

# A Human-Robot Collaborative 3D SLAM-XR Framework

Malak Sayour, Mohamad Karim Yassine, Nadim Dib, Imad H. Elhaji, Boulos Asmar, Elie Khoury, Daniel Asmar

**Abstract**—In this paper, we propose a collaborative centralized 3D mapping and localization framework that harnesses the capabilities of both SLAM (simultaneous localization and mapping) and XR (eXtended reality). On one hand, our framework allows for the integration of local maps generated by a multitude of heterogeneous agents (*e.g.* robots) into a unified map. On the other hand, it allows human intervention at multiple levels: first, humans can inspect and intervene in the mapping process in-situ to produce 3D maps, to overlay virtual assets, and to add annotations, all of which can contribute towards producing the first layer of a digital twin. Second, beyond the mapping aspect a human can also intervene in the localization task of any collaborating robot, by inspecting and correcting its generated paths and if necessary enforcing any desired trajectory. The proposed system is successfully demonstrated inside a real experimental setting.

## I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) stands as one of the testaments to the strides made in autonomous systems and robotics. Initially conceived as a method for robotic navigation [1], SLAM has transcended its original confines and has become a useful tool in a myriad of applications. In recent years, SLAM has witnessed a paradigm shift in its utilization, driven by the autonomous advancements in various industries. From self-driving cars [2] to digital twins [3], SLAM has proven its mettle in providing machines with the spatial awareness necessary for intelligent decision-making.

Similarly, the world of eXtended Reality (XR) has undergone a transformative evolution, transcending the realm of entertainment to revolutionize diverse sectors [4]. Industries ranging from healthcare and education to manufacturing and design have embraced XR, leveraging their potential to enhance training, visualization, and collaboration [5], [6]. The ability of XR to bridge the gap between the physical and digital worlds has opened avenues for innovative applications that redefine traditional workflows.

Finally, digital twins constitute living digital surrogates of objects and processes from the world around us. A digital twin of a physical space can include many semantic layers to it (*e.g.*, furniture, electric works, etc), but the base

M. Sayour, M. K. Yassine, I. H. Elhaji, and D. Asmar are with the Vision and Robotics Lab, Maroun Semaan Faculty of Engineering and Architecture, American University of Beirut, 1107 2020, Riad El Solh, Beirut, Lebanon; email: ms423@aub.edu.lb, mhy16@mail.aub.edu, ngd04@mail.aub.edu, ie05@aub.edu.lb, da20@aub.edu.lb. (M. Sayour and M. K. Yassine are co-first authors.)

B. Asmar and E. Khoury are with idealworks, Riesstraße 22, 80992 München, Germany; email: Boulos.Asmar@idealworks.com, Elie.Khoury@idealworks.com

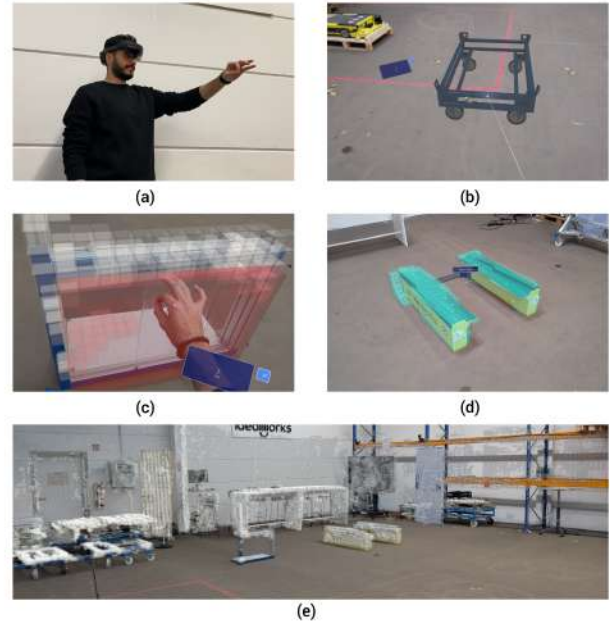


Fig. 1. Overview of the proposed system. (a) a human wearing a Microsoft HoloLens; (b) adding an asset into a digital twin; (c) human editing the produced map; (d) automatic labeling of assets; and (e) the generated map seen from the head mounted display.

foundational layer is a reconstructed digital map (2D or 3D) upon which the remaining layers are superposed.

The synergy between SLAM and XR heralds a new era of spatial understanding and interaction. By integrating SLAM's real-time mapping capabilities with XR's immersive experiences, a host of possibilities can be unlocked. The convergence of SLAM and XR can redefine spatial cognition, navigation, and interaction by seamlessly integrating digital information with the physical environment.

In this paper, we introduce a modular framework designed to facilitate collaborative mapping by harnessing the synergies of XR and SLAM. Our proposed system focuses on critical aspects such as map quality enhancement, robot navigation precision, map editing efficiency, map annotation, and its implications for multi-agent mapping scenarios. The aim is to create a dynamic digitized version of a navigated environment using the same agents responsible for their respective daily autonomous tasks, while at the same time empowering a human operator with the capability to intervene in the digitization process, and enabling immersive monitoring capabilities. The targeted end application for this

work is the production of a base physical layer of a digital twin.

The proposed approach enables a collaborative environment between humans and robots via XR, addressing SLAM inaccuracies arising from sensor noise or reflective and translucent surfaces. This collaborative framework significantly enhances the quality of generated maps, and empowers human agents to intervene and correct errors in robot ego pose estimates. The framework also allows for the inclusion of semantic information, which could be used to build additional layers for the digital twin. We introduce an automated labeling approach using a 6D object pose estimation model complemented by human manual labeling for added flexibility. As the landscape of multi-agent SLAM continues to evolve, our modular architecture is capable of fusing sub-maps from any number of agents, leveraging the human agent and XR technology for precise sub-map alignment. The key contributions of our paper include:

- A modular multi-agent framework enabling collaborative heterogeneous mapping.
- Enhanced mapping accuracy by leveraging human-robot collaboration.
- Improved localization and navigation thanks to improved path planning on human-corrected map, human-assisted localization, and the use of multi-agent maps for new robot navigation.
- Semantic enrichment by incorporating manual and autonomously generated semantic information into the mapping process.
- Advanced visualization and editing capabilities, by allowing for predefined CAD insertion of irregular shapes leading to modular map editing all while providing an immersive visualization of the generated map.

The remainder of the paper is structured as follows: Section II provides a concise overview of the relevant literature, Section III introduces our proposed methodology, Section IV discusses the experiments and their results, and Section V serves as the paper's conclusion. The implemented code is available on GitHub<sup>1</sup>.

## II. RELATED WORK

Numerous efforts have been made to integrate XR and SLAM for enhanced autonomous navigation. Kozlov *et al.* [7] suggested the utilization of Augmented Reality (AR) as a tool for debugging, assessing, and refining SLAM algorithms. They contended that AR offers the capability to visualize the internal program state of a robot and pertinent information, specifically within the realm of testing and debugging SLAM algorithms. By leveraging the uncertainties and error sources inherent in SLAM, as identified in existing literature, the authors formulated requirements that an AR system must meet to optimize the testing, debugging, and design phases in SLAM. Nevertheless, their proposed methodology falls short in addressing post-map processing and lacks the functionality for human agents to intervene in

mapping processes, accurately localizing a misplaced robot, or assigning semantic labels.

Daily *et al.* [8] converted a robot swarm into a distributed interface integrated into the environment. Their methodology leverages AR to convey information from numerous small-scale robots to humans, facilitating situation awareness, monitoring, and controlling in applications such as surveillance, reconnaissance, and path-finding. Despite successfully addressing several challenges, including supervisory management and monitoring of both individual and collective states, their proposed approach falls short in granting users the capability to intervene in the mapping process by making corrections or additions to problematic or virtual obstacles.

Nunez *et al.* [9] devised a mechanism enabling the overlay of planning, and sensory data from the robot onto the human field of view using AR. This design allows the human user to set and manipulate map nodes, as well as visualize and correct the robot's path planning in an intuitive manner.

In our previous work [10], we examined the impact of human-assisted SLAM utilizing AR, where we harnessed the distinctive capabilities of mobile robots and human operators to improve the quality of generated maps. The system was capable of enhancing 3D maps, aligning sub-maps, and visualizing the final map. However, it falls short in many aspects towards creating a base digital replica for a sought after environment. First, the system supports only two agents, where one has to be a human and the other a robot. Second, the merging logic relies on a predefined decision table, which lacks flexibility, generality, and risks being applied in an ad-hoc manner. Third, the contributions of the human agent to the mapping process is constrained, with interventions primarily focused on voxel edits and sub-map alignment. Users are unable to add assets or inspect autonomously generated paths by mobile robots. Forth, semantic data is absent from the system, since it lacks the ability to generate semantic data and the architecture cannot integrate captured semantics. Finally, the system does not encompass aspects of robot navigation and localization. In this paper, our new proposed system overcomes these limitations by allowing scalability to accommodate more agents, expanded human intervention capabilities, inclusion of semantic data generation and integration, and an enhanced focus on robot navigation and localization within the AR-assisted SLAM framework. The resulting output map constitutes a more promising candidate for a digital twin base layer.

## III. SYSTEM OVERVIEW

We adopt a centralized approach, where all computations except for mapping are executed at the central unit (Fig. 2). The connectivity framework involves mapping agents linked to the central unit through the ROS2 DDS server, while XR devices establish connections via the ROS TCP endpoint package within Unity [11]. Heterogeneous agents can participate in the mapping process. Mobile robots equipped with 3D LiDARs or cameras generate 3D octomaps of the navigated environment using LiDAR-based and vision-based SLAM, respectively. While the XR agent participates in multi-map

<sup>1</sup><https://github.com/AUBVRL/SLAM-XR>

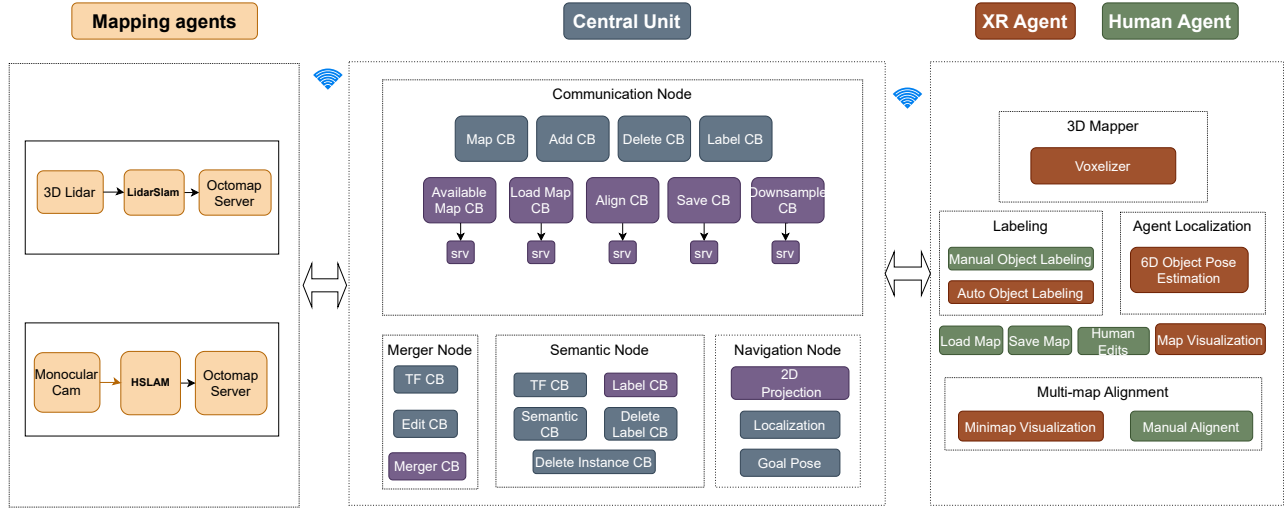


Fig. 2. System architecture

alignment, map accuracy inspection, edits application, and labeling objects of interest. The central unit comprises four key nodes: the Communication node, the Merger node, the Semantic node, and the Navigation node.

1) *Communication node:* the Communication node operates as a multi-threaded entity responsible for managing communication among various agents, functioning as a communication layer with multiple callbacks. The *mapcallback* oversees the retrieval of submaps from all mapping agents, storing them in an octomap dictionary. Each mapping agent, identified by a unique ID, publishes its generated map to the */octo - ID* topic as an octomap ROS message. Additional callbacks, namely *add*, *delete*, and *label*, handle the acquisition of voxels added to the map, voxels removed from the map, and voxels assigned specific labels by the human agent, respectively. Beyond these callbacks, services are established and activated from the communication node based on human input. All services are part of a re-entrant callback group, facilitating multi-threaded execution. The *downsampler* service, triggered by an ID from the human agent, responds with a down-sampled version of the requested agent sub-map, facilitating easier visualization. The *align* service, initiated with two IDs from the human agent and an initial transformation as a ROS twist message, computes a refined transformation using the Iterative Closest Point (ICP) algorithm. It subsequently publishes the refined transformation, aligning the sub-maps from the two provided agents. For extra flexibility, the *Save* service is designed to store the fused final map for future applications. This service offers the capability to save the fused map in two formats: a binary tree file suitable for octomap visualization and a .pcd file tailored for semantic point cloud visualization. In cases where mapping is based on previously saved maps, the *availableMap* service furnishes the user with a list of all offline maps at their disposal, while the *loadMap* service facilitates the integration of a user-selected offline map into the fusion process.

2) *Merger node:* the Merger node undertakes the task of integrating sub-maps from all agents into a cohesive and comprehensive map. This node operates with two primary callbacks on a single thread. The Edit callback handles the incorporation of human edits received from the communication node into the merging process. The pivotal functionality of the merger node unfolds in the Merge callback, a periodic callback triggered at a frequency set by the user. During each invocation, the Merge callback iterates over all sub-maps and identifies those that have been aligned for further processing. Aligned maps, if updated, are then considered in the merging process. Each agent is assigned a pre-defined confidence level that influences its weight in contributing to the final merged map. A weighted average is subsequently employed to calculate the occupancy probability of a particular voxel as follows:

$$MT[n] = \frac{(WT[n] * MTO[n] + AW * SO[n])}{(WT[n] + AW)}, \quad (1)$$

$$WT[node] = WT[n] + AW * WT[n], \quad (2)$$

where *MT* stands for mergedTree occupancy tree, *WT* for weightTree storing the cumulative node weight, *AW* for agentWeight specified in the central unit database, *SO* for the agent's submap occupancy value and *n* for the node in question.

Voxels that have been modified by human actions (either added or deleted) are assigned specific and clearly defined log-odds values, ensuring their distinct identification from other voxels and preserving their integrity in subsequent fusion processes.

3) *Semantic node:* the Semantic node is responsible for associating semantic data with the fused map, and features five key callbacks. The *tf* callback and the *label* callback are engaged in acquiring refined transformation and manual/automatic label information. The *semantic* callback is tasked with integrating semantic data into the merged map to generate a semantic point cloud. The resultant point

cloud exhibits three custom fields, in addition to the regular xyz fields: a probability field indicating the confidence of occupancy, a label field denoting the class to which a point belongs, and an instance field used for tracking the available instances of a certain class. To enhance flexibility and manipulation, the *deleteLabel* callback facilitates the deletion of all instances associated with a specific class. Simultaneously, the *deleteInstance* callback is designed to remove a particular instance linked to a specific class from the fused map.

4) *Navigation node*: the Navigation node plays a crucial role in preparing the necessary data for enabling robot navigation, featuring two key callbacks: the *occupancyGrid* callback and the *localize* callback. The *occupancyGrid* callback generates a 2D occupancy grid derived from the 3D octomap, projecting obstacles within a specified height range onto the robot plane for navigation purposes. On the other hand, the *localize* callback establishes the transformation between the map frame, upon which the merged map is based, and the baselink frame of the robot. Following this calibration, the robot gains the capability to navigate the environment autonomously, utilizing the nav2 bringup node provided by the navigation stack of ROS2.

#### A. XR Agent

In addition to the capabilities that the XR agent had in our previous work [10], the current system introduces the following:

1) *Voxel grid mapping*: the XR agent utilizes the map generated by the device to create a voxelized version of the environment. The *Voxelizer* node handles the voxelization and refinement of the map simultaneously. To attain the concurrency of addition and deletion, ray cast techniques are replaced by box collider overlaps, which are instantiated in a grid-like manner in the scene with dimensions of 5 cm for each side. Each collider checks if it is overlapping another one; if the discovered collider is the spatial mesh generated by the XR agent, then the voxel's probability at the collider's position is increased by 5%. If the overlapped collider is that of a misplaced voxel (where it does not intersect with the spatial mesh), then that voxel's probability is decreased by 1%. The voxel's probability is assessed after each frame to either instantiate or destroy the voxel based on a preset threshold of 60%.

2) *Automatic labeler*: in an industrial setting, most of the recurring objects in the environment are known objects that possess a 3D CAD model such as robots, charging stations, pallets, etc. Therefore, 6D object pose estimation models and tracking techniques are employed to automatically detect and track the objects of interest. A neural network model is trained using the provided 3D CAD models to generate a deployable Unity package, which is utilized by the XR to detect, track, and acquire the 3D transformation of the objects of interest. Once the user confirms a successful alignment, a voxelized version of the object is positioned at the obtained transformation and labeled with a name and an instance number. The labeled voxels are then published

to the *SemanticNode* to incorporate them into the global map.

3) *Agent localization*: agents that are equipped with low-quality sensors might have trouble applying SLAM, making them incapable of consistently and accurately localizing themselves in a given environment. The XR agent can remedy that issue by localizing them using 6D object pose estimation. The transformation of the physical agent is obtained and used to localize the agent in the XR's local map. Knowing the transformation between the XR's local map and the global map from the alignment step, the agent's location is transformed from the local map frame to the global map frame. The resulting transformation is published to the navigation node to update the global map with the agent's location, making the agent virtually visible on the grid for other agents and itself.

4) *Path visualization and editing*: autonomous agents rely on path-planning algorithms to reach desired goals in a given environment. The XR agent is capable of visualizing these paths by overlaying them onto the environment. The user can then assess if the agent has successfully planned a safe path around the obstacles in the environment. Knowing that some obstacles might be problematic for the autonomous robots to map, human intervention alleviates these limitations by manually editing any part of the map that is not accurate. These edits will be reflected in the autonomous agent's planned routes since the agent will now account for the previously unseen obstacles. This also facilitates robots and humans working side by side by enhancing safety.

#### B. Human-in-the-loop

The human agent's involvement is cardinal as it ensures the refinement and accuracy of the global map. The key influences on the system are the following:

1) *Multi-map alignment*: the human agent possesses the flexibility to align any agent's sub-map using the XR's user-friendly interface displayed in Fig. 3. Initially, the user aligns the XR agent's sub-map with any available agent's sub-map, ensuring that the two sub-maps share overlapping features from the environment for successful alignment, resulting in the creation of a global map accessible to all agents on-site. Subsequently, the user can align any agent's sub-map into the global map using the same approach. Each successful alignment integrates the sub-map into the global map, showcasing the collaborative mapping capabilities inherent in our system.



Fig. 3. XR agent's user interface



2) *Voxel editing and asset injection*: the XR agent offers the user the choice of editing the map in real-time. The edits are of three types:

**Voxel addition**: when an agent fails to map the environment correctly, such as in unreachable areas or problematic obstacles, the user can correct the global map by adding voxels to the environment. The added voxels are published to the semantic node to update the global map.

**Voxel deletion**: the process follows a similar methodology to that mentioned above. The deleting feature is crucial for scenarios where an agent maps an undesired area or obstacle.

**Asset injection**: assets such as robots, charging stations, pallet docker, etc. are pre-loaded to the XR agent as CAD files. The 3D models are embedded with the physical size, shape, and name of the real assets. The human agent can inject these assets into the environment, which automatically incorporates the global map with the changes. The added assets are also labeled with the CAD's name and published to the semantic node.

3) *Manual labeling*: manual labeling is employed for objects that recur in the environment but lack a pre-loaded CAD file. This feature utilizes the same selection technique applied in the Voxel Editing section up to the point of confirmation. Following the confirmation of the selected voxels, the user is prompted to either input a new label name or select a previously added label. The labeled voxels are then published to the semantic node, which treats each instance of a label as a distinct unit. The labeling feature empowers the human agent to delete a specific instance of an object by merely selecting it on-site. Additionally, the user can opt to delete all instances of a particular object from the environment.

#### IV. EXPERIMENTS AND RESULTS

The system underwent testing in a real industrial setup at the IDEALworks GmbH research and development center in Munich, Germany. Three different agents collaborated in the mapping process:

- The iw.hub Version 4 robot, equipped with an LS CH32R 3D LiDAR and operating on Ubuntu 20 with ROS 2 Foxy. Being a LiDAR-equipped agent, the robot runs the *lidarSlam* package on board, explained in [10].
- The iw.hub Version 4 robot, equipped with Intel D435 camera and operating on Ubuntu 20 with ROS2 Foxy. This agent maps the navigated environment using the RTAB-Map package [12].
- The Microsoft HoloLens 2 running the developed application detailed in section III-A.

Each mapping agent navigates a different part of the warehouse, and builds its own sub-map. The generated octomaps are subsequently published to the DDS server, each under its specific */octoID* topic for inclusion in the fusion process. Following the mapping process, and after covering a significant portion of the environment, the human agent aligns the XR's generated map with the robot's sub-map as detailed in [10]. The alignment generates a merged

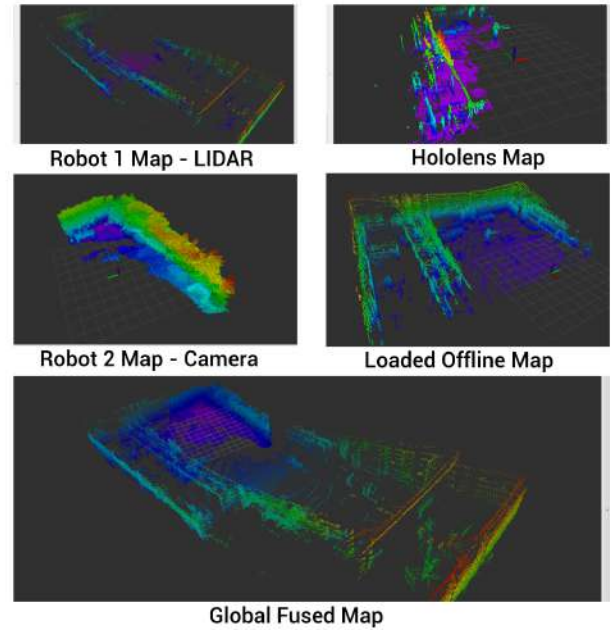


Fig. 4. Sub-maps generated by the different agents and the resultant fused map.







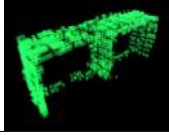
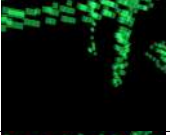

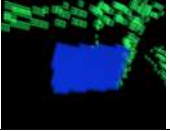
global map that the user uses to align the other sub-maps to it. All aligned sub-maps are then merged, generating a fused global map. The result is a complete representation of the environment that is achieved in a remarkably fast time. Instead of waiting for each agent to map the entire environment, partial maps are accurately integrated together. To save even more time, the user can load any part of the environment that was previously mapped, and it gets automatically embedded into the merging process leading to a complete global map as shown in Fig. 4.

To assess the resultant map, the user toggles the view merged map button which allows for the overlay of the generated map onto the real environment. Table I shows how various environmental features achieved accurate representation only upon the fusion of the agent's sub-maps. For areas that are inaccessible by the robot, the human agent maps the area in question and fuses the new region with the global map, making this region virtually accessible to all other agents on site. After examining the 1:1 generated map, the user identifies transparent objects in the mapped environment that are not captured by any of the three agents and proceeds to edit the generated map by adding the missing obstacles.

To enrich the map with semantic data, the XR agent navigates the environment and automatically detects pre-trained objects using a 6D pose estimation model provided by Vuforia. The developed algorithm assigns on-the-go labels to mobile robots, charging stations, dollies, and pallet dockers. Fig. 5 shows the automatic labeling of the iw.hub robot. Once the iw.hub is identified, the mesh representation is overlaid on the real robot, and the user is prompted with a confirmation menu. Once the confirm button is pressed, a

TABLE I

MAPPING RESULTS. GROUND TRUTH (ROW 1); ROBOT 3D MAP (ROW 2); MERGED MAP (ROW 3); HUMAN EDITED MAP (ROW 4). THE GROUND PLANE AND WALLS ARE CLIPPED FOR VISUALIZATION PURPOSES.

	IDEALworks GmbH Research and Development center		
	Table	Glass panel	Inaccessible area
Ground Truth			
Robot Map			
Merged Map			
Edited Map	No Edits Needed		No Edits Needed

voxelized representation of the asset is injected to the fused map along with the asset's label. For easier tracking, each labeled object is represented in a light blue color within the merged map. In instances where the human operator desires to label a custom object of interest not covered by the pre-trained models, manual labeling is accomplished by pinching and selecting the volume of interest, as depicted in Fig. 6. After selecting the object, the user is prompted to input the desired label or choose one of the previously labeled objects. If the label is not already present, the object is assigned an instance value of one, meaning it is the first asset belonging to the inputted label. If the label is already present, the instance value is incremented by one. This modular and flexible approach for semantic map creation allowed for a focused representation of only the pertinent objects, preventing unnecessary data overload on the server.



Fig. 5. Automatic labeling. From left to right, the human agent first looks at the physical iw.hub. Once detected, an overlay is then visualized along with a confirmation menu. After confirmation, the asset is voxelized and shown in cyan.

At this stage, a complete and correct representation of the surrounding environment with semantic-enriched data is achieved. The user then proceeds to the pre-deployment phase. For the facilitation of modular robot deployment and



Fig. 6. Manual labeling. The human agent first selects the voxels of interest using hand gestures (selected region is shown in burgundy on the left). The label name is inputted using a virtual keyboard (middle), and the voxels of interest are successfully updated (shown in yellow on the right).

accurate environment representation, the user is capable of adding obstacles and assets to the generated map, even if the physical version is absent from the scene. Fig. 7 demonstrates the process of adding a charging station in its anticipated future location facilitating robot deployment. The human agent chooses the object of interest from a drop-down menu and proceeds to insert the selected object in the desired location. Once satisfied with the position, the user presses the confirmation button, and a light blue voxelized version of the asset is added to the merged map along with its label. The XR agent is also capable of adding virtual obstacles to force the navigation node to generate a certain autonomous path, compensating for potential drifts as seen in Fig. 8.

Following the generation of the fully labeled map, the user proceeds to save the generated map. The human agent triggers the save map button using the XR menu and inputs the desired file name for the current global map to be saved. The map is saved as a .bt file for loading it into the system, and a .pcd file for a point cloud representation of the environment.

At the deployment phase, the user loads the generated map and aligns it with the XR agent's current map. The navigation node generates the 2D occupancy grid by projecting a section of the 3D octomap onto it. Next, the XR agent automatically detects the iw.hub robot using the same 6D pose estimation model used for automatic labeling. The obtained transformation of the robot is published to the navigation node. Following this step, the robot is then ready to navigate the environment autonomously. Tests are conducted to evaluate the impact of edits on the robot's navigation. Commands are issued to the iw.hub, allowing the user to visualize and



Fig. 7. Adding assets. From left to right, the human first chooses the asset he would like to add and places it in the desired location using hand gestures. After confirmation, the asset is voxelized and labeled.

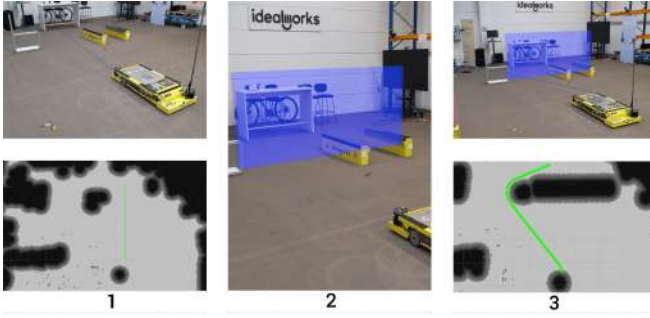


Fig. 8. Forcing a desired path. The path planned by the robot (left); the process of adding an obstacle to block the robot's planned path (middle); the corrected robot's path after an obstacle is added. The upper row represents the path visualized from the HoloLens while the lower row shows the path displayed in RViz

inspect the planned path. As depicted in Fig. 9, the correction of the map's inaccuracies and the addition of unmapped obstacles played a crucial role in securing a safe path, thereby ensuring that the robot avoids a potential catastrophic collision. The path visualization feature allows the user to inspect the planned path which is a very helpful tool for Day Zero robot deployment. Upon the completion of the tests, the human agent is then capable of saving the generated map for future use or visualizations in alternative software applications. This functionality allows for the preservation and utilization of the map data beyond the testing phase.

## V. CONCLUSION

In this paper, we presented a collaborative human-assisted SLAM approach based on XR. By fusing XR and SLAM, our system proved efficient in fusing maps from any number of heterogeneous agents, while increasing the generated map's quality. The proposed approach allows for automatic and manual labeling, leading to semantically rich maps, that lend themselves to be used as base layers for digital twins of environments. The proposed system was successfully tested in a real experimental setting. Future iterations of our framework can explore automatic point cloud segmentation based on user-generated labeled point clouds. This extension has the potential to further enrich the topological map with semantic information, subsequently improving the navigation capabilities of autonomous agents. The collaborative nature of our human-robot framework signifies a step towards cognitive mapping, harmonizing human reasoning with the mapping capabilities of robots. This collaborative synergy lays the groundwork for advanced applications in mapping, semantic understanding, and autonomous navigation.

## ACKNOWLEDGMENT

This work was supported by the DIDYMOS-XR Horizon Europe project (grant number 101092875–DIDYMOS-XR, [www.didymos-xr.eu](http://www.didymos-xr.eu)), and the Research Board at the American University of Beirut.

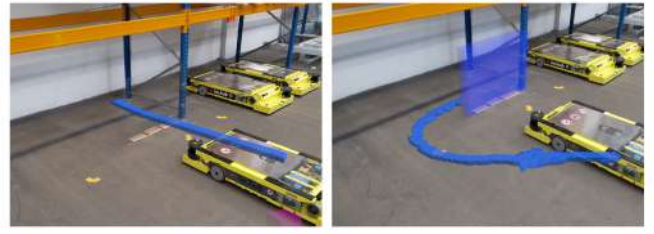


Fig. 9. Effect of human edits on robot navigation. Path before edits would have caused a robot collision with the glass pane that was not automatically detected.

## REFERENCES

- [1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.
- [2] T. T. O. Takleh, N. A. Bakar, S. A. Rahman, R. Hamzah, and Z. Aziz, "A brief survey on slam methods in autonomous vehicle," *International Journal of Engineering & Technology*, vol. 7, no. 4, pp. 38–43, 2018.
- [3] M. Sommer and K. Seiffert, "Scan methods and tools for reconstruction of built environments as basis for digital twins," *DigiTwin: An approach for production process optimization in a built environment*, pp. 51–77, 2022.
- [4] S. H.-W. Chuah, "Why and who will adopt extended reality technology? literature review, synthesis, and future research agenda," *Literature Review, Synthesis, and Future Research Agenda (December 13, 2018)*, 2018.
- [5] T. Morimoto, T. Kobayashi, H. Hirata, K. Otani, M. Sugimoto, M. Tsukamoto, T. Yoshihara, M. Ueno, and M. Mawatari, "Xr (extended reality: virtual reality, augmented reality, mixed reality) technology in spine medicine: status quo and quo vadis," *Journal of Clinical Medicine*, vol. 11, no. 2, p. 470, 2022.
- [6] J. M. M. Ferreira and Z. I. Qureshi, "Use of xr technologies to bridge the gap between higher education and continuing education," in *2020 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2020, pp. 913–918.
- [7] A. Kozlov, B. MacDonald, and B. Wünsche, "Towards improving slam algorithm development using augmented reality," in *Proceedings of Australasian Conference on Robotics and Automation*. Citeseer, 2007.
- [8] M. Daily, Y. Cho, K. Martin, and D. Payton, "World embedded interfaces for human-robot interaction," in *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*. IEEE, 2003, pp. 6–pp.
- [9] R. Nunez, J. Bandera, J. Perez-Lorenzo, and F. Sandoval, "A human-robot interaction system for navigation supervision based on augmented reality," in *MELECON 2006-2006 IEEE Mediterranean Electrotechnical Conference*. IEEE, 2006, pp. 441–444.
- [10] M. Sayour and M. Yassine, "Hac-slam: Human assisted collaborative 3d-slam through augmented reality," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.
- [11] Unity-Technologies, "Ros-tcp-endpoint," 2021. [Online]. Available: <https://github.com/Unity-Technologies/ROS-TCP-Endpoint.git>
- [12] IntRoLab, "rtabmap\_ros." [Online]. Available: [https://github.com/introlab/rtabmap\\_ros/tree/ros2](https://github.com/introlab/rtabmap_ros/tree/ros2)