that robots may face in the real world. Additionally, S3E [8] proposes a long-term multi-modal dataset using multiple ground robots in both indoor and outdoor environments based on four well-designed trajectory paradigms. Tian *et al*. [23] also acquired data using eight ground robots, including dynamic objects (*e.g*., vehicles and pedestrians) and different lighting conditions in environments, such as urban outdoor scenes and indoor environments like tunnels and corridors.

Unlike most C-SLAM datasets that are mainly acquired in urban outdoor scenes or limited indoor environments, our `CSE` dataset covers various indoor environments specialized for service robots. Furthermore, existing C-SLAM datasets do not reflect the characteristics that occur when real robots move since humans control the robot manually. In contrast, we leverage the ROS Navigation Stack[1] for robot driving, which enable more realistic scenarios, allowing the robots to recognize their environment and avoid dynamic objects.

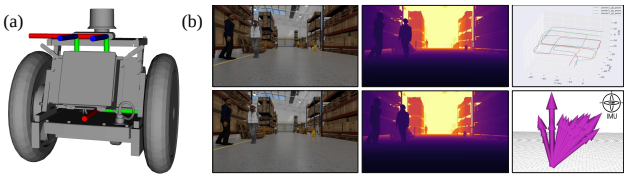


Figure 2. **Robot configuration and sensor data example**. (a) The NVIDIA Carter, our robot platform. (b) Examples of acquired sensor data (stereo RGB, stereo depth, GT poses and IMU).

3. C-SLAM Dataset in Service Environments

In this section, we present a new benchmark synthetic dataset for C-SLAM in service environments, `CSE` dataset. The proposed `CSE` dataset is basically built upon NVIDIA Isaac Sim [2] and NVIDIA Omniverse [3]. NVIDIA Isaac Sim is a robotics simulator powered by the Omniverse platform that provides photo-realistic and physically accurate virtual environments. Thus, it allows us to collect accurate GT poses and time-synchronized sensor data. The proposed `CSE` dataset is acquired in three indoor service environments with challenging cases, including serious occlusions by dynamic objects, homogeneous floors, and redundant objects, *etc*. We categorize each environment into static and dynamic, thereby providing a total of 18 sequences for SLAM (6 sequences for C-SLAM).

This section is organized as follows: Sec. 3.1 describes the robot platform and how to navigate the robot in the simulation. Sec. 3.2 provides the details about the sensor specifications, including types and properties of sensor modalities and time-synchronization method. Sec. 3.3 introduces how we construct the service environments used for data acquisition. Finally, Sec. 3.4 offers considerations and details about the scenarios for the robots in each environment.

3.1. Robot Configurations

As a robot platform, we utilize the NVIDIA Carter URDF[2] provided by Isaac Sim (see Fig. 2(a)). Carter[3] is a differentiable drive robot with two wheels on each side that is designed for verifying the capabilities of the Isaac SDK[4]. Isaac SDK is intended to develop applications for

[1] https://wiki.ros.org/navigation

[2] URDF (Unified Robot Description Format) is a standard data format for describing a robot's structure in XML that defines its components, size, shape, position, and joint information.

[3] https://docs.nvidia.com/isaac/archive/2020.2/doc/tutorials/carter_hardware.html

[4] https://docs.nvidia.com/isaac/archive/2021.1/doc/overview.html

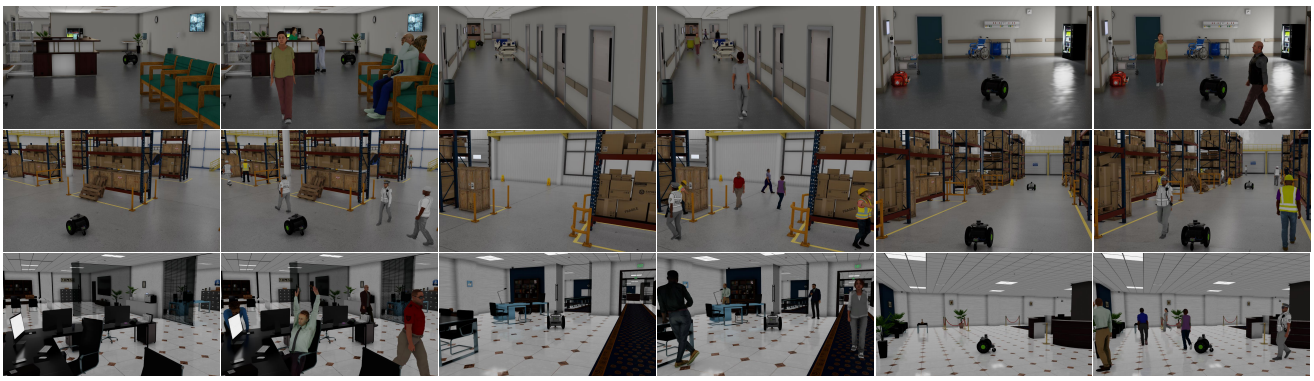


Figure 3. **Example of service environments in the proposed CSE dataset**. Each row shows the service environments we built (*Hospital*, *Warehouse*, and *Office* in order) from several viewpoints. Odd columns represent static environments, while even columns represent dynamic environments. In particular, we can observe dynamic objects having suitable actions and clothes for each environment.

complicated use cases, such as delivery robots, and the Carter is developed as a delivery robot. Accordingly, we choose Carter as our robot platform for building the C-SLAM dataset tailored for service robots. We deploy three Carters as service robots in the target environments. We operate this Carter using ROS Navigation Stack integrated within Isaac Sim. ROS Navigation Stack is a 2D navigation stack that generates a path to a target pose using odometry and sensor data. It is utilized in the real world for robot navigation, employing a global planner to plan the global path, and a local planner detects and avoids surrounding dynamic objects to navigate. By utilizing the ROS Navigation Stack, we can simulate behaviors that occur when real robots navigate. For example, robots may avoid when they encounter each other or wait for a while when their path is interrupted by dynamic objects.

It should be worth noticing that our approach reflects realistic scenarios where robots are aware of their surroundings and interact with dynamic objects. On the other hand, existing datasets [7, 8, 23, 27] are generated by humans controlling the robots manually. This approach does not reflect the actions shown by real robots as they perceive their surroundings and encounter unexpected dynamic objects.

3.2. Sensor Configurations

For each robot, we attach several types of sensors for perception in service environments, as shown in Fig. 2(b). Specifically, the sensor system of each robot is equipped with RGB and depth stereo cameras and an IMU sensor. The resolution of both stereo RGB and depth cameras is 1280 x 720, and the baseline length is set to 12cm, following the specification[5] of the camera on board Carter. We also provide camera parameters, such as intrinsic and extrinsic between sensors, as the left RGB camera as the reference. For IMU, we provide empirically tuned IMU parameters.

Using this sensor system, we can acquire stereo RGB images, stereo depth images, IMU measurements, and GT poses for each robot through robot operating system (ROS), as shown in Fig. 2(b). To follow sensor configurations in the real world, we set Isaac Sim's physics settings to 120 to set the IMU to 120Hz. However, in this setting, all sensor data are acquired at 120Hz. To handle this issue, we implement a simple ROS node that filters image topics published at 120Hz to 30Hz and ensures sensor synchronization. In addition, we provide a function that extracts 3D sparse point clouds (*i.e.*, pseudo-LiDAR measurements) from the depth images. Details of sensor specifications and ROS topics are available in the Appendix.

3.3. Service Environments

We select three common indoor service environments: *Hospital*, *Warehouse*, and *Office*, where actual service robots operate. Furthermore, we categorize each environment into static and dynamic, building a total of six environments. Figure 3 shows several examples of each environment, including static and dynamic environment. By categorizing each environment into static and dynamic, we expect the following effects from our dataset. 1) It is possible to build data that can cover challenging cases in the service environment itself (static) and external challenges that occur due to the addition of dynamic objects. 2) These categorizations provide an opportunity to evaluate the effectiveness of SLAM algorithms handling dynamic objects. 3) Moreover, the robot navigates with the same goal point in both static and dynamic environments, which makes the evaluation more valid.

Service Scenes. We utilize basic scenes provided by NVIDIA and modify them with additional construction for our purposes. In the case of *Hospital* and *Office*, we proceed with structural and textural modifications to reflect the challenging cases. In addition, we build

[5] https://docs.nvidia.com/isaac/archive/2020.2/doc/tutorials/carter_hardware.html

(a) *Hospital* (b) *Office* (c) *Warehouse*

Figure 4. **Challenging cases in the viewpoint of robots**. Each column shows the challenging cases that occur in each environment. In particular, it indicates severe occlusions due to dynamic objects.

realistic environments by manually placing various assets suitable for each environment. For *Warehouse*, we use the SceneBlox tool[6] of NVIDIA Omniverse Replicator to generate a basic scene. SceneBlox is a tool for creating large and consistent simulation scenes. Based on this tool, we generate the basic scene for *Warehouse*, and do the same post-processing as *Hospital* and *Office*.

Hospital, with 76m $\times$ 45m, mainly consists of narrow and long corridors. Specifically, the walls and floors of *Hospital* are homogeneous, making it hard to extract the visual information and re-localization difficult. For example, there are a series of doors and objects with the same design in the corridors. In other words, there are lots of redundant objects placed, which can create ambiguity in feature matching and significantly reduce pose estimation accuracy. *Office* with 31m $\times$ 94m comprises two large spaces connected by short corridors. It includes the various challenging cases since it has homogeneous walls and floors and ceilings with repetitive patterns. In addition, the floor is made up of reflective materials; reflective properties, especially, can cause another challenge by light reflections. *Warehouse* is a cuboid shape with 56m x 74m, where large grid structures are regularly arranged. Similar to *Hospital*, it consists of homogeneous walls and floors, as well as a large number of redundant doors and boxes, which can make feature matching challenging.

Dynamic Objects. All dynamic environments are built by placing dynamic objects based on each static environment. As dynamic objects, we utilize human assets provided by NVIDIA and purchased assets from ActorCore[7] suitable for each environment. Dynamic objects are evenly distributed spatially and perform suitable actions to each environment. For example, in the *Hospital* environment, there are doctors walking around the rooms or nurses talking at the reception desk. For *Office*, the office workers are sitting in chairs, conversing with each other, and answering phone calls. In addition, dynamic objects move around specific areas continuously. Through these movements, as illustrated in Fig. 4, severe occlusions can occur, obstructing the camera's view. We show the human asset figures for each environment used in our dataset in the Appendix.

[6] https://docs.omniverse.nvidia.com/isaacsim/latest/replicator_tutorials/tutorial_replicator_sceneblox.html

[7] https://actorcore.reallusion.com/

3.4. Scenarios

C-SLAM involves multiple robots collaborating to explore the environments and build a map. During this process, each robot visits the same location repeatedly (*i.e.*, intra-robot loop closure) and frequently encounters each other (*i.e.*, inter-robot loop closure). Such inter- and intra-robot loop closures are essential for the robots to estimate their pose and build the map accurately. Based on this fact, we design the scenarios considering inter- and intra-robot loop closures. In particular, for inter-robot loop closures, we categorize it into more detailed interaction types (Follow, Intersection, Overlap). Follow describes a situation where one robot follows behind another, and Intersect means where the robots are across each other. These occur when robots are in the same area at the same time. Finally, Overlap refers to a situation where they pass through the same region at different times in the same direction. Illustration for each type is available in the Appendix.

For each environment, the static and dynamic scenarios share the same goal points for operating each robot. However, since we use ROS Navigation Stack to navigate robots, the paths between goal points in static and dynamic scenarios do not match perfectly, but they are almost identical. The scenarios for each robot across the three environments are as shown in Fig. 5, and Table 2 mentions the characteristics and interaction types of the scenarios for each environment. Additionally, our dataset can be utilized by dividing each sequence into multiple sub-sequences. We acquired data over a long period of time, and the robot encounters a variety of challenges in different regions as it drives. Therefore, each sequence can be separated and utilized according to its intended purpose.

Hospital. The scenarios for *Hospital* are depicted in Fig. 5(a). All robots navigate from different starting points and return to them.

Static: All of the robot's scenarios consist of driving down long corridors and exploring small spaces. In common, they perform large rotations when exploring small spaces to avoid collision. These large rotations can quickly change the camera's view, making pose estimation difficult. ROBOT 1 and ROBOT 3 include the intra-robot loop closure when they arrive at the endpoints, but ROBOT 2 does not include it. However, for ROBOT 2, the scenario consists of

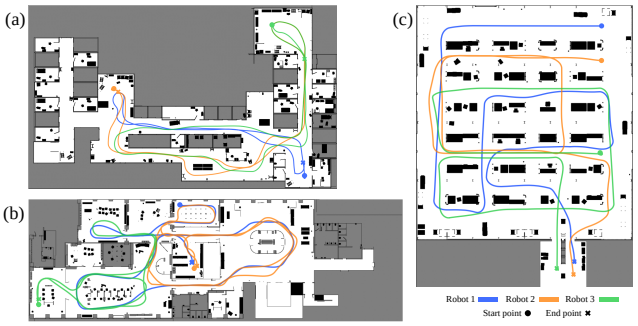


Figure 5. **Scenarios in the proposed CSE dataset**. We visualize scenarios for each dynamic environment on its 2D occupancy map with the same scale. Note that scenarios in this illustration only show dynamic environments.

exploring the overall area of the *Hospital*, which results in a lot of Overlap with both robots. Therefore, in the case of ROBOT 2, there are no intra-robot loop closures, but there are many inter-robot loop closures with other robots. In addition, there are scenarios where ROBOT 3 encounters the other robots (Intersect), and ROBOT 3 follows ROBOT 1, resulting in inter-robot loop closure.

Dynamic: In dynamic environments, humans avoid obstacles (i.e., robots) only when they observe them at close range. This leads to challenging cases where a large number of dynamic objects obstruct the robot's view. In particular, for ROBOT 2 and ROBOT 3, the robots capture the motion where dynamic objects avoid the robots at very close. This causes extreme occlusions in the robot's view, as shown in Fig. 4(a). This can lead to challenges where the robot makes wrong feature associations.

Office. Figure 5(b) is the scenarios for *Office*. Similar to scenarios in the *Hospital*, every robot has different starting points. ROBOT 2 and ROBOT 3 finish their navigation near their starting points, whereas ROBOT 1 ends at the different location from its starting point.

Static: ROBOT 2 and ROBOT 3 drive over a specific region repeatedly, and ROBOT 1 drives in the overall area of *Office*. In the case of ROBOT 2, it consists of complicated scenarios that involve driving through both rooms and corridors. This causes ROBOT 2 to make large rotations when going through the narrow doors. The scenarios for ROBOT 2 and ROBOT 3 have many intra-robot loop closures due to repeatedly driving over the same areas. However, in the case of ROBOT 1, there are no intra-robot loop closures, but has lots of Overlap that causes the inter-robot loop closures because it drives through the entire environment.

Dynamic: The dynamic *Office* environment has about twice as many humans as *Hospital*. As a result, all robots suffer from extreme occlusions, where their camera views are blocked by many humans, as shown in Fig. 4(b). Furthermore, due to dynamic objects moving around a specific area, each robot captures the same person in a different location. This can lead to challenges with incorrect inter-robot loop closures between robots.

Warehouse. Figure 5(c) shows the scenarios for *Warehouse*. All robots start from different starting points and converge in the same space to finish their navigation.

Static: In this environment, the robots drive between large, regularly aligned structures. The scenarios for ROBOT 2 and ROBOT 3 involve intra-robot loop closures; especially, ROBOT 2 contains many intra-robot loop closures since it repeatedly drives over the same regions. In contrast, ROBOT 1 contains no intra-robot loop closures. However, for the *warehouse* scenarios, all robots contain many inter-robot loop closures. For example, ROBOT 1 may drive back through the area that ROBOT 3 passed through (Overlap), and ROBOT 2 may drive behind ROBOT 1 (Follow). In particular, ROBOT 1 and ROBOT 2 intersect each other, which leads to the challenges of observing each other at close range, resulting in extreme occlusion. In addition, ROBOT 1 and ROBOT 2 overlap their driving paths in narrow sections, causing ROBOT 2 to perform a recovery behavior. Recovery behavior means that when the robot encounters unexpected obstacles and dynamic environment changes during the path planning process, the robot recognizes and solves the problem by itself so that it can drive again. This happens because we utilize the ROS Navigation Stack. In this environment, ROBOT 2 sees ROBOT 1 as a dynamic obstacle and stops, then resumes driving once it is sure that it is no longer obstructing its path. In this case, the robot can cause severe occlusions as a dynamic object. These scenarios can be seen as well reflecting the real-world situations.

Dynamic: Dynamic *Warehouse* environment also has lots of humans, similar to dynamic *Office* environments. As a result, all robots will encounter many humans at fairly close range. Through this, extreme occlusions occur very frequently in the robot's camera view.

4. Experiments

In this section, we perform validation on our dataset by evaluating diverse SLAM algorithms. We describe SLAM algorithms utilized to evaluate our proposed dataset. We also provide analysis of results on our dataset for each algorithm. Experimental details related to single-robot SLAM and visualization results of SLAM algorithms are available in Appendix.

4.1. Baseline

C-SLAM for Multi-Robot. COVINS [19] is a centralized visual-inertial SLAM system that utilizes data collected from multiple robots. This system gathers data generated by ORB-SLAM3 from numerous robots to perform global optimization, and it enhances joint estimation by

Table 2. **Dynamic scenario configurations.**

Environment	Robot	Duration (s)	Length (m)	Loop Closure				Characteristics / Challenges†	Size	# of dynamic objects*
				Intra	Inter					
					Follow	Intersection	Overlap			
Hospital	ROBOT 1	373.1	124.0	✓		✓	✓	Long corridors / Homogeneous floor	76m × 45m	3 + 16
	ROBOT 2	563.1	200.6			✓	✓			
	ROBOT 3	509.1	182.8	✓	✓	✓	✓			
Office	ROBOT 1	604.5	210.1		✓		✓	Complex space with rooms and corridors / Repetitive pattern (floor, ceiling), Reflective material floor	31m × 94m	3 + 37
	ROBOT 2	703.5	238.2	✓	✓		✓			
	ROBOT 3	508.5	175.9	✓			✓			
Warehouse	ROBOT 1	645.0	250.1		✓	✓	✓	Large regular grid structures / Homogeneous floor	56m × 74m	3 + 38
	ROBOT 2	643.1	254.7	✓		✓	✓			
	ROBOT 3	631.1	251.0	✓	✓	✓	✓			

† In common, service environments in our dataset include visual redundancy, dynamic objects, and homogeneous walls.

* The number of objects that include robots and humans.

incorporating place recognition and eliminating redundant data. Kimera-Multi [6] also is a SLAM system designed for multi-robot environments, where each robot communicates with others to understand the environment in real-time. It improves local estimations through inter-robot loop closures and constructs a globally consistent, real-time metric-semantic 3D mesh model. Swarm-SLAM [12] employs a decentralized approach, utilizing novel techniques for efficient communication between robots and rapid convergence. This framework is well-suited for large-scale deployment, and its effectiveness in terms of accuracy and resource utilization efficiency has been demonstrated through empirical experiments.

Swarm-SLAM-D is an algorithm we developed to assess the impact of dynamic objects on SLAM algorithms. This algorithm is designed to maintain localization performance in dynamic environments by incorporating the dynamic feature removal module of DS-SLAM [25] into Swarm-SLAM. The dynamic feature removal module of DS-SLAM identifies and eliminates specific feature points based on whether they are dynamic, using both previous and current frames as references. The implementation details of Swarm-SLAM-D is available in Appendix.

4.2. Experimental setup

To comprehensively evaluate various sensor modalities offered by our dataset, we conduct evaluations using RGB-D, stereo-inertial, and mono-inertial modalities provided by each baseline algorithm. The trajectories estimated from each baseline are measured for performance using the absolute trajectory error (ATE). The whole evaluation process is conducted with EVO [1] python library package.

For multi-robot SLAM algorithms, we adjust the play rate to $0.5\times$ speed and evaluate using three robots concurrently for each scene. The adjustment of the play rate is necessary to facilitate the smooth running of multi-robot SLAM, given its characteristic requirement for substantial computational resources.

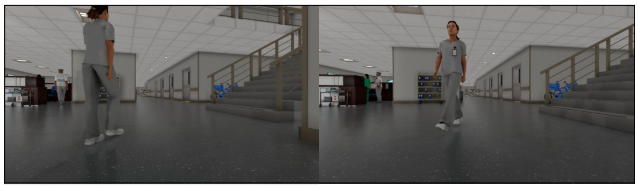


Figure 6. **Challenging case on SLAM**. This figure represents the failure case that occurred during evaluating SLAM algorithms on our dataset. In this case, invalid feature matching occurred due to the moving and dynamic object at different times.

4.3. Results and analysis

C-SLAM for Multi-Robot. Table 3 shows the evaluation results of multi-robot SLAM on our dataset. In static environments, Swarm-SLAM shows robust performance compared to other multi-robot SLAM algorithms on RGB-D and stereo setup. On the other hand, in dynamic environments, there is a notable reduction in performance for both setup, especially in *Hospital*. This performance drop indicates that Swarm-SLAM encounters challenges across both modalities in dynamic environments. However, Swarm-SLAM-D shows higher accuracy compared to Swarm-SLAM due to the dynamic feature removal module as shown in Table 3. It indicates that the presence of dynamic objects has a significant impact on the performance of Swarm-SLAM. The result of dynamic feature removal of Swarm-SLAM-D is available in Appendix.

For COVINS, it shows inferior performance and numerous failures throughout our dataset. This is because centralized SLAM shares one global frame among the robots; a failure case occurring in a single robot can also affect another. Similarly, we also experimented with Kimera-Multi, which failed in all sequences.

In addition, we observe that multi-robot SLAM can fail even if the place recognition module performs well. For example, although a robot recognizes the same place by the place recognition module (see Fig. 6), dynamic objects observed in the viewpoint can cause incorrect

Table 3. **Baseline evaluation of C-SLAM for multi-robot** (RMS ATE in meter).

			COVINS [19]	Swarm-SLAM [12]		Swarm-SLAM-D (w/ Dynamic)	
Sequences			Monocular-Inertial	RGB-D	Stereo	RGB-D	Stereo
Static	*Hospital*	ROBOT 1	9.431	0.176	0.387	–	–
		ROBOT 2	2.010	0.149	0.405	–	–
		ROBOT 3	△	0.130	0.318	–	–
	Office	ROBOT 1	9.232	1.281	0.104	–	–
		ROBOT 2	8.917	1.288	0.121	–	–
		ROBOT 3	8.774	0.739	0.067	–	–
	Warehouse	ROBOT 1	×	0.827	0.189	–	–
		ROBOT 2	×	0.697	0.165	–	–
		ROBOT 3	×	0.588	0.120	–	–
Dynamic	*Hospital*	ROBOT 1	×	9.441	13.670	0.585	0.323
		ROBOT 2	×	1.797	0.426	1.586	0.440
		ROBOT 3	×	8.431	1.522	2.343	0.739
	Office	ROBOT 1	×	0.888	0.113	0.198	0.089
		ROBOT 2	×	0.721	0.111	0.376	0.072
		ROBOT 3	×	0.491	0.079	0.229	0.126
	Warehouse	ROBOT 1	9.560	0.894	0.228	15.721	0.313
		ROBOT 2	7.203	0.582	0.192	0.231	0.159
		ROBOT 3	11.539	0.670	0.151	15.031	0.122

× Fail to estimate trajectory due to algorithm halt during operation.

– We do not evaluate static conditions for Swarm-SLAM-D.

△ Only run ROBOT 1 and ROBOT 2, excluding ROBOT 3 due to its influence on the algorithm's shutdown.

feature matching. As a result, the existence of dynamic objects can lead to inaccurate bundle adjustment. This rare edge case in single-robot SLAM becomes more prevalent in multi-robot SLAM, where robots exchange observations in various encounter cases, such as Overlap, Follow, and Intersect. The performance of multi-robot SLAM is affected not only by dynamic characteristics but also by various other characteristics in our dataset. We provide additional analysis of these characteristics and influences in the Appendix.

In summary, the proposed CSE dataset can help analyze how the characteristics contained in the service environment affect SLAM algorithms and have the potential to improve SLAM performance in the service environments. We will release our dataset for promoting C-SLAM in robotics community.

5. Conclusion

In this work, we have proposed the CSE dataset, a new synthetic C-SLAM benchmark dataset for multiple service robots. Unlike previous C-SLAM datasets that are mainly acquired in urban outdoor scenes or limited indoor environments (*e.g.*, corridors or room-size laboratories), we focused on acquiring C-SLAM datasets with three robots from three common indoor service environments that reflect diverse, challenging cases that may occur in real-world service environments. Each environment is divided into static and dynamic, and each robot drives through the same scenario in both static and dynamic environments. This design strategy provides an opportunity to evaluate the effectiveness of multi-robot SLAM in dealing with a variety of dynamic objects. We also build data that reflects the driving properties of real-world service robots since the robots recognize their surroundings and drive themselves in the simulation. Through all these various characteristics, we expect our CSE dataset will contribute to the advancement of C-SLAM research for multiple service robots.

Acknowledgements. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00907, Development of AI Bots Collaboration Platform and Self-organizing, No.2020-0-01336, Artificial Intelligence Graduate School Program (UNIST)).

References

[1] https://github.com/michaelgrupp/evo. 7

[2] Nvidia isaac sim, https://developer.nvidia.com/isaac-sim. 2, 3

[3] Nvidia omniverse, https://www.nvidia.com/en-us/omniverse/. 3

[4] https://www.market-prospects.com/articles/what-is-a-service-robot. 1

[5] Siddharth Agarwal, Ankit Vora, Gaurav Pandey, Wayne Williams, Helen Kourous, and James McBride. Ford multi-av seasonal dataset. *IJRR*, 39(12):1367–1376, 2020. 2

[6] Yun Chang, Yulun Tian, Jonathan P How, and Luca Carlone. Kimera-multi: a system for distributed multi-robot metric-

semantic simultaneous localization and mapping. In *ICRA*, 2021. 7

[7] Rodolphe Dubois, Alexandre Eudes, and Vincent Frémont. Airmuseum: a heterogeneous multi-robot dataset for stereo-visual and inertial simultaneous localization and mapping. In *MFI*, 2020. 2, 3, 4

[8] Dapeng Feng, Yuhua Qi, Shipeng Zhong, Zhiqiang Chen, Yudu Jiao, Qiming Chen, Tao Jiang, and Hongbo Chen. S3e: A large-scale multimodal dataset for collaborative slam. In *arXiv*, 2022. 2, 3, 4

[9] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous robots*, 8:325–344, 2000. 2

[10] Cullen Jennings, Don Murray, and James J Little. Cooperative robot localization with vision-based mapping. In *ICRA*, 1999. 2

[11] A.A. Nippun Kumaar and Sreeja Kochuvila. Mobile service robot path planning using deep reinforcement learning. *IEEE Access*, 2023. 2

[12] Pierre-Yves Lajoie and Giovanni Beltrame. Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. In *arXiv*, 2023. 7, 8

[13] Pierre-Yves Lajoie, Benjamin Ramtoula, Fang Wu, and Giovanni Beltrame. Towards collaborative simultaneous localization and mapping: a survey of the current research landscape. In *arXiv*, 2021. 2, 3

[14] Keith YK Leung, Yoni Halpern, Timothy D Barfoot, and Hugh HT Liu. The utias multi-robot cooperative localization and mapping dataset. *IJRR*, 30(8):969–974, 2011. 2, 3

[15] Shih-Yun Lo, Benito Fernandez, and Peter Stone. Iterative human-aware mobile robot navigation. In *RSS*, 2017. 1

[16] Ming Ouyang, Xuesong Shi, Yujie Wang, Yuxin Tian, Yingzhe Shen, Dawei Wang, Peng Wang, and Zhiqiang Cao. A collaborative visual slam framework for service robots. In *IROS*, 2021. 1, 2

[17] Stefanie Paluch, Jochen Wirtz, and Werner H Kunz. Service robots and the future of services. *Marketing Weiterdenken: Zukunftspfade für eine marktorientierte Unternehmensführung*, pages 423–435, 2020. 1, 2

[18] Laurel D Riek. The social co-robotics problem space: Six key challenges. In *RSS*, 2013. 1

[19] Patrik Schmuck, Thomas Ziegler, Marco Karrer, Jonathan Perraudin, and Margarita Chli. Covins: Visual-inertial slam for centralized collaboration. In *ISMAR-Adjunct*, 2021. 6, 8

[20] Xuesong Shi, Dongjiang Li, Pengpeng Zhao, Qinbin Tian, Yuxin Tian, Qiwei Long, Chunhao Zhu, Jingwei Song, Fei Qiao, Le Song, et al. Are we ready for service robots? the openloris-scene datasets for lifelong slam. In *ICRA*, 2020. 1, 2

[21] Jörg Stückler, Kathrin Gräve, Jochen Kläß, Sebastian Muszynski, Michael Schreiber, Oliver Tischler, Ralf Waldukat, and Sven Behnke. Dynamaid: Towards a personal robot that helps with household chores. In *RSS*, 2009. 1

[22] Sebastian Thrun. A probabilistic on-line mapping algorithm for teams of mobile robots. *IJRR*, 20(5):335–363, 2001. 2

[23] Yulun Tian, Yun Chang, Long Quang, Arthur Schang, Carlos Nieto-Granda, Jonathan P How, and Luca Carlone. Resilient and distributed multi-robot visual slam: Datasets, experiments, and lessons learned. In *arXiv*, 2023. 2, 3, 4

[24] Stefan B Williams, Gamini Dissanayake, and Hugh Durrant-Whyte. Towards multi-vehicle simultaneous localisation and mapping. In *ICRA*, 2002. 2

[25] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *IROS*, 2018. 7

[26] Yali Zheng, Shinan Chen, and Hong Cheng. Real-time cloud visual simultaneous localization and mapping for indoor service robots. *IEEE Access*, 8:16816–16829, 2020. 1

[27] Yilin Zhu, Yang Kong, Yingrui Jie, Shiyou Xu, and Hui Cheng. Graco: A multimodal dataset for ground and aerial cooperative localization and mapping. *RA-L*, 8(2):966–973, 2023. 2, 3, 4

[28] Danping Zou, Ping Tan, and Wenxian Yu. Collaborative visual slam for multiple agents: A brief survey. *Virtual Reality & Intelligent Hardware*, 1(5):461–482, 2019. 2

A Benchmark Synthetic Dataset for C-SLAM in Service Environments

Appendix

Overview

In this Appendix, we provide additional information that could not be handled in the main paper due to space constraints, as follows:

- In Sec. 1, we describe the details involved in constructing the dataset such as 1) the details of sensor modalities for each robot (Sec. 1.1), 2) the human assets examples we used to build the environments (Sec. 1.2), and 3) the various interaction types we considered when configuring the scenarios (Sec. 1.3).
- In Sec. 2, we describe additional details of experiments on SLAM algorithms such as 1) experiments on single-robot SLAM (Sec. 2.1), 2) implementation details of Swarm-SLAM-D (Sec. 2.2), 3) additional analysis of failure cases of SLAM in challenging scenarios (Sec. 2.3).

1. Additional details of the CSE dataset

In this section, we provide the additional information involved in constructing our CSE dataset.

1.1. Sensor configurations

Table 1 shows the sensor specifications, ROS topics, and topic message types provided for each robot. The topic name starts with the namespace of each robot (*e.g.*, carter1). RGB images are compressed for efficient storage, and depth images are scaled by a factor of 1000. It means that the units of the depth images corresponds to millimeter (*e.g.*, a pixel of 1000 of depth images is the same with a distance of 1 meter from the camera).

1.2. Dynamic objects

Our dataset, CSE, is collected in three common indoor service environments: *Hospital*, *Office*, and *Warehouse*, where real service robots operate. For configuring the more realistic environments in our dataset, we utilize the dynamic objects having suitable actions and clothes for each environment. For example, *Hospital* environment is mainly composed of doctors and nurses, who perform natural movements, such as walking around the rooms, answering a phone, or having conversations at the reception desk. Not only that, these dynamic objects can also cause challenges for the robot, such as extreme occlusions, since humans repeatedly walk around specific regions. In addition, these dynamic objects can cause incorrect feature matching. In the case of *Office* and *Warehouse* as well, they are composed of office workers or construction workers, all of whom perform movements appropriate to the environment, which can be seen in Fig. 1. As dynamic objects, we uilitze the human assets provided by NVIDIA and purchased assets from ActorCore [1] suitable for each environment.

(a) *Hospital* (b) *Office* (c) *Warehouse*

Figure 1. **Dynamic objects for each environment**. Each column shows the examples of the human assets used in each service environment. The first row represents the collections of some human assets used in each environment. The second row shows the example illustrations of human assets performing the appropriate motions for each environment.

(a) Follow (b) Intersection (c) Overlap

Figure 2. **Scenario Types**. Each column shows the examples of each scenario type. (a) The first column (Follow) describes a situation where one robot follows behind another. (b) The second column (Intersection) means where the robots are across each other. (c) In the third column (Overlap) refers to a situation where they pass through the same region at different times in the same direction. The blurry-colored and the dark-colored robots are different robots, and they pass through the same area at different times.

1.3. Scenario types

We design the scenarios considering diverse interactions, including inter- and intra-robot loop closures. In particular, in the case of inter-robot loop closure, we divide it into three types (Follow, Intersection, Overlap), as shown in Fig. 2. We configure our scenarios by evenly distributing these scenario types across the robot-specific scenarios.

2. Additional details of experiments on SLAM

In this section, we provide the additional details of experiments on single-robot SLAM and multi-robot SLAM algorithms. Visualization results for all SLAM algorithms we used can be seen in Fig. 6

[1] https://actorcore.reallusion.com/

Table 1. **Sensor Configurations.**

Data	Resolution	Rate	Topic Name	Message Type
Stereo left RGB	1280×720	30Hz	/carter/rgb_left/compressed	sensor_msgs/CompressedImage
Stereo right RGB	1280×720	30Hz	/carter/rgb_right/compressed	sensor_msgs/CompressedImage
Stereo left Depth	1280×720	30Hz	/carter/depth_left	sensor_msgs/Image
Stereo right Depth	1280×720	30Hz	/carter/depth_right	sensor_msgs/Image
IMU	-	120Hz	/carter/imu	sensor_msgs/Imu
Ground Truth Pose	-	120Hz	/carter/gt_pose	nav_msgs/Odometry

2.1. Experiments on single-robot SLAM

Baseline SLAM for single-robot is a task where SLAM is executed using a single robot, and notable instances of this approach include ORB-SLAM3 [1] and VINS-Fusion [3]. ORB-SLAM3 facilitates a range of camera configurations and demonstrates outstanding performance by leveraging pre-constructed maps in scenarios with restricted visual data. VINS-Fusion is an optimization-based odometry framework that utilizes visual and inertial information, integrating sensor data into pose graph optimization for accurate position estimation.

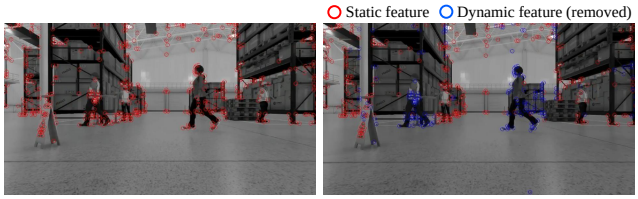
Experimental setup To comprehensively evaluate various sensor modalities offered by our dataset, we conduct evaluations using RGB-D, mono-inertial, and stereo-inertial provided by each baseline algorithm. For single-robot SLAM algorithms, we set the play rate of our dataset to 1.0× speed and perform evaluations of the all sequences for all robots in each environment.

Results and analysis In Table 2, we show the evaluation results for single-robot SLAM algorithms on our dataset. We observe that both ORB-SLAM3 and VINS-Fusion achieve the highest accuracy with the stereo-inertial setup on average. Notably, in the *Hospital*, various static objects, such as chairs and desks, make feature matching easier than in other environments, resulting in higher pose estimation accuracy. However, due to the challenging cases of each environment, specific sequences lead to bad estimation results. For example, We observe that ROBOT 3 performs incorrect place recognition in ORB-SLAM3 due to redundant objects in the *Warehouse*, resulting in low accuracy (see Fig. 3). Moreover, in the *Warehouse*, ROBOT 1 and ROBOT 2 encounter each other at close distance (see in Fig. 5(a)), which makes obstructions in the camera view of the moving robots, which leads to inadequate feature matching in the VINS-Fusion. We can see that these challenging cases adversely affect the accuracy of SLAM algorithms.

2.2. Implementation details of Swarm-SLAM-D

Swarm-SLAM with dynamic environment (Swarm-SLAM-D in short) is an algorithm we have implemented by incorporating a moving consistency check module into the existing Swarm-SLAM [2]. The module is based on DS-SLAM [4]. It identifies feature points as either static or dynamic and then removes dynamic feature points accordingly. The procedure of this module is as follows:

Figure 3. **Challenging case on single-robot SLAM**. This figure represents the failure case that occurred during evaluating SLAM algorithms on our dataset. In this case, failure of place recognition occurred due to the similar structure at different location.



(a) (b)

Figure 4. **The process of detecting and removing dynamic features**. (a) The left image shows the previous frame image, where red circles represent feature points before their status is determined. (b) In the right image, red circles indicate static feature points, while blue circles represent dynamic feature points detected and removed by the moving consistency check module.

This module takes the previous frame image, feature points of the previous frame image, and the current frame image as inputs. From these inputs, it calculates the optical flow to extract feature points for the current frame image. Based on these extracted feature points, it computes the fundamental matrix and then uses it to calculate the epipolar lines. When feature points exceed a specified threshold with respect to the epipolar lines, they are identified as dynamic and removed dynamic feature points accordingly. Figure 4 illustrates the process of removing dynamic features based on the previous and current frames image.

Table 2. **Baseline evaluation of single-robot SLAM** (RMS ATE in meter).

			ORB-SLAM3 [1]			VINS-Fusion [3]	
Sequences			RGB-D	Monocular-Inertial	Stereo-Inertial	Monocular-Inertial	Stereo-Inertial
Static	*Hospital*	ROBOT 1	0.015	0.191	0.017	4.863	0.331
		ROBOT 2	0.057	16.056	0.061	0.513	0.210
		ROBOT 3	0.040	7.897	×	2.555	0.387
	Office	ROBOT 1	0.058	3.764	0.038	2.042	0.213
		ROBOT 2	0.023	8.903	6.676	3.513	0.471
		ROBOT 3	3.432	0.094	3.040	4.244	0.170
	Warehouse	ROBOT 1	0.565	×	×	17.027	1.809
		ROBOT 2	0.030	0.198	0.030	10.726	0.328
		ROBOT 3	16.490	0.373	4.707	5.832	1.957
Dynamic	*Hospital*	ROBOT 1	0.023	0.108	0.019	2.244	0.525
		ROBOT 2	0.098	×	×	1.999	0.346
		ROBOT 3	0.030	10.340	0.033	3.376	0.348
	Office	ROBOT 1	0.090	0.256	13.424	2.546	0.253
		ROBOT 2	0.060	×	×	6.740	0.442
		ROBOT 3	0.067	0.072	0.022	3.450	0.263
	Warehouse	ROBOT 1	0.054	0.552	0.068	4.322	1.322
		ROBOT 2	0.021	0.259	0.035	4.268	0.496
		ROBOT 3	16.682	0.399	0.076	5.385	0.364

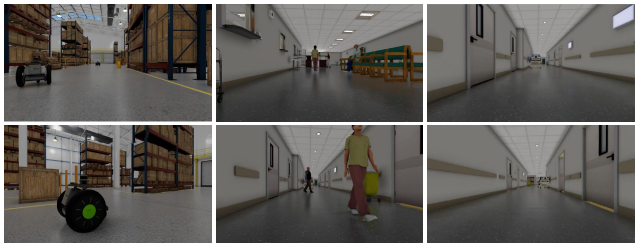
× Fail to obtain trajectory due to algorithm halt during operation.

2.3. Analysis of failure cases in challenging scenarios

We note further scenarios in which SLAM algorithm faces difficulties due to the challenging characteristics we introduced. There is a scenario where robots are interacting with each other (see Fig. 5(a)). When ROBOT 1 and ROBOT 3 are close, the moving robots block parts of the camera's view, making it hard to capture visual features. Also, the movement of robots to avoid each other changes their speed. This change can make it difficult for SLAM algorithms to accurately calculate the distance the robot moved. These problems cause the algorithm to make an incorrect trajectory path.

In service environments, it is common for humans and robots to closely interact. Our dataset replicates this phenomenon. In *Hospital* environment, a robot traversing the corridor encounters a walking nurse, resulting in the robot's camera being substantially obscured by the nurse (see Fig. 5(b)). This interaction, where a dynamic object (the nurse) extensively obscures the static environment, interrupts the extraction of features of the static object. Consequently, this leads to a degradation in the performance of visual SLAM algorithms.

There is another challenging case in a *Hospital* environment, where the algorithm incorrectly identifies different spaces as the same (see Fig. 5(c)). This issue is arises because the *Hospital* environment is mainly composed of homogeneous floors and walls that are almost identical, and even if robots are arrive at different locations, these characteristics can lead to incorrect inter-robot loop closing. During the feature-matching process, distinguishing between features becomes challenging, and the limited number of features further interrupts effective matching. This leads to computational bottlenecks in the operation of SLAM algorithm.



(a) (b) (c)

Figure 5. **Challenging case on SLAM**. Each column presents the moments when SLAM algorithms fail to estimate the accurate trajectory. (a) The first column shows the robot-robot interaction, (b) the second column shows the dynamic human object, and (c) the third column shows the homogeneous region.

References

[1] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *T-RO*, 37(6):1874–1890, 2021. 2, 3

[2] Pierre-Yves Lajoie and Giovanni Beltrame. Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. In *arXiv*, 2023. 2

[3] Tong Qin, Shaozu Cao, Jie Pan, and Shaojie Shen. A general optimization-based framework for global pose estimation with multiple sensors. In *arXiv*, 2019. 2, 3

[4] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *IROS*, 2018. 2

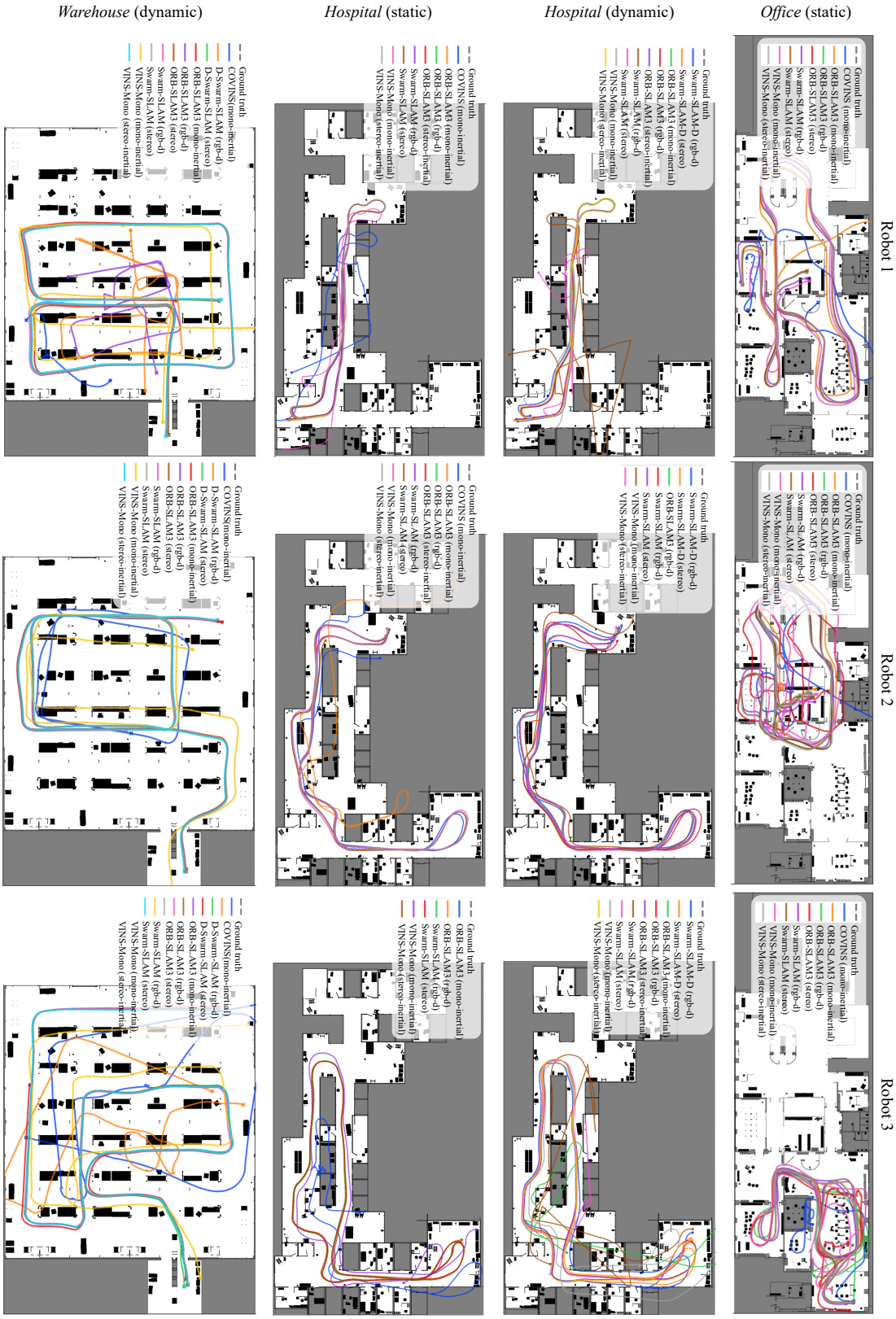


Figure 6. **The visualization of the results of SLAM algorithms in our dataset**. Note that we only visualize the experiments that success in full sequence.