Spot-On: A Mixed Reality Interface for Multi-Robot Cooperation

Tim Engelbracht tengelbracht@ethz.ch
Petar Lukovic plukovic@ethz.ch
Kai Lascheit klascheit@ethz.ch
Tjark Behrens tbehrens@ethz.ch

Abstract

Recent progress in mixed reality (MR) and robotics is enabling increasingly sophisticated forms of human–robot collaboration. Building on these developments, we introduce a novel MR framework that allows multiple quadruped robots to operate in semantically diverse environments via a HoloLens interface. Our system supports collaborative tasks involving drawers, swing doors, and higher-level infrastructure such as light switches. A comprehensive user study verifies both the design and usability of our app, with participants giving a "good" or "very good" rating in almost all cases. Overall, our approach provides an effective and intuitive framework for MR-based multi-robot collaboration in complex, real-world scenarios.

1. Introduction

Recent advances in MR and mobile robotics have opened the door to new possibilities in human–robot collaboration. Head-mounted displays, such as the HoloLens, have enabled intuitive remote operation of sophisticated robotic platforms, while concurrent developments in autonomous robot teaming have improved coordination among multiple agents. Existing work in the area - such as recent efforts on single-robot control using augmented reality headsets [5] or basic multi-robot teaming frameworks [2] - demonstrates the potential of overlaying digital cues on the physical environment to enable more intuitive command and coordination. However, relatively few systems address rich multi-robot collaboration and object-specific interactions in cluttered or semantically diverse spaces. Moreover, most existing approaches only incorporate simplified digital representations of the environment, thereby limiting the robots' ability to accurately localize, manipulate, and perform collaborative tasks that require an understanding of specific objects. Building on these insights, this paper presents a novel framework for controlling multiple quadruped robots in a shared workspace via a HoloLens interface. Semantic instance segmentation is applied to identify and distinguish different objects, while a scene graph encodes the relationships and affordances within the space. This digital clone of the environment allows operators to command robots to interact with various elements - ranging from drawers and swing doors to movable or more functional objects - with a high degree of autonomy. The semantic states of and connections between objects are updated within the scene graph while the robots operate and inspect the environment. Our approach extends the capabilities of previous systems [8] by enabling multi-robot collaboration on tasks such as object manipulation and joint inspection or activation of room infrastructure (e.g. light switches and lamps). In doing so, the proposed system demonstrates a more sophisticated paradigm for mixed reality–based control of multiple robots, one that is suited to complex and dynamic real-world scenarios. Contributions of this work include:

Figure 1. **Spot-On**. We introduce Spot-On, a Mixed-Reality application that allows Multi-Robot Control and Colaboration

- An interactive mixed reality interface for coordinated operation of multiple robots using a digital clone of the environment.
- A scene graph as the underlying environment model to enable robust object understanding and interaction.
- Mechanisms for advanced collaborative manipulation tasks (drawers, swing doors, dynamic drag-and-drop).
- Real-world demonstration of collaborative inspection tasks.

2. Related Work

We build our implementation on impressive recent advancements in core robotics tasks such as scene reconstruction [1, 2], instance segmentation [13–15], and object grasping [4, 9]. The Spot-Compose framework [8] integrates several of these techniques, providing a unified platform for robot-object interaction and manipulation. Specifically, it combines OpenMask3D [15] for open-vocabulary instance segmentation, AnyGrasp [4] for arbitrary object grasping, and a custom method for robot repositioning to facilitate object handling. Additionally, Spot-Compose employs YOLOv8 [6] for classifying drawers and handles and introduces a novel approach to estimate the axes of motion of drawers for automated opening. We extend this work by incorporating additional robotic capabilities and providing an intuitive and immersive mixed reality interface for remote robot control. HoloSpot [5] provides a compelling example of such an interface, implementing a digital twin of a scene in Unity and translating user inputs into robot commands via Spot-Compose. In doing so, HoloSpot facilitates object drag-and-drop functionality on the Microsoft HoloLens [10], allowing users to specify actions in mixed reality, which are then executed in the real world by the robot. While HoloSpot demonstrates robust capabilities in object picking and placing, it has two primary limitations. First, it does not account for relationships between objects within the scene, which are often critical for contextual manipulation tasks, such as operating light switches. Second, it supports only a single robot, limiting its applicability in collaborative scenarios. To address these limitations, we propose a novel mixed reality interface that incorporates elements of SpotLight [3], enabling robots to assess and modify the states of lamps and light switches (i.e., on or off) based on user commands. Furthermore, we develop a novel method for coordinating two robots within a shared scene to collaboratively execute pick-and-place operations and more complex object manipulation tasks.

3. Method

3.1. Scene Graph

While open-language instance segmentation [15] offers higher flexibility, we identified the resulting data structure (point cloud with open language features) as unsuitable for efficient and robust robotic downstream tasks. Compared to previous approaches [2, 5], we rely on an object-centric scene graph [1] that captures semantic relations between those instances, which inherently offers semantic scene understanding. Hence, we can effectively employ this scene graph for querying the scene via the the MR interface. Given the scene graph as a set of nodes and edges $G = (V, E)$, each node consists of a semantic label and 3D points, which we obtain by running a semantic instance segmentation algorithm [14] on the point cloud of the initially scanned environment. In addition to classical nearest-neighbor relationships, we employ YOLO detection algorithms to find drawer and light switch instances within the scene. Interactions of a robot as explained in 3.4 allow us to associate light switches with lamps, as well as to check the states of these lamps and drawers. To enable efficient transfer between our central compute unit and the HoloLens, we save the scene graph as a lightweight JSON object.

3.2. Communications

As shown in Fig. 3, our wireless networking infrastructure relies on a server for each robot. Through these servers, the robots both post their own state $x_{robot} = (x_{battery},\ p,\ \theta,\ x_{status})$ as well as changes to the state of an object $y_{object} = (y_{ID},\ y_{state})$ in the scene they caused by interactions. We denote $x_{battery} \in [0, 1]$ as the battery status, $p \in \mathbb{R}^2$ as the 2D coordinates in the map, $\theta \in [0,\ 2\pi]$ as the orientation and $x_{status} \in \{\texttt{IDLE},\ \texttt{BUSY}\}$ as the busy status. Furthermore, $y_{ID} \in \mathbb{N}_0$ corresponds to the object ID in the scene graph and $y_{state}\{0,\ 1\}$ to the binary state of the object. Binary states can be represented as $(0,\ 1)$, which differ in interpretations depending on the context:

- For a drawer: $(0,\ 1)$ corresponds to `(open, closed)`.
- For a lamp: $(0,\ 1)$ corresponds to `(off, on)`.

On the HoloLens side, each scene graph object queries both servers at 10 Hz for state changes of its physical counterpart. Once a change is posted by one of the robots, the scene graph on the HoloLens and thereby the scene in the user interface gets updated. In addition to state information, a job queue is hosted on the servers, routing user commands to the corresponding robot. To execute Multi-robot tasks (See 3.4), the robots can also send commands to each other and query their respective states.

3.3. Hololens App

Our app consists of two main parts, as shown in Fig. 4. First, a home menu housing the app's core functionalities, and second, a $3D$ scene for interacting with different objects. Upon start-up, the app will display a splash screen displaying the Spot-On logo, followed by a widget asking the user whether they would like to use our tutorial or not. After accepting, the user will be shown the collapsible helper window, followed by the main menu. If the user declines the tutorial, they are directed towards the main menu immediately. At this point, the voice command feature is also available. In the following, we will give in-depth explanations on the tutorial, home menu and scene view. Additionally, we present scene interactions and the voice command feature.

Tutorial We implement a short tutorial to help inexperienced users familiarize themselves with the app and its functionalities. For the inital release of the app, the tutorial consists of textual explanation for interacting with the MR interface and the exact functionalities of the app. The tutorial is arranged as a circular buffer of text prompts

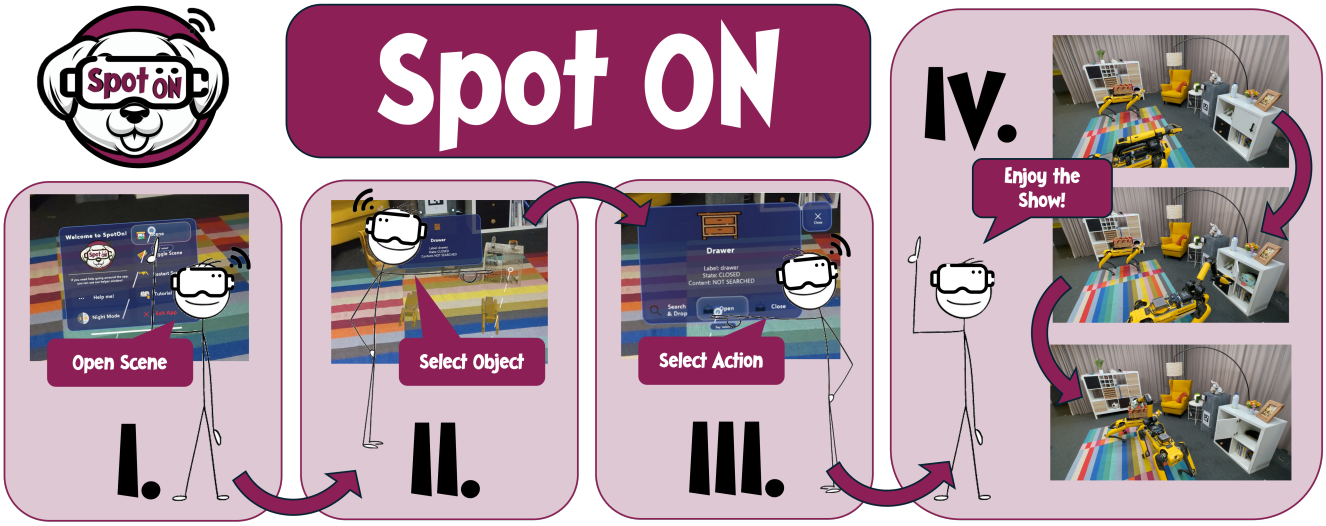


Figure 2. **Spot-On Overview.** Our user friendly top-level mixed-reality interface allows for intuitive robotic control. Starting from the main menu, the user opens the scene view (I.). The user is presented with an interactable 3D reconstruction of the scene. By hovering objects in the scene and clicking them, objects can be selected (II.). Now, object specific actions are highlighted and can be selected at will (III.). Once the action is selected in the virtual user interface, the robots automatically start performing the task in the real scene (IV.).

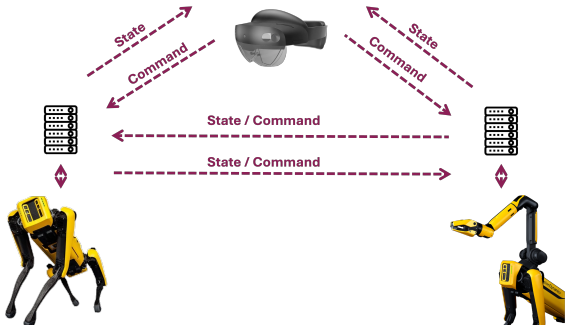


Figure 3. **Networking.** The networking infrastructure relies on one server per robot. The HoloLens posts commands for the robots to execute, while the robots post their states and changes they have made in the environment. We query these changes, so that the scene graph representation in the user interface stays up-to-date.

Figure 4. **Scene and Menu.** Figure shows scene (left) and start menu (right) as they are seen by user in the Hololens.

containing information about the following: general interaction, start menu, interactable scene elements, and voice commands. Here, all the functionalities are explained in detail. The user can activate the tutorial anytime later both from start menu and by using voice commands (See Fig. 4).

Home Menu Our main menu is shown in Fig. 4 (right) and Fig. 9. The home menu let the user access functionalities such as scene view or tutorial. A comprehensive list of functionalities is given in Tab. 1. All mentioned functionalities are also available through voice interface which will be explained later.

Scene View In the scene view, as in Fig. 4 (left), the user can interact with the scene, with both objects and robots. The user enters scene view by either clicking on the 'Scene' button in the main menu or by using the voice interface. The Scene view displays a 3D point cloud representation of a real, scanned environment (Sec. 3.1). Like all app widgets, the scene view follows the user's head movements. However, it is less sensitive compared to prompts to minimize users fatigue, particularly in the eyes and neck.

Interactions Once in the scene view, the user can interact with all robots and objects in the scene. We discern two basic interaction types: hovering and selecting. Whenever the user hovers over an interactable object, a temporary prompt will appear and show basic information about the currnet object. As soon as the user's pointer is outside of the object bounding-box, this temporary pop-up disappears. To select a hovered object, the user performs a clicking action. This will open a custom object widget which allows the user to execute the functions explained in section 3.4. A comprehensive set of custom object widgets is displayed in Fig. 5. The user may leave the scene view and enter the main menu by selecting any point that is not an interactable object.

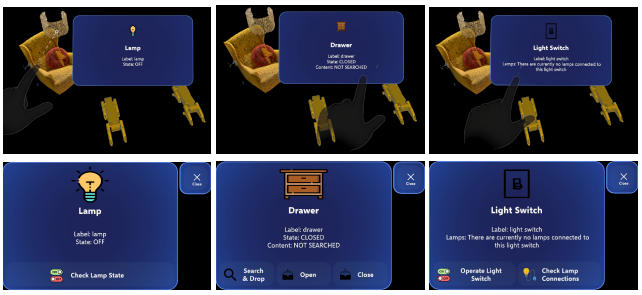


Figure 5. **Interface.** Figure depicts the app's response to user interaction by hovering and clicking on different object types. The first row shows hovering events (popups), and the second row shows clicking events (prompts). From left to right, interactions are with a lamp, drawer, and light switch.

Voice Interface An additional useful way of interacting with the app are the voice commands. We implement voice commands as a result of user feedback during our pre-study (see Sec. 4.1). Voice commands reduce the number of clicking interactions a user has to perform and thereby leading to a smoother user experience. Note that voice commands are used only to navigate the app, not to issue commands to the robots. The reason for this approach is to prevent accidental voice commands that could lead to unintentional physical action. Additionally, we do not allow critical voice commands such as exiting the app. Furthermore, while voice commands are permitted for certain important but less critical actions, such as restarting the scene, these actions require the user to confirm by clicking an additional prompt to prevent unintended operations. A comprehensive list of all voice commands is provided in Tab. 1.

Day/Night Mode As an additional functionality, we enable the switching between lower contrast day mode and a high contrast night mode. Modes are automatically activated depending on the time of the day, however, user can change mode using both start menu and voice commands.

Voice command	Description
✓ *"Show scene"*	Opens 3D scene
✓ *"Toggle scene"*	Toggles scene colors
✓ *"Open tutorial"*	Opens tutorial window
✓ *"Help me"*	Opens helper window
✓ *"Restart scene"*	Restarts scene to start pose
✓ *"Day/Night mode"*	Switches to day/night mode
"Close"	Closes current prompt
"Open menu"	Opens main menu

Table 1. **Commands.** Table displays list of available commands in a main menu and their corresponding voice commands. Considering that voice command list is longer, commands that have their corresponding menu button are indicated with a checkmark.

3.4. Robotic Collaboration for Multi-Robot Tasks

Agents Our method is deployed on two Boston Dynamics Spot robots. The first robot is equipped with an arm and gripper, whereas the other robot does not have grasping capabilities. In the following, we will refer to the robot with arm simply as "Fluffy" and the robot without arm as "Softy". Due to asymmetric capabilities, all tasks that require gripping actions are automatically delegated to Fluffy. To increase Softy's utility, we equip it with an RGB camera and a basket fixed to its back. This allows the robot to observe its environment and carry objects.

Tasks For colaboration we define a set of collaborative tasks that can be decomposed into a single robot tasks. Before elaborating on multi-robot tasks, we will first discuss single robot tasks. The user commands the robot to interact with a specific object through interactive buttons that appear upon hovering and clicking an interactable object in the mixed reality interface. After interaction, all state changes are posted on the server and the scene graph is updated accordingly. The tasks are listed as follows:

- **Drawer Interaction** For opening and closing of linear drawers, we utilize the Spot-Compose [8] framework. The drawer handle is detected using a fine-tuned YOLOv8 [6], while the axis of motion is computed as normal of the drawer plane.
- **Swing Door Interaction** Similar to the drawer detection, the handle is detected using a YOLO model. If the handle is detected to be eccentrically placed on the door, we compute a 3D swing trajectory, similar to [3].
- **Light Switch Operation** We have the agent operate light switches using the SpotLight [3] framework. Here, the robot first refines the 3D position of the light switch and thereafter predicts the affordance, i.e. the type of interaction necessary for operation. Lastly, it estimates a 3D motion needed for operation.
- **Grasp Object** As done in [8], object grasps are calculated using AnyGrasp [4], which computes possible grasps based on the 3D segmentation mask of the object.
- **State Check** Depending on the object type, one can assign states to the objects in the scene. To this end, we define a set of states for drawers $x_{drawer} = \{\texttt{open}, \texttt{closed}\}$ and lamps $x_{lamp} = \{\texttt{on}, \texttt{off}\}$. Taking the idea from [3], the agents capture images of the objects and utilize the GPT4 API [12] to infer the current state of the captured object.

Having defined single robot tasks, we will define multi-robot tasks as a coordinated sequence of sub-tasks as described in the single robot tasks. Combining a number of subtasks into a single multi-robot command leads to better usability since only one button has to be pushed. Similarly to single robot tasks, these tasks are sent to the agents via a button bound to a specific object. The multi-robot tasks are listed below.

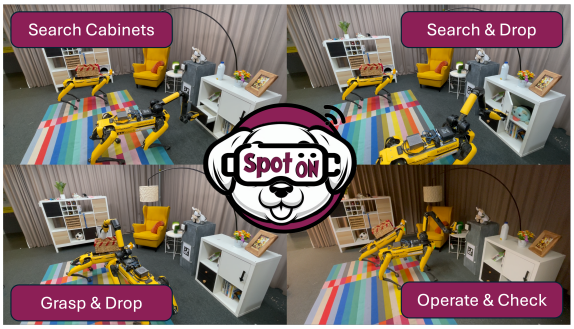


Figure 6. **Robots in action.** Spot-On offers a wide range of robotic interactions: Opening drawers (top left) and swing doors (top right), search cabinets, grasping objects and fetching it to the other robot (bottom left), operating light switches and checking lamp states (bottom right).

- **Fetch & Drop** This task is defined only for movable objects (e.g. bottles). As above, a grasp it estimated and Fluffy moves to the object and grasps it. The robot then drops the object into Softy's basket. To facilitate the drop, Softy lowers its base by crouching. Movements of both robots are synchronized via inter-robot communication.
- **Search & Drop** This task is defined for drawers and swing doors. Fluffy first opens the drawer and uses an open vocabulary object detector [11] to check its contents. If a queried object is found, the robot proceeds to grasp it and drop it into Softy's basket. Again, Softy facilitates the drop by crouching and moving closer to Fluffy.
- **Operate & Check** This task is defined for light switches within the scene. While Fluffy operates the light switch, Softy checks the states of all lamps in the scene. If Softy detects a state change in one of the lamps, a new lamp-light switch connection in the scene graph is created. The information is again relayed to the HoloLens and the scene graph gets updated with the newly found link.

Body Planning In order to interact with objects without colliding with the environment or the other agent, we implement a planning algorithm that takes the other agents as a static obstacle into account. We first approximate an upper bound of the robot's spatial extend as a cuboid and render it into the 3D map. Afterwards, we generate a 2D occupancy map by projecting the 3D map of the environment onto the xy-plane (including the cuboid, excluding the floor). Given the xy-coordinates of the interactable object, we first need to generate a suitable body position that is collision free and close enough to the object to grasp it. We do this by generating a set of xy-coordinates on a circle with a $r = 1$ meter radius around the object. We then select as a final position the xy-coordinates that are both closest to the current robot position in terms of euclidean distance and have a distance of at least 0.6 meters to other obstacles. Note that more complex algorithms for planning, such as RRT* [7], are not needed in the current setup, although they are implementable given our current map representation.

4. User Study

Real user feedback is crucial throughout the application's development, guiding it toward the final version that meets the needs of its end users. Therefore, we separate the user study into two parts. The first part is held at our demo event. We gathered user feedback at this event to improve the our software at that time, in order to implement the final version of the application. We shall refer to this part of the user study as the pre-study. The second and primary part includes the complete user study, during which participants test our application and provide qualitative feedback on its interface and functionalities. To assess intuitiveness of the interface, we conduct quantitative tests by measuring how quickly users navigate and familiarize themselves with the application. The final study is conducted on a wide variety of users, with different ages (68.8% aged 15 to 24, 31.2% aged 25 to 34) and levels of experience with MR software (68.8% without any experience with MR or VR). Our participants also come from different backgrounds, ranging from medical fields, business, computer science, food science and more.

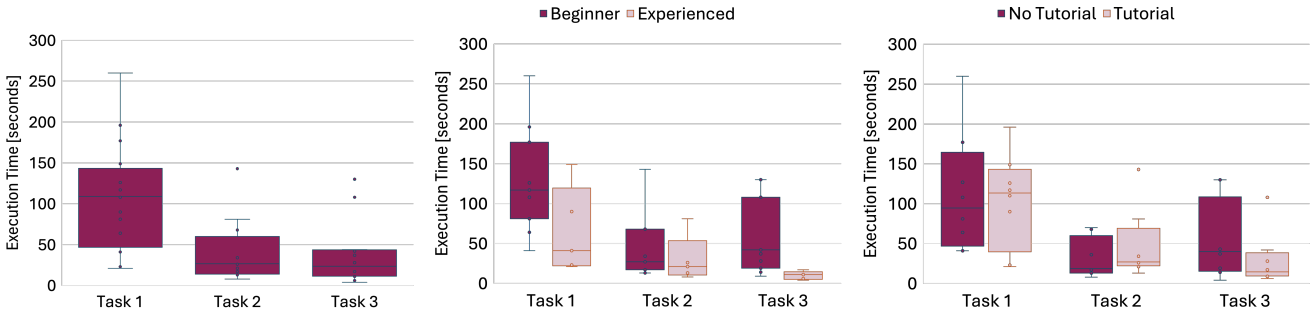
4.1. Pre-Study

Throughout the development process, we gather informal user feedback through a pre-study to guide our functionality and design choices. The purpose of this pre-study is to increase the number of functionalities in our application and therefore give later user study participants the opportunity to test a wider range of functions. During this pre-study, users are given the opportunity to test a basic set of functions in a real scene with two robots. This is intended to give user a real life scenario example.

Voice Commands A key insight of the pre-study is that users, especially beginners, struggle with general interaction with the scene, i.e. with clicking and hovering. Therefore, to enhance the usability of the app, we implement the voice interface explained in the voice commands Sec. 3.3. This voice command feature reduces the need for physical interaction with the MR interface during app usage.

Head-Following Widgets When designing a Mixed Reality application one has to decide on using either spatially anchored or head-following widgets (e.g. menus, message windows). While we opted for spatially anchored widgets during the earlier iterations of the design process, we now use head-following widgets in the final version of the app. This design change, driven by early user feedback, addresses the issue of users losing track of widget locations when shifting their gaze or body position. In the final version, scene, menu and prompts are in the user's focus at all times, further aiding intuitive usage of our application.

Further Functionalities Lastly, the pre-study provided valuable insights and ideas for additional app functionalities desired by users. As a result, we implement a tutorial



(a) General times (b) Beginner vs. Expert times (c) Tutorial vs. No Tutorial times

Figure 7. **Time Graphs.** The figure compares execution times for three tasks. Subfigures show $25th$ to $75th$ percentile boxes with median.

Task	Description
Task 1	*"Have the robots identify the connections between the lamps and the only light switch in the scene."*
Task 2	*"Have the robots open the large white swing door of the white cabinet."*
Task 3	*"Find the current battery level of the robot named Fluffy."*

Table 2. **Tasks.** Summary of tasks performed by participants. Each participant is given the description and has to localize the button in the user interface that leads to the desired outcome.

in the form of a helper window and introduce a night mode for low-light conditions. All of these newly added features will be assessed in detail in the following user study.

4.2. Main Study

During the main study, the users have the opportunity to test the final version of our application, including the final set of functionalities. They also get the chance to give feedback on the app's design choice and propose further improvements. We separate the main study into two parts: qualitative and quantitative analysis. This gives us feedback on design choices and intuitiveness, respectively.

4.2.1. Quantitative Analysis

In the first part of the main study, we first evaluate how intuitive the app is to users, and later investigate the utility of a tutorial as well as the impact of prior MR/VR experience. To this end, we measure the time it takes a participant to execute three tasks, which we explained to them beforehand. These tasks entail navigating the user interface until a button corresponding to a free-form prompt is retrieved and operated by the participant. We select tasks of arguably the same level of difficulty. All tasks start from a common point (the start menu), and time is measured from the moment participants switch to the scene view until they press the desired command button. The tasks are listed in Tab. 2.

General Evaluation Initially, we evaluate the time it takes the user to complete each task. As seen in Fig. 7a, the median completion time for task 1 is 109 seconds (mean: 108 seconds), while the time decreased significantly for task 2 (median: 27 seconds, mean: 39 seconds) and task 3 (median: 24 seconds, mean: 41 seconds). Assuming a comparable level of difficulty across all tasks, this indicates a steep learning curve, with the median completion time dropping considerably after just one task, highlighting the intuitiveness of our app.

Impact of Experience We also find it insightful to compare how previous experience with MR interfaces influences task completion times. As shown in Fig. 7b, experienced users completed tasks in 64.8, 29.8, and 10 seconds, with corresponding median times of 41, 21, and 11 seconds. In contrast, beginner participants required an average of 127.9, 42.6, and 54.9 seconds to finish each task, with median times of 117, 27, and 42 seconds. One can observe that the level of experience also has a notable impact on the learning curve, with experienced users demonstrating faster learning progress. For beginner users on the other hand, the learning process appears less straightforward and progresses at a slower pace.

Impact of Tutorial Lastly, we compare the impact of showing the tutorial on task execution times. As shown in Fig. 7c, the execution times do change significantly. Specifically, participants without prior tutorial need on average 112.4 seconds to complete task 1, compared to 104 seconds for those participants who viewed it. For tasks 2 and 3, these times are 30 seconds versus 46.5 seconds, and 52.6 seconds versus 29.1 seconds, respectively. Participants are evenly distributed across the tutorial and non-tutorial groups. We attribute these results to two factors: the tutorial provides only textual information without interactive practice, and execution times depend more on prior experience with MR devices. As both experienced and inexperienced users participate, some perform better without the tutorial, as seen in tasks 1 and 2. Thus, the tutorial offers limited performance improvement, as execution times are primarily driven by interaction skills.

4.2.2. Qualitative Analysis

The second part of our user study aims at improving our user interface and design. Participants are asked for their opinion on fundamental design choice. For this, we confront the participants with pairs of different designs. We also prompt the users to rate aspects of the app such as the overall intuitiveness and design of the app. Lastly, we pose a number of open box questions, in order to give the users the freedom to name any further suggestions or complaints.

Design Choices During this part of the study participants are given three design choices shown on Fig. 8, Fig. 9 and Fig. 10. Considering the color scheme of the app, the majority of users (69%) prefer the default (blue) color scheme over the purple Spot-On color scheme. A further design choice considers the transition from the main menu to the scene view. Here, the user has the choice of either using a "close" button separated from the menu or an integrated "Scene" button. A majority of 63% voted for transitioning to the scene using the integrated "Scene" button as shown in Fig. 10. Lastly, users are asked whether they find the night mode useful, as shown in Fig. 9. An unanimous 100% of participants affirm the utility of this feature.

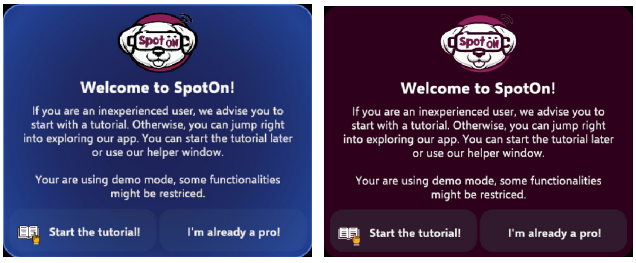


Figure 8. **Color scheme comparison.** Comparison of the two choices for main app color scheme.

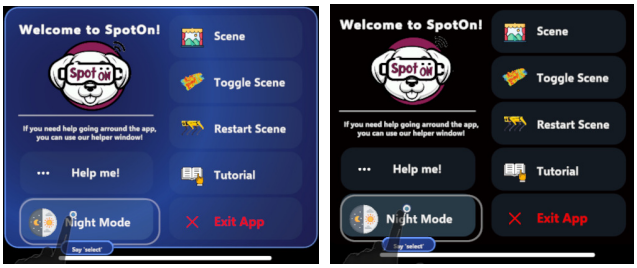


Figure 9. **Day/Night mode.** Comparison between night and day mode of our app.

Design Grading We prompt the users to rate different aspects of our app, such as overall visual appeal and user-friendliness. They are asked to give us feedback by giving grades on a scale of 1 to 5, with 1 being "very poor" and 5 being "very good". The exact questions and the corresponding mean scores are shown in Tab. 3. We find this feedback positive as most grades averaged above 4.2, indicating that most participants enjoy using our interface. We would like to highlight two underperforming areas related to robot visualization (Q3) and app responsiveness (Q5). The first issue stems from the robots being static in the scene, as they were not connected to real robots during the user study. The second issue involves the app's inconsistent response to interactions, which users report as being caused by failures in hand detection, particularly for users with smaller hands.

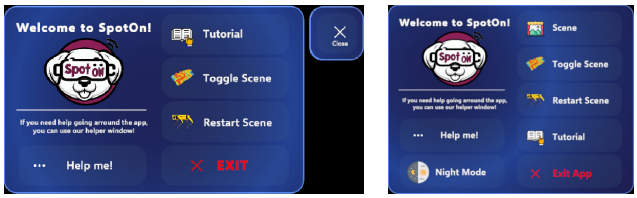


Figure 10. **Close and Scene button.** Comparison between two ways to transfer to scene view, close and scene button.

ID	Question Text	Mean Grade
Q1	How visually appealing was the interface?	4.31
Q2	How intuitive was the interaction with objects and buttons in the app?	4.25
Q3	Did the app respond accurately to your selections?	3.69
Q4	How useful was the on-screen feedback provided by the app?	4.25
Q5	How realistic were the robot visualizations?	3.56
Q6	How would you rate the speed of the app's responses to your inputs?	4.69
Q7	How engaging was the overall experience?	4.50

Table 3. **Questions and Grades.** This table shows text of each question presented to participants of our user study along with its corresponding grade.

5. Conclusion

We present Spot-On, a mixed reality interface for multi-robot control, hosted on the Microsoft HoloLens 2. The application was developed with the help of valuable user feedback to create an intuitive and immersive experience. Key features include a digital twin of the robots' environment, interactable buttons and objects, a voice interface and a tutorial for new users. Once an action is selected in the virtual scene, two Boston Dynamics Spot robots collaboratively execute the task in the real-world environment, with live visualizations provided in the app. For future work, we propose to implement an open-vocabulary voice interface for enhanced ease-of-use, and to enable on-the-fly environment mapping using an on-board depth sensor, eliminating the need for a manually configured digital scene.

References

[1] Tjark Behrens, René Zurbrügg, Marc Pollefeys, Zuria Bauer, and Hermann Blum. Lost found: Updating dynamic 3d scene graphs from egocentric observations, 2024. 2

[2] Jiaqi Chen, Boyang Sun, Marc Pollefeys, and Hermann Blum. A 3d mixed reality interface for human-robot teaming. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11327–11333. IEEE, 2024. 1, 2

[3] Tim Engelbracht, René Zurbrügg, Marc Pollefeys, Hermann Blum, and Zuria Bauer. Spotlight: Robotic scene understanding through interaction and affordance detection, 2024. 2, 4

[4] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains, 2023. 2, 4

[5] Pablo Soler Garcia, Petar Lukovic, Lucie Reynaud, Andrea Sgobbi, Federica Bruni, Martin Brun, Marc Zünd, Riccardo Bollati, Marc Pollefeys, Hermann Blum, et al. Holospot: Intuitive object manipulation via mixed reality drag-and-drop. *arXiv preprint arXiv:2410.11110*, 2024. 1, 2

[6] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, 2023. 2, 4

[7] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. 5

[8] Oliver Lemke, Zuria Bauer, René Zurbrügg, Marc Pollefeys, Francis Engelmann, and Hermann Blum. Spot-compose: A framework for open-vocabulary object retrieval and drawer manipulation in point clouds, 2024. 1, 2, 4

[9] Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Shafiullah, and Lerrel Pinto. Demonstrating ok-robot: What really matters in integrating open-knowledge models for robotics. In *Robotics: Science and Systems XX*. Robotics: Science and Systems Foundation, 2024. 2

[10] Microsoft. Hololens. https://www.microsoft.com/en-us/hololens. Accessed: 2025-01-04. 2

[11] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. Simple open-vocabulary object detection with vision transformers, 2022. 5

[12] OpenAI. Chatgpt api, 2025. 4

[13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 2

[14] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D: Mask Transformer for 3D Semantic Instance Segmentation. 2023. 2

[15] Ayça Takmaz, Elisabetta Fedele, Robert W Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. Openmask3d: Open-vocabulary 3d instance segmentation. *arXiv preprint arXiv:2306.13631*, 2023. 2