©SHUTTERSTOCK.COM/GOLDEN SIKORKA

Spatial Computing and Intuitive Interaction

Bringing Mixed Reality and Robotics Together

By Jeffrey Delmerico, Roi Poranne, Federica Bogo, Helen Oleynikova, Eric Vollenweider, Stelian Coros, Juan Nieto, and Marc Pollefeys

Digital Object Identifier 10.1109/MRA.2021.3138384
Date of current version: 14 January 2022

Spatial computing—the ability of devices to be aware of their surroundings and to represent this digitally—offers novel capabilities in human–robot interaction. In particular, the combination of spatial computing and egocentric sensing on mixed reality (MR) devices enables robots to capture and understand human behaviors and translate them to actions with spatial meaning, which offers exciting possibilities for collaboration between people and machines. This article presents several human–robot systems that utilize these capabilities to enable novel use cases: mission planning for inspection, gesture-based control, and immersive teleoperation. These works demonstrate the power of MR as a tool for human–robot interaction and the potential of spatial computing and MR to drive future developments.

1070-9932/22©2022IEEE

Digital Representation

Since the term was first coined almost 20 years ago [1], the rise of virtual, augmented, and MR technologies has highlighted the importance and applications of spatial computing as a novel paradigm for interacting with spatial information through our devices. Spatial computing refers to the digitization and modeling of the device's environment and the objects within it such that the device has spatial context. What distinguishes MR from virtual reality is the ability to simultaneously observe the physical and digital worlds, with digital content aligned to the real spatial environment, and the ability to interact with physical and digital objects. While augmented reality (AR) devices offer some of these capabilities through touchscreens, MR is characterized by more immersive visualization and interaction through head-mounted displays.

In the world of mobile robotics, spatial computing is a requirement for most operations. To navigate an environment, avoid obstacles, and perform useful functions, mobile robots need to build and use a digital representation of their understanding of the area. Often, this takes the form of a map, where the robot maintains an estimate of where it is in space, as well as the structure of the environment. More broadly, this representation can be a digital twin of the environment, where any data about the world that has spatial meaning can be embedded in a digital framework intended to capture information about the space and the digital devices in it as accurately and completely as possible. The trend in devices taking advantage of spatial computing is often called the third wave of personal computing, after the first two waves, where desktops provided access to home computing before mobile devices initially made this power ubiquitous and portable, although the devices lacked awareness of the space they were in. Mobile robots and MR devices thus have significant synergies since they require many of the same spatial computing capabilities, regardless of whether they are designed to be used by a human or behave autonomously.

This alignment of spatial understanding between human-oriented MR devices and robots provides an opportunity to unlock new modes of interaction between people and machines. In particular, by sharing these spatial representations and digital twins among human and robotic devices, we can enable all the agents involved to leverage that information for greater capabilities. This enables them to do more and provides a common understanding so that humans and robots can work together more efficiently through natural and intuitive interaction and collaboration. Additionally, by leveraging the egocentric sensing and immersive visualization of MR devices, we can provide embodied teleoperation experiences for remote devices. In this scenario, the user projects his or her actions to a remote robot while using the robot's spatial understanding to provide immersive feedback about the task at hand. In this article, we aim to show our efforts in this domain, with three works that illustrate how MR can enable more flexible collaboration and even new types of interactions between humans and robots.

Sharing Spatial Information

We first consider a scenario in which devices share spatial information in a temporally decoupled fashion. Here, we demonstrate that spatial data can be persisted through time by defining them with respect to world-locked reference coordinate systems. The general workflow begins with one device defining a reference coordinate system by building a visual map of the environment. Spatial information is then anchored to this reference frame, which is fixed with respect to the space, enabling it to persist in that location. Another device, or the first device at a later time, can then relocalize to this map, recover the reference coordinate system, and access the stored information in the same place in the world where it was defined.

In this section, we show how this type of workflow can be utilized to enable the planning of robotic inspection missions. This work is motivated by the need for automated inspection in many commercial and industrial settings, where mobile robots have the navigation capabilities to actually execute the missions but where planning the path that a robot should take is still a cumbersome process. In these large, dynamic environments and in unconstrained spaces, such as disaster zones, it is often not possible to augment the environment with fiducial markers to facilitate the sharing such spatial information from the planner to the robot. We therefore propose to use shared, world-locked coordinate systems through MR as a way to provide a common spatial reference.

Existing commercial solutions for mission planning either use a computer interface that is decoupled from the environment or require the user to drive the robot through the trajectory first, in a teach-and-repeat fashion. Planning a robot trajectory within a high-fidelity 3D model or mesh of the environment using a computer is certainly possible and currently deployed in commercial services [e.g., Rocos (https://www.rocos.io/content/robotics-fleet-management)]. However, this approach requires the model to contain a high level of detail about the inspection targets to accurately indicate the area the robot should observe without actually being there. This type of approach has been used for ground robot [2] and drone mission planning in mixed [3] and virtual reality [4] by using 3D terrain models, which are far more available than high-resolution digital twins of inspection sites for ground robots. Additionally, AR [5] and MR [6] have been used with a virtual robot arm to facilitate trajectory programming, but connecting these simulated arms to a real robot has used fiducial markers rather than colocalization with any spatial context.

Unlike these prior works, utilizing the spatial context of the environment provides a more natural way to define a robot trajectory. However, while solutions such as Boston Dynamics' Autowalk feature for Spot enable the user to plan missions in context while physically present in the space, such a teach-and-repeat approach requires the user to manually control the robot through the environment every time he or she defines a mission, and it is not possible to edit missions without rerecording them. While this is a powerful feature, it

still has a strict requirement that both the user and the robot are present and available every time a mission must be defined. The inability to change saved missions also leads to a situation where missions should be repeatedly executed for the process of defining them to be worth the effort. Otherwise, the robot could be used in teleoperation mode with the same level of efficiency.

Mission Planning

We propose to use MR as a tool for defining inspection missions in context, without the requirement of first teleoperating the robot through the desired trajectory. This concept is illustrated in Figure 1(a). It is made possible by using a common coordinate system that persists in space and through time to share spatial information about the mission among devices. In particular, we leverage the cloud-based localization service Azure Spatial Anchors (ASA) to create such a reference coordinate frame (and enable localization to it by other devices) and spatial data persistence. In this system, a human user with a HoloLens defines an inspection mission by placing holographic markers to indicate desired poses for the robot. At a later time, an autonomous robot can localize to the spatial anchors that were placed during mission creation; obtain the set of waypoints and inspection poses, which are defined relative to these anchors; and autonomously execute the assignment. This approach has the advantage that the robot, whose availability is likely a bottleneck in current work environments, is required only for executing the mission, not for defining it. Furthermore, these missions can also be edited in MR, without the need to recreate an entirely new trajectory every time the tasks need to be adapted.

In practice, the workflow for this system proceeds as follows. The HoloLens user moves around her environment, manipulating holograms representing inspection waypoints. These waypoints can be created, deleted, and connected in an arbitrary graph structure, including branching. Since localization to spatial anchors depends on observing the same part of the environment where the anchor was created, anchors are automatically fashioned as the user moves away from existing ones, ensuring that waypoints are always defined with respect to an anchor to which they are in close proximity. We use a radius of 2.5 m from previous anchors as the threshold for creating a new one, which was empirically determined. Anchor localization accuracy and recall degrade beyond 4–5 m, but an appropriate value here would depend on the structure and appearance of the environment. Once the user is satisfied with the waypoints, the mission can be saved by serializing it and storing these parameters in a cloud-based database. The user experience during mission planning is shown in Figure 2(a), where the waypoints are represented by spheres, with camera frustums for the orientation of the inspection pose and coordinate axes representing spatial anchor reference frames.

When the robot is ready to execute the mission, it obtains the serialized parameters from the database, localizes to the spatial anchors, and then proceeds through the waypoints. If the HoloLens user is colocalized to the anchors and running the app, she can monitor the progress of the mission in MR, with an articulated model of the robot overlaid on its pose in the real world [see Figure 2(b)]. Branching decisions at interconnected waypoints can be made through application programming interface (API) calls based on the results of the inspection and by the user selecting the desired branch if the mission is being executed in this interactive mode with the HoloLens. The structure of a mission is conceptualized in Figure 1(b).

ASA

ASA is an MR cloud service designed for localization. The fundamental concept of the service is that a small visual map,

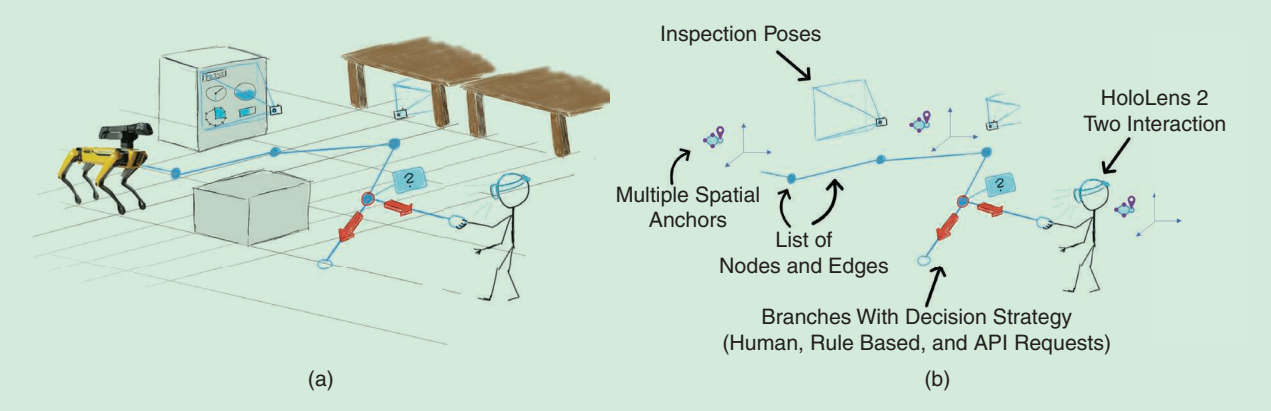


Figure 1. The (a) mission planning workflow and (b) mission components. A human user with a HoloLens device moves through the environment to be inspected and places holograms that represent waypoints defining a trajectory. Inspection poses can be defined to trigger the robot to capture data at regions of interest. The underlying structure of a mission is a list of nodes and edges representing a graph of waypoints, with branching decisions in the graph handled by one of several strategies. As the user moves through the space and places waypoints, Azure Spatial Anchors are automatically placed to cover the trajectory so that each robot pose can be defined with respect to a nearby reference coordinate system. This mission structure is then serialized in a JavaScript Object Notation format so that it can be retrieved from a database by the robot and executed autonomously after localizing to the spatial anchors in the mission. API: application programming interface.

created through structure from motion and representing a specific location in the world, is stored in the cloud with a unique ID. The set of features and descriptors constituting this map then define a coordinate system that is fixed to the world; if those features can be detected later and their 3D positions can be recovered, the coordinate system can also be recaptured. Devices wishing to localize to a particular anchor, known by its unique ID, then create their own structure-from-motion map, which they submit to the cloud service as a query. If correspondences for the feature points in the query can be found in the reference map in the cloud, the service returns a relative pose.

This service is cross platform and relies on a device's onboard visual odometry to build these visual maps to create and query an anchor. On HoloLens devices, the onboard head-tracking processes handle this. For mobile devices, ARKit and ARCore provide the image features and poses for iOS and Android, respectively. On robotic systems, a special research version of the software development kit (SDK) was released with Ubuntu Linux support as well as a Robot Operating System (ROS) wrapper (https://github.com/microsoft/azure_spatial_anchors_ros). Unlike other devices, where the visual pose estimation happens in the context of a platform-specific odometry system (e.g., an ARCore or ARKit session), pose estimation solutions on mobile robots can be quite diverse. The Linux version of the SDK thus accepts undistorted images and poses and computes its own features. The pose of the camera can be directly estimated through visual odometry on the robot and by attaching the camera with a calibrated transformation to a mobile base that estimates its pose through some other means (e.g., with lidar). While the success of localization queries to ASA depends heavily on the particular cameras, trajectories, and environments in question, in practice, the performance is in the range of centimeter-level accuracy. This is documented in Figure 2(b), where the robot's estimate of its pose with respect to the anchor closely matches the actual robot pose when it is transformed into the HoloLens's field of view, as seen in the close alignment of the robot and its holographic model.

System Overview

The system is organized in a modular and distributed fashion, with components deployed on the HoloLens, on the robot, and in the cloud. It is possible to run the HoloLens and robot at the same time, where the mission parameters and some visualizations are sent between the devices with ROS, and to define and run the mission at different times, in which case the mission definition is serialized and stored in a databse in the Azure cloud. An overview of these components and the flow of data among them appears in Figure 3.

The HoloLens runs an application that is built using the Unity 3D engine (https://unity.com/). Some prebuilt components from the MR Toolkit (https://github.com/microsoft/MixedRealityToolkit-Unity) handle the user's interaction with the app's holograms. The ROS# (https://github.com/siemens/ros-sharp) package acts as a bridge between the Unity environment and ROS communication framework. HoloLens provides support for creating and querying ASA through an SDK, and this functionality is connected to Unity through an XR plugin layer. These components work together to enable the user to create and move holographic markers in the environment and store the poses of these waypoints with respect to spatial anchor coordinate systems.

On our robot, a Boston Dynamics Spot, we connect a companion computer (an Intel NUC running Ubuntu 18.04) through the robot's ethernet port to run our user code. This code includes a wrapper for Spot's Google Remote Procedure Call-based SDK (https://github.com/boston-dynamics/spot-sdk) that connects its data streams to ROS topics (https://github.com/clearpathrobotics/spot_ros) as well as a corresponding bridge node that receives messages from the ROS# component in the HoloLens Unity app that are sent over a

Figure 2. Mission planning and execution. (a) The view through the HoloLens as the user moves holographic markers for the trajectory waypoints. Each waypoint is represented by a sphere for the desired position and a camera frustum to define the orientation the robot should take for inspection. The axes to the right of the trajectory mark the origin of an automatically created spatial anchor coordinate system. (b) The user's view as the robot executes the mission, with a robot model overlaid on the physical one and a sample data capture at an inspection pose. This user view during mission execution is only for illustration, as the robot can autonomously execute the mission without human intervention.

WebSocket. An additional ROS package of utilities (https://github.com/EricVoll/spot-mr-core) helps to interface the Spot with the ASA service. The robot's behavior is orchestrated by the ASA ROS commander node, which obtains the mission definition from either ROS or the cloud database. Once the robot localizes to the spatial anchors defined in the mission using the ASA service, this node sends the waypoints as trajectory commands through the wrapper to the Spot SDK and monitors the robot's progress through the mission. An important note is that this mission planning system relies on the robot's underlying navigation capabilities to actually plan and execute paths to the waypoints that are sent as goals.

Finally, the cloud components consist of the ASA service (see the "ASA" section) and a CosmosDB database to store serialized mission parameters. The waypoint poses, spatial anchor IDs, and other mission information are serialized into JavaScript Object Notation (JSON) format and stored in the database with a unique ID as a key. When operating in the temporally decoupled mode, the ASA ROS Commander node can look up a particular mission in the database, download the JSON, and then execute the mission once it is localized to the mission's anchors.

Outlook

This system represents a milestone in spatial computing and interaction. By sharing common reference frames, the MR device can provide the robot with actionable spatial information that the robot can understand in its own spatial context, leading to an improvement in efficiency in common commercial and industrial robotics tasks. One limitation of the current spatial anchors service is that the anchor maps are not automatically connected in the cloud to form large-scale, continuous digital twins even if they are located in the same area. This use of discrete spatial anchors limits scalability for this application, but future work on using continuous digital twins for the localization of multiple heterogeneous devices will enable this type of shared spatial understanding on a large scale.

Colocalization and Interaction

The colocalization of devices requires that they are able to localize themselves to a common reference coordinate system. Through their individual poses with respect to this common coordinate frame, the relative transformation between localized devices can be computed and used to enable new behaviors and collaboration between devices. In the scenario described in the "Sharing Spatial Information" section, the robot and MR device do not directly interact and are not necessarily colocalized at the same time, but they share spatial information that is anchored to a common spatial reference frame. Here, we consider a scenario in which multiple devices simultaneously colocalize to a shared coordinate system, thereby enabling temporally synchronized behaviors and interaction.

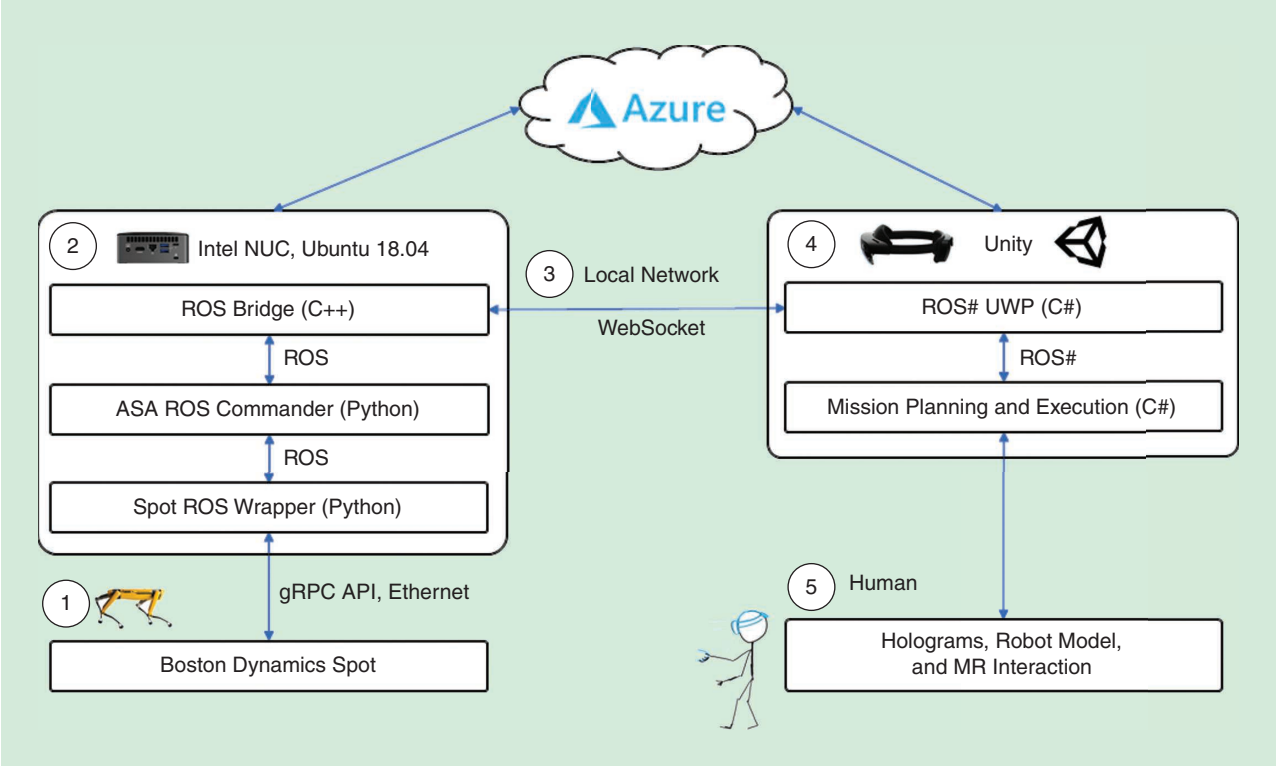


Figure 3. The components of the mission planning framework. (1) The robot communicates through its SDK to an ROS wrapper and our mission planning orchestration node, which (2) run on a companion computer. This computer communicates with the HoloLens across (3) the local network and through serialized missions stored in an Azure database. The HoloLens interfaces with the ROS communication framework using (4) an ROS# Unity plugin to the mission planning application, which ultimately provides an interactive MR experience for (5) the user. NUC: Next Unit of Computing; gRPC: Google Remote Procedure Call; UWP: Universal Windows Platform.

When the colocalized devices are all mobile robots, swarming and collective behaviors could be possible, as could the parallelization of spatial tasks. On the other hand, if there is a mix of human-centered devices (e.g., MR headsets and mobile devices) and robots, colocalization can unlock more natural human–robot interaction. This section describes a system that exploits the colocalization of a human wearing a Microsoft HoloLens 2 and a mobile robot to demonstrate intuitive hand gestures for robot control. This work is motivated by a need for semiautonomous behaviors with shared control as a way to reduce the attentional load for the operator. Defining high-level tasks for the robot to perform autonomously, particularly through intuitive interfaces, enables the user to focus on other objectives and control multiple devices simultaneously. This is desirable in search-and-rescue environments [7] and will be important in the increasingly robot-filled workplace of the future.

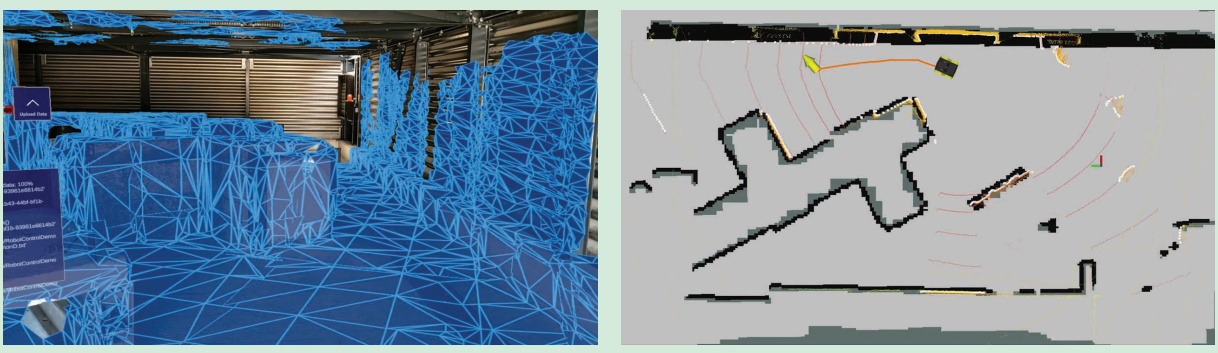
Colocalizing the Human and Robot

We consider two approaches for colocalizing devices to a common reference frame: sharing a map and utilizing a visual localization service in the cloud. Prior work in this domain has explored the use of AR for more efficient human–robot interaction and visualization of the robot's state and intent. However, these works have relied on colocalization through the detection of landmark objects [8] and the use of fiducial markers mounted on the robot [9]. While these approaches for colocalization are sufficient to perform AR visualization, they do not provide further spatial understanding for the AR device or robot. Here, we propose to share not just a relative pose but a shared map between the devices, enabling both devices to take advantage of a common digital twin of the space.

To share a map from the HoloLens with the robot, we execute an offline procedure to create and then convert a map of an environment for colocalization. We leverage the onboard spatial mapping processes of the HoloLens, which constantly build a sparse visual feature map for tracking the motion of the device as well as a dense mesh of the environment. The user observes the environment with the HoloLens depth camera to build the dense mesh, and our application provides visual feedback to show areas of the space that have been mapped [see Figure 4(a)]. The sparse map is aligned to the dense representation, and it enables the HoloLens to relocalize to the environment in a future session.

Once the spatial mesh has been captured, we apply several processing steps to convert it to an occupancy grid representation that can be used by the robot for lidar-based localization. We take the mesh as input, which typically consists of several connected components, and apply Poisson reconstruction to make it watertight. This watertight mesh is used to initialize a signed distance function (SDF) representation of the space. Finally, a horizontal 2D slice from the SDF is extracted at a user-defined height such that the implicit surfaces in the SDF define the occupied cells in the occupancy grid representation, and the voxels in the SDF with positive distance represent free space cells. An example of such a map being used for lidar localization is provided in Figure 4(b). The *xy* origin of the map is preserved during these conversion steps, so 2D coordinates given in the HoloLens coordinate frame correspond to the same *xy* position in the plane where the robot is navigating. In the shared map scenario, when the HoloLens and the robot localize to their respective maps, spatial information can be translated between the two map representations.

We also demonstrate the use of the ASA cloud localization service to colocalize the HoloLens and robot to the same anchor. This service was described in more detail in the "ASA" section. For this application, we require a single anchor that both devices can localize to, which then provides a common reference frame. In our workflow, the HoloLens observes the environment and creates an anchor through the service. We synchronize the unique ID of the created anchor with the robot through ROS, then leverage the ASA ROS wrapper to query this anchor by using a stream of camera images and poses from the robot. Once the robot localizes to the desired anchor, the anchor's reference frame is added to the ROS *tf* tree, where it can be used



(a) (b)

Figure 4. The colocalization of a robot and HoloLens through a shared map. (a) The user's view of a spatial mesh, which was captured from the HoloLens, overlaid on the real world. (b) This map is converted to a 2D occupancy grid representation, whose coordinate frame is aligned with that of the mesh, to enable robot localization with lidar.

to transform spatial data to or from the robot's other reference frames. For the purposes of our system, these approaches are interchangeable. They both provide the devices with a common coordinate system to share spatial information, but the shared map approach is tailored to robots equipped for lidar navigation, while the ASA approach is designed for robots using visual navigation.

Gesture Recognition

In the colocalized scenario we consider here, egocentric sensing by human-oriented devices can give spatial meaning to human motions, expressions, and gestures. We focus on detecting and classifying hand gestures with a HoloLens 2 headset and translating these into navigation commands for the robot. The HoloLens 2 features a variety of cameras for sensing the environment and actions of the user and estimating how the device moves through space. In particular, we leverage its hand tracking capability, which uses the depth camera to track the user's articulated hands and exposes the joint positions through an API.

We implement our gesture classification model as a neural network. Namely, we use a multilayer perceptron (MLP) [10], [11], which takes as input HoloLens hand tracking data and outputs probabilities across a set of predefined gesture classes. As for the hand data, we assume that only one hand is in view (or we consider the right hand if both are in view) and extract local angles for a total of 19 joints. In our experiments, we found that using only joint flexion angle values is enough to obtain an accurate gesture classification. We therefore parameterize each joint by a single value. Relying on single-frame predictions can lead to noisy classification results. To improve robustness, we leverage temporal information, running the model through time windows of 12 frames (which correspond to roughly 0.2 s with the app running at 60 frames per second): to classify a gesture at time t, we consider all the hand joint angle values tracked within the interval $[t, t - 12)$, flattening them into a (12×19)-dimensional vector that constitutes our MLP input. For the output, we consider three main gesture classes for human–machine interaction: "stop," "come here," and "point." We also add a background class to identify frames in which the user is not interacting with the robot and therefore not performing any gesture. This gives us a total of four classes.

The MLP network has a total of four layers and uses rectified linear units (ReLUs) [10] as activation functions. All hidden layers are 128-dimensional. Per-gesture confidence values are obtained by applying the sigmoid function to the output of the last layer. Finally, the gesture with the highest confidence is chosen as the classifier output. Figure 5(a) exemplifies the app output: the first line describes the output gesture ("stop"), which is the one obtaining the highest confidence value (0.62). We found that adding an attention layer [12] right before the classification one helps capture spatial and temporal correlations among joints and therefore yields more accurate results. The network produced good results when trained on a small training set: we asked six subjects to twice perform each of the three gestures (plus random actions for the background class), recording the corresponding hand tracking data with an ad hoc app. We trained our model on data from five subjects, withholding one subject for validation. We implemented our MLP network in PyTorch and reimplemented it in C# to perform inference in Unity.

Robot Control With Holograms and Gestures

In this system, the HoloLens acts as a human–robot interface for giving the robot navigation commands. The user can select an arbitrary navigation goal for the robot by moving a holographic marker to a location in the environment and confirming this as the robot's target pose by clicking a holographic button [see Figure 5(c)]. Alternatively, intuitive hand gestures (see the "Gesture Recognition" section) are detected on the HoloLens and translated to navigation commands for the robot. Figure 6 illustrates the components of the system and flow of data among them.

In the case of a navigation goal set by the holographic marker, the target pose is selected in the reference frame of the HoloLens and transformed to the common reference frame of the map that is shared between the robot and HoloLens so that

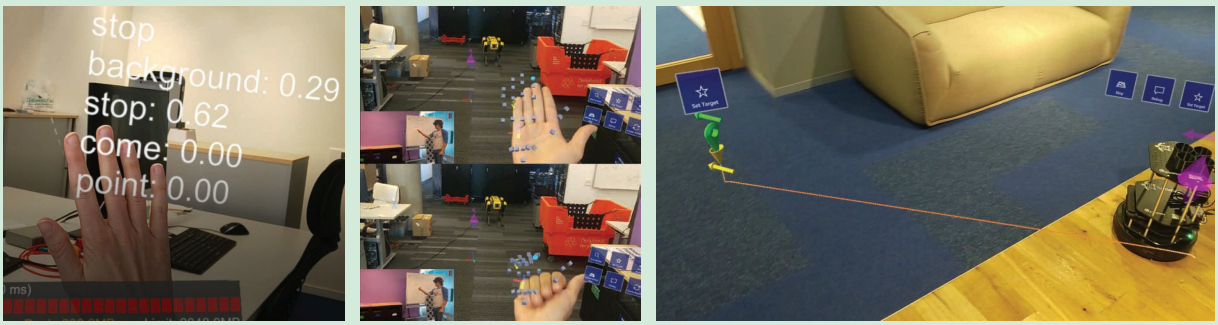


Figure 5. Human–robot interaction through hand gestures and navigation goals. (a) The user makes the "stop" gesture, which the classifier correctly identifies. This gesture is mapped to an action that preempts and cancels an existing trajectory on the robot. (b) Another gesture that is recognized is "come here," which triggers the robot to plan a path to the position of the user, which it knows due to colocalization. (c) Finally, the robot can provide visual feedback of its intent, as seen by its planned path.

the robot can interpret that goal in its own reference frame. The "come here" gesture in Figure 5(b) leverages the shared spatial understanding of the two devices to provide the semantically meaningful action that corresponds to the gesture. When the gesture is detected, the "here" in "come here" is translated to be the HoloLens's location in the shared map, and thus a goal pose in front of this location is sent to the robot. The "point" gesture works similarly in the sense that when it is detected, the pose of the HoloLens and the user's hand position while making the gesture are used to determine the position they are pointing to, and this is sent as the goal. The "stop" gesture is employed to preempt execution of the current trajectory if the robot is already moving toward a target pose.

We use ROS for communication between the HoloLens and robot, with ROS# (https://github.com/siemens/ros-sharp) providing the interface between the Unity-based holographic app running on the HoloLens and the other ROS nodes. The design of the system is robot agnostic and relies on the robot's own navigation stack to actually execute any motion commands that are generated from the HoloLens. For example, we demonstrated this system as a human–robot interface for controlling a Turtlebot 2 and a Clearpath Jackal by sending `PoseStamped` messages from the HoloLens to the robot's *move_base* node [see Figure 5(c)]. We also controlled a Spot with the same system by passing the target pose for the robot to the Spot SDK's (https://github.com/boston-dynamics/spot-sdk) `trajectory_command` function and enabling its onboard navigation to move the robot to reach the goal.

An important aspect of using an MR human–robot interface is that spatial information can be shared in both directions—from user to robot and from robot to user—within the environmental context. In the case of this system, spatial information shared by the robot takes the form of a marker for its position and a holographic line representing the planned path to reach its current target pose. The ability to simultaneously visualize the robot's state and intentions with the surrounding environment provides a clear advantage in terms of human–robot safety. When the HoloLens user can see the path that the robot is following overlaid on the real world, she can understand the robot's intent more clearly than with colored lights or a 2D interface on a mobile device. We envision safer and more efficient human–robot collaboration through this type of feedback, where visualization of the robot's intent can fill in the gaps left by body language and verbal communication that make human–human collaboration more straightforward.

Outlook

We demonstrated that gesture control for robots can be more intuitive when the human and robot share a spatial context and gestures can have spatial meaning. This is made possible by the egocentric sensing of the HoloLens and a shared map representation among the devices. However, the gesture detection system proposed here captures only a small set of commands. There are many opportunities to expand these types of spatially relevant gestural interactions when the robot and human are colocalized through a shared understanding of their space.

Immersive Teleoperation and Embodiment

In the previous section, we discussed gesture-based interaction in a shared environment. Here, we remove the shared environment and understanding of space and explore the projection of the user's actions to a remote robot and the robot's sense of space back to the user. We consider several levels of immersion based on touching and manipulating a model of the robot to control it and the higher-level immersion of becoming the robot and mapping the user's motion directly to it. The works described here focus on immersive teleoperation of robotic arms, where challenges include the

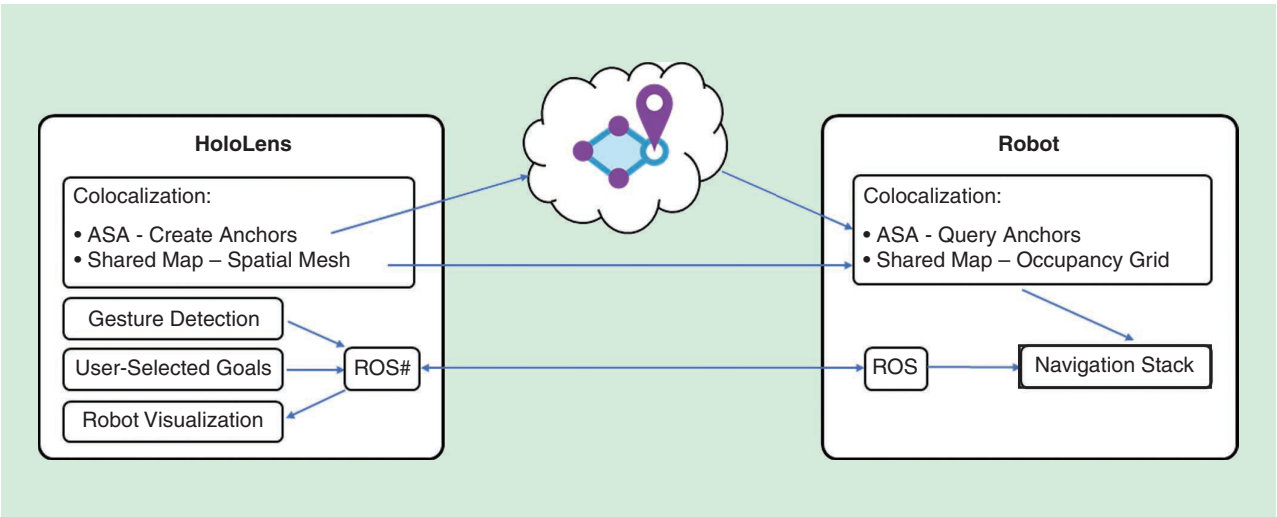


Figure 6. The interaction framework. Colocalization between the HoloLens and robot is achieved either by both devices localizing to the same ASA or by creating a map on the HoloLens (a 3D mesh) and sharing it with the robot in a representation that it can use (an occupancy grid). The devices communicate spatial information (navigation goals, robot states, and paths), which is defined with respect to their shared coordinate system via ROS. Human–robot interaction is achieved through intuitive gesture recognition and by placing holographic markers to give target poses to the robot.

differences in structure and articulation between the human and robot and the lack of proprioception in teleoperation. We present several novel approaches that address these challenges through MR: embodied teleoperation, motion retargeting, and task autocorrection. In each, an MR device provides a way to collect multimodal inputs (e.g., hand poses) and provide immersive visual feedback to the user. All these approaches are motivated by a desire to improve the ease of use and user experience in controlling a robot to perform manipulation tasks.

A basic level of proficiency in teleoperating a robotic arm, for example, using a joystick, may be reached quite quickly. But achieving a mastery of performing complicated actions can require a significant time investment, particularly if the arm has many degrees of freedom. It is mentally taxing, as the operator must internally compute transformations and rely on muscle memory to perform even simple manipulation tasks. One alternative approach for controlling an arm is kinesthetic programming, where the operator directly touches and moves the arm to record motions and later replay them. This is impossible to do from afar, and sensory disabilities and limited user mobility may also make it impractical to program the desired motions in person. Instead, immersive teleoperation enables us to manipulate a digital twin of a remote arm by holographic touching [see Figure 7(a)]. Using a HoloLens 2, an operator grabs the end effector of dual-armed ABB YuMi hologram by pinching and repositioning its hologram. The underlying system simply solves an inverse kinematics (IK) problem in real time and can stream joint angles to a remote YuMi.

Another emerging approach for immersive teleoperation is via embodiment, where the robot is treated as an avatar of the operator. In this case, the operator possesses the robot by transferring her motions to it and optionally viewing the environment from its perspective. An example of this using the same YuMi/HoloLens application appears in Figure 7(b). In this case, the palms of the users are tracked by the HoloLens, and their positions are set as IK targets for the end effectors of the YuMi. A pinching gesture also closes the gripper.

A notable demonstration of embodiment that leverages tactile feedback is the Tactile Telerobot (https://www.convergerobotics.com/), a system that includes a robot hand with multiple sensors mounted on a robot arm, which is operated using a tactile feedback glove. Other recent approaches leverage different combinations of tracking methods, head-mounted displays, and robotic systems. For example, a recent vision-based system, DexPilot [13], was used to teleoperate a robotic hand–arm system by observing human hand via four red–green–blue–depth cameras.

Ultimately, our goal with regard to the embodiment approach is to enhance the sense of body ownership, that is, the ability of the operator to comfortably control the robot as if it were his or her own body. Robots are still far from true body ownership, with significantly lower-than-human dexterity, tactile sensing, and feedback that can not mimic reality well, and they have engineering limitations, such as high latency and errors in tracking. We consider some of these fundamental challenges as we explore the mapping of human inputs to robot outputs with motion retargeting.

Motion Retargeting

Central to embodiment is the question of mapping human motions to an avatar. The difficulty lies in the vast differences between human and robot proportions and morphology. Two of the most commonly used approaches are joint angle mapping and IK. Angle mapping plots the operator's joint angles directly to the avatar's joint angles. Naturally, this is feasible only when the avatar has similar morphology to the human. While motions mapped with this approach may appear plausible, they lack some semantics. For example, Figure 8(a) depicts a joint angle mapping of a human hand to a robot hand. The human hand is posed such that it is touching a finger to a thumb, but this pose is not captured by the robot hand because of these differences in morphology. Alternatively, the IK approach finds joint angles to map the avatar's hands, fingertips, and end effectors relative to the operator's hand and fingertips [see Figure 8(b)]. In this case, the operator has more control over the exact positioning of the fingertips, but the hand pose itself might appear very different. This could be an issue when, for example, the operator wishes the avatar to make a specific hand gesture.

Several alternative approaches for retargeting have been proposed with the intent of making retargeting more effective for performing tasks, thus also improving body ownership.

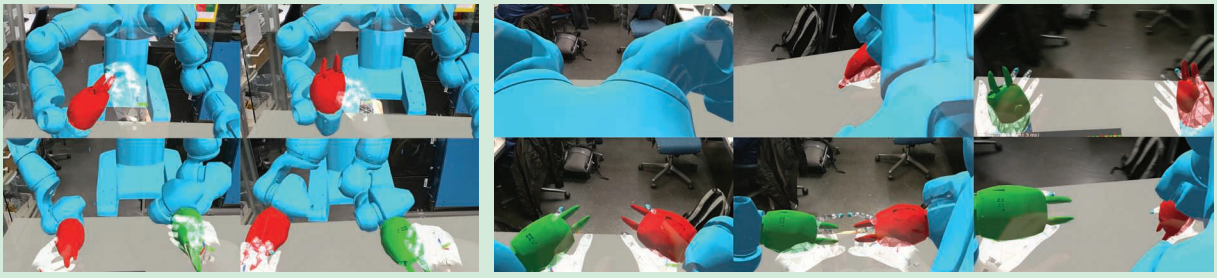


Figure 7. Teleoperation by controlling a digital twin can be done from the outside via kinesthetic programming and from within by embodying the robot and having it follow the motions of the operator. (a) Teleoperation by touch. (b) Teleoperation by embodiment.

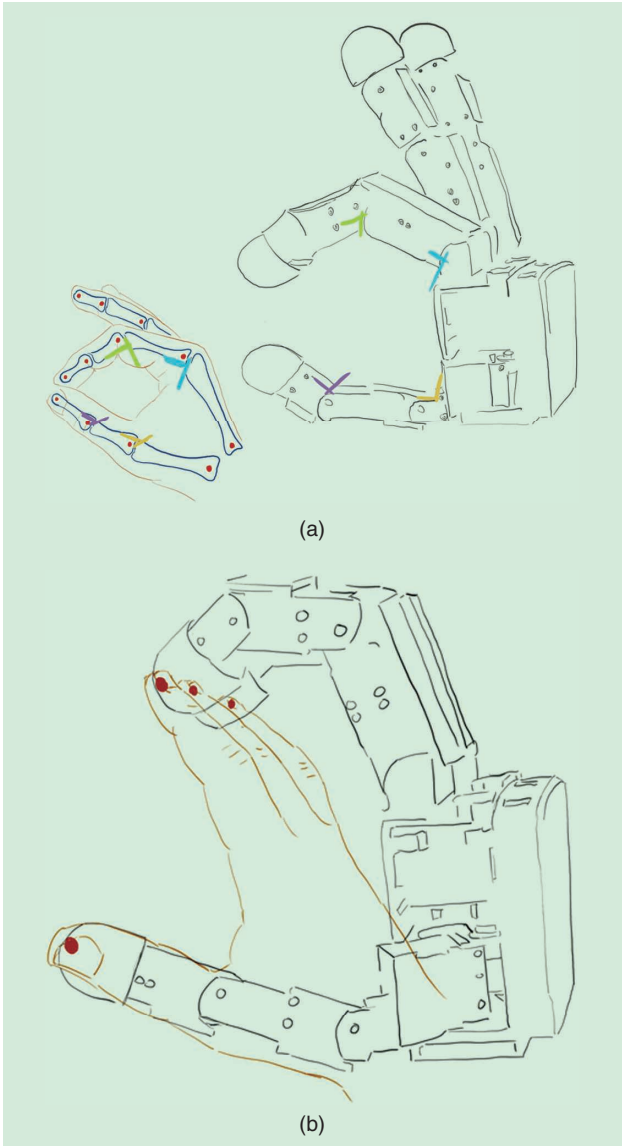


Figure 8. Retargeting hand poses can be accomplished with many different strategies. The simpler strategies involve directly mapping the operator's joint angle to the robot and using the operator fingertips as IK targets. (a) Angle mapping. (b) IK-based mapping.

These methods commonly formulate an optimization problem that minimizes combinations and variations of objectives similar to the ones in the preceding. For example, Rakita et al. [14] used a weighted sum of an IK objective and other smoothness terms, while the authors of [13] minimized a weighted sum of differences among operator finger and robot finger distances, both with heuristically determined weights.

We conducted experiments to evaluate these retargeting methods and found that the technique of choice, and the ideal parameters for it, change drastically among users, to the point where the choice of one user is absolutely unworkable by another. Retargeting does not result in what they perceive as the motion they aimed to perform, and thus it does not lead to the desired degree of body ownership. One of the main reasons is that simplistic retargeting approaches do not faithfully convey the operator's intent. The setup to test this is presented in Figure 9, where we use a system consisting of an Allegro hand mounted on a UR5. The operator could see a holographic digital twin of the system overlayed over his or her hand, which provided some visual feedback, as well as the real robot in the background. The users were asked to manipulate the hand and perform a pick-and-place task with different objectives and parameters. Following the experiment, they were asked to describe their experience, including sources of frustration in operating the system. In the ensuing discussions, it became evident that various users have widely differing preferences for the retargeting model. The hypothesis is that personalizing the model to the user will have him or her perform tasks faster and more accurately and reduce mental strain (evaluated using, e.g., the NASA Task Load Index).

The HoloLens provides a new and exciting methodology for experimenting and optimizing the user experience of retargeting. This is thanks to the ability to overlay a hologram of the robot directly over the operator's body, both sharing the same frame of reference, in contrast to the operator watching the real robot react to her motions from an entirely different perspective, which we observed to be quite confusing for many people. The methodology we propose for user-specific retargeting is as follows: 1) users are shown different robot poses as holograms; 2) they then attempt to pose themselves

Figure 9. Immersive teleoperation by embodiment of a system comprising an Allegro hand mounted on a UR5. The system tracks the motions of the operator shown in the image using a HoloLens. This data is continuously retargeted and submitted to the robot. (a) The operator moves into a pose that will cause the arm to grab the block on the table. (b) The operator changes pose to lift the cube. (c) The inside view of the Hololens, showing the robot hologram superimposed over the user's hand, with the real robot in the background.

to match the robot; 3) then, given a retargeting multiobjective, we can solve an optimization problem that finds the weights that reproduce the users' poses. This involves solving a bilevel optimization problem that can be done, for example, using sensitivity analysis [15]. A preliminary study showed that there is variability to the set of weights optimized for every user. It remains to be seen what the exact distribution of weights is, across different users, and whether a single average set will be effective.

Task Autocorrection

Despite many advances, demonstrations show that state-of-the art systems, while impressive, are far from exhibiting human dexterity, and it is clear that a totally immersive experience will not be attainable in the foreseeable future. One of the issues that will certainly linger is the difference between human and robot morphology. Even with personalized retargeting, this difference will inhibit the sense of body ownership. However, robots can autonomously perform many simple tasks, such as pick and place, that human operators struggle with. Therefore, the question is how to bridge this gap. One approach is known as assistive teleoperation. As described in [16], it is the process of arbitrating, i.e., blending, the operator's motion with a learned optimal policy for a specific task. Thus, if the task is known in advance, the system can retarget the motions by making subtle tweaks to the user's inputs so that his or her maneuvers successfully accomplish the objective and the modifications are not too intrusive. We build on this concept and propose a new concept, which we term *task autocorrection*, which is similar but geared more toward immersion. Task autocorrection refers to the process of making small modifications to the movements of the avatar such that they match the intention of the operator.

The autocorrection framework consists of several components. In addition to the teleoperation setup, there is a retargeter that maps input motions to the avatar, an intent predictor that outputs the user's intent, and the task corrector that adapts the retargeter based on the predicted intent. The retargeter and predictor provide continuous output as the user controls the robot. The task corrector computes the optimal robot motion for the predicted task and blends the user's motion with the optimal one, based on the confidence level of the prediction.

As a proof of concept, we devised a simple deep learning-based autocorrection prototype for the holographic teleoperation setup described previously. In this system, the operator uses a HoloLens 2 to manipulate a holographic robotic arm in a scene consisting of the robot, two desks, and several balls [Figure 10(a)]. The tasks that were considered were picking up and placing the balls and sliding a grasped ball between two walls. The input data for training the intent predictor was the hand palm trajectory, the trajectories of the 26 hand joints, the gaze (origin and direction) trajectory, and the stacked scene data. These were labeled by the action itself (e.g., grab and release) and the object it was applied to (e.g., a blue ball).

The input is transformed into four equal-sized embedded vectors by forward propagation along two stacked long short-term memory layers followed by one ReLU layer in each branch. The embedding vectors are fused by learnable weighted averaging and fed into a final ReLU network before computing the softmax output. The output is a vector, which represents the probability for each action. For the prototype, the actions were "pick," "place," "slide between walls," and "none." In case a pick or place action was predicted, the target indicated which object the action was applied to. The training data set was generated by recording users performing these tasks, e.g., picking up a ball and randomly placing it on a desk and picking up a ball and sliding it between two walls. Around 5 h of training data were gathered. The trained model achieved 89.48% accuracy for action prediction and 85.02% for target prediction on the test set. Next, for each action, an optimal trajectory was defined. For example, to place a ball on a desk, a smooth motion that does not penetrate the desk must be computed. Sliding a ball between the two walls must be done in a perfect linear motion. The optimal trajectory then arbitrates the operator's motion based on the confidence level and an aggressiveness factor that is experimentally determined.

Autocorrection User Study

To evaluate the performance of the prototype, we held a small user study (n = 7), where participants were invited to interact with our system, with or without the assistance of autocorrection. Similar studies have compared the usability of several

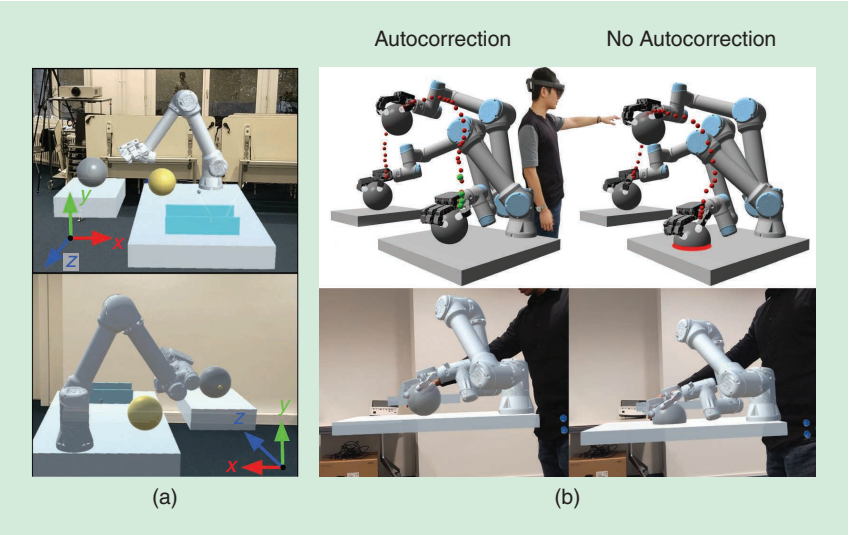


Figure 10. Autocorrection makes it easier for the user to perform tasks accurately. Without autocorrection, the user might place the ball through the table. With autocorrection, the system identifies the task and assists with the optimal motion. (a) The scene. (b) The task performed with autocorrection on and off.

non-MR teleoperation methods [17] and provided useful insights into the effect of the user interface on the helpfulness and ease of use in manipulation tasks. For other tasks such as robotic inspection, AR interfaces have demonstrated significant benefits for users in the speed and performance of robot teleoperation [18]. The participants were required to grab a gray ball and place it on a spot marked by an indicator (shown as a small sphere in the hologram) and then grab a yellow ball and place it next to another indicator. They were told to try to place the ball properly; that is, it should not have been placed in the air or through the surface of the desk. As evident in Figure 10(b), without autocorrection, due to the operator's limited capacity to control the robot, the ball is maneuvered through the desk, but with autocorrection enabled, it is perfectly placed.

In a separate experiment, participants were asked to pick the gray ball and twice drag it from side to side along a slot. In this case, they were told to keep the ball as low as possible without colliding with the desk and walls. Before starting the experiment, a warm-up trial was conducted to let the participants become familiar with the basic operations. For each trial, the participants were asked to perform either a pick-and-place or slide task once with or without enabling autocorrection and without knowing whether the function was enabled or not.

The recorded experiments were evaluated using two quantitative metrics during simulation: violation distance and operation time. In addition, the participants were asked to rate the degree of naturalness, from one being "very unnatural" to five being "very natural," during the control for each trial. A task load index was also rated by participants for this question, referring to the effort rating of the NASA Task Load Index, which we rescaled from one, meaning "very easy," to five, meaning "very hard." The hypotheses were that the autocorrected motions would cut the violation distance, shorten the operation time, and reduce the task load index while lowering the naturalness score as well. Analysis shows that the improvement was statistically significant, with the downside being the reduced naturalness score. This drawback could be addressed by using a more refined autocorrection system.

Outlook

The solutions presented in this section address some of the challenges with the remote teleoperation of robot arms. Motion retargeting and task autocorrection make human–robot interaction feel more natural and help users behave more naturally through assistance with executing accurate moves. Looking forward, there are several limitations to the current framework. The MR device acts as an egocentric sensor platform, capturing the human's actions, but these works do not yet take advantage of either the robot's or the human's spatial context. Extending the proposed system to tasks that require dynamics will demand physical simulation. A realistic simulation will further enable gathering data that include different grasps and more dexterous manipulations tasks. Considering its scalability to many tasks, another limitation of the proposed approach is the necessity to obtain data for each task and define each optimal trajectory. To this end, a database of tasks and optimal trajectories must be established, and autocorrection could even be used to bootstrap it. In addition to improving the generalization of these methods to more and more arbitrary tasks, future work to leverage spatial computing will provide immersive and embodied teleoperation experiences with additional capabilities to accomplish tasks in real-world environments. Separate from the addition of new features, an expanded user study with more participants and deeper experiments would provide valuable information to guide future work in preparing autocorrection to be applied to real-world tasks.

Conclusion

This article presented several prototype systems that utilize robots and MR devices to provide novel solutions to compelling real-world applications through human–robot interaction. The two key technologies that enable these solutions are the spatial computing and egocentric sensing capabilities of MR devices. All three systems make use of one or both of these to provide new spatial capabilities as well as intuitive and natural interaction. However, they are only preliminary explorations toward these real-world applications, and our initial results have uncovered more questions than they have answered.

The primary barrier to the practical deployment of the proposed mission planning framework from the "Sharing Spatial Information" section is the lack of support for large-scale, continuous maps in ASA (and any other spatial anchor cloud provider). Ideally, we would like to have our robots and devices continuously localized as they move through the environment, not just when they are near a particular anchor. This capability will require large-scale, continuous maps in the cloud and a service that provides localization to a venue (rather than an individual anchor) that is robust to the network dropouts that may occur in large indoor spaces. The robots and MR devices should also be able to leverage a priori spatial data from building information models and CAD data to visualize and use as-planned information from these designs and as-built information from physically being on site. These features are not present in current commercial localization services, but state-of-the-art research has demonstrated that these capabilities are ready to be applied to these real-world scenarios.

On the other hand, there are still many open avenues for research in human–robot interaction through MR. In particular, although we have demonstrated intuitive control by giving a robot commands through natural hand gestures, the robot's understanding of this interaction is no different than if these navigation goals were provided by a keyboard and mouse, and its representation of its environment is still one of obstacles and free space. While this lack of semantic understanding is not a barrier for simple tasks, such as navigation to a goal, further work will be required to endow the robot with the understanding of human presence and the semantic context necessary to perform more complex tasks that can be communicated in a natural way by humans. We have released

several pieces of open source software (see the "Sharing Spatial Information" section) to help facilitate research in this domain that interfaces HoloLens 2 devices and robots.

Within immersive teleoperation, some of the major open questions will need to be resolved through better human understanding. The user studies around motion retargeting and task autocorrection have, so far, been limited in scope. Consequently, there are questions about whether it is possible and beneficial to provide personalized models for every user to provide an improved experience from a single method. Deeper and broader user studies should be performed to better understand the requirements for general usability and provide insights on future research questions to pursue.

Bringing spatial and human understanding together in a human–robot system offers promising new modes of interaction. We have shown several MR-based systems that leverage these technologies to provide solutions to real-world robot use cases. While these capabilities are deployable today and applicable to many commercial and research scenarios beyond the scope of this article, these works represent the tip of the iceberg, with many possibilities for future research in spatial computing and human–robot interaction.

Acknowledgments

The authors would like to thank Oswaldo Ferro, Jan Stühmer, Lukas Gruber, and Simon Huber for their contributions to these projects.

References

[1] S. Greenwold, *Spatial Computing*. Cambridge, MA, USA: MIT Press, 2003.

[2] M. Gianni, G. Gonnelli, A. Sinha, M. Menna, and F. Pirri, "An augmented reality approach for trajectory planning and control of tracked vehicles in rescue environments," in *Proc. 2013 IEEE Int. Symp. Saf., Security, Rescue Robot. (SSRR)*, pp. 1–6, doi: 10.1109/SSRR.2013.6719360.

[3] D. Vaquero Melchor, "Technologies and concepts for enhancing interaction and enabling collaboration in augmented and mixed reality," Ph.D. dissertation, Universidad Politécnica de Madrid, Madrid, Spain, 2020.

[4] Y. Liu *et al.*, "Usability evaluation for drone mission planning in virtual reality," in *Proc. Int. Conf. Virtual, Augmented Mixed Reality*, 2018, pp. 313–330, doi: 10.1007/978-3-319-91584-5_25.

[5] H. Fang, S. K. Ong, and A. Y.-C. Nee, "Robot programming using augmented reality," in *Proc. 2009 IEEE Int. Conf. CyberWorlds*, pp. 13–20, doi: 10.1109/CW.2009.14.

[6] M. Ostanin and A. Klimchik, "Interactive robot programing using mixed reality," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 50–55, 2018, doi: 10.1016/j.ifacol.2018.11.517.

[7] J. Delmerico *et al.*, "The current state and future outlook of rescue robotics," *J. Field Robot.*, vol. 36, no. 1, pp. 1171–1191, 2019, doi: 10.1002/rob.21887.

[8] K. Chandan, V. Kudalkar, X. Li, and S. Zhang, "ARROCH: Augmented reality for robots collaborating with a human," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2021, pp. 3787–3793, doi: 10.1109/ICRA48506.2021.9561144.

[9] F. Muhammad, A. Hassan, A. Cleaver, and J. Sinapov, "Creating a shared reality with robots," in *Proc. IEEE ACM/IEEE Int. Conf. Human-Robot Interact. (HRI)*, 2019, pp. 614–615, doi: 10.1109/HRI.2019.8673191.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[11] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2009.

[12] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[13] A. Handa *et al.*, "DexPilot: Vision-based teleoperation of dexterous robotic hand-arm system," in *Proc. 2020 IEEE Int. Conf. Robot. Automat. (ICRA)*, pp. 9164–9170, doi: 10.1109/ICRA40945.2020.9197124.

[14] D. Rakita, B. Mutlu, and M. Gleicher, "A motion retargeting method for effective mimicry-based teleoperation of robot arms," in *Proc. 2017 12th ACM/IEEE Int. Conf. Human-Robot Interact.*, pp. 361–370, doi: 10.1145/2909824.3020254.

[15] S. Zimmermann, G. Hakimifard, M. Zamora, R. Poranne, and S. Coros, "A multi-level optimization framework for simultaneous grasping and motion planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2966–2972, 2020, doi: 10.1109/LRA.2020.2974684.

[16] A. Dragan and S. Srinivasa, "Formalizing assistive teleoperation," in *Proc. Robot., Sci. Syst.*, pp. 73–80, Jul. 2012, doi: 10.15607/RSS.2012.VIII.010.

[17] D. Kent, C. Saldanha, and S. Chernova, "A comparison of remote robot teleoperation interfaces for general object manipulation," in *Proc. 2017 ACM/IEEE Int. Conf. Human-Robot Interact.*, pp. 371–379, doi: 10.1145/2909824.3020249.

[18] H. Hedayati, M. Walker, and D. Szafir, "Improving collocated robot teleoperation with augmented reality," in *Proc. ACM/IEEE Int. Conf. Human-Robot Interact.*, 2018, pp. 78–86, doi: 10.1145/3171221.3171251.

Jeffrey Delmerico, Microsoft MR and AI Lab, Zürich, 8001, Switzerland. Email: jeffrey.delmerico@microsoft.com.

Roi Poranne, Computational Robotics Lab, ETH Zürich, Zürich, 8092, Switzerland, and Department of Computer Science, University of Haifa, Haifa, 3498838, Israel. Email: roi.poranne@inf.ethz.ch.

Federica Bogo, Microsoft MR and AI Lab, Zürich, 8001, Switzerland. Email: federica.bogo@microsoft.com.

Helen Oleynikova, NVIDIA, Zürich, 8004, Switzerland. Email: holeynikova@nvidia.com.

Eric Vollenweider, Microsoft MR and AI Lab, Zürich, 8001, Switzerland. Email: eric.vollenweider@microsoft.com.

Stelian Coros, Computational Robotics Lab, ETH Zürich, Zürich, 8092, Switzerland. Email: scoros@ inf.ethz.ch.

Juan Nieto, Microsoft MR and AI Lab, Zürich, 8001, Switzerland. Email: juan.nieto@microsoft.com.

Marc Pollefeys, Microsoft MR and AI Lab, Zürich, 8001, Switzerland. Email: marc.pollefeys@microsoft.com.