Chapter 2
Augmented Reality for Interactive Robot Control

Levi Manring, John Pederson, Dillon Potts, Beth Boardman, David Mascarenas, Troy Harden, and Alessandro Cattaneo

Abstract Robots are widely used to support mission-critical, high-risk and complex operations. Human supervision and remote robot control are often required to operate robots in unpredictable and changing scenarios. Often, robots are controlled remotely by technicians via joystick interfaces which require training and experience to operate. To improve robot usage and practicality, we propose using augmented reality (AR) to create a more intuitive, less training-intensive means of controlling robots than traditional joystick control. AR is a creative platform for developing robot control systems, because AR combines the real world (the environment around the user, the physical robot, etc.) with the digital world (holograms, digital displays, etc.); it can even interpret physical gestures, such as pinching two fingers.

In this research, a Microsoft Hololens headset is used to create an AR environment to control a Yaskawa Motoman SIA5D robot. The control process begins with the user placing an interactable holographic robot in 3D space. The user can then select between two control methods: manual control and automatic control. In manual control, the user can move the end effector of the holographic robot and the physical robot will respond immediately. In automatic control, the user can move the end effector of the holographic robot to a desired location, view a holographic preview of the motion, and select execute if the motion plan is satisfactory. In this preview mode, the user is able to preview both the motion of the robot and the torques experienced by the joints of the manipulator. This gives the user additional feedback on the planned motion. In this project we succeeded in creating an AR control system that makes controlling a robotic manipulator intuitive and effective.

Keywords Augmented reality · Microsoft Hololens · Robotic arm · Force feedback · Motion planning

2.1 Introduction

Augmented reality (AR) is an emerging field that offers new tools for human-based control and interaction with complex systems. AR combines the sensory experience of a user with information from a digital system; this contrasts with virtual reality, which seeks to eliminate the user's perception of the real world. Physical-digital interaction allows for intuitive control of digital systems with sensory feedback that can be more easily interpreted by the user. Intuitive AR systems are used in applications such as virtual surgery training and robotic tele-operation. AR is particularly advantageous for robotic control where direct human control is needed; because conditions and objectives are unpredictable or rapidly changing, decisions must be made by an operator in real time.

Current robotic control is difficult because of non-intuitive controls (e.g. moving a joystick to the right to rotate a gripper) and a lack of force feedback, since haptic feedback interfaces are still niche products. Without force feedback, it is difficult for an operator to have a sense of how much force a robot is exerting on an object; this often leads to over- or under-applying

L. Manring
Department of Mechanical Engineering and Materials Science, Pratt School of Engineering, Duke University, Durham, NC, USA

J. Pederson
Department of Mechanical Engineering, The Fu Foundation School of Engineering and Applied Science, Columbia University, New York, NY, USA

D. Potts
Department of Mechanical Engineering, Ira A. Fulton College, Brigham Young University, Provo, UT, USA

B. Boardman · T. Harden · A. Cattaneo (✉)
Engineering Technology and Design Division, Los Alamos National Laboratory, Los Alamos, NM, USA
e-mail: cattaneo@lanl.gov

D. Mascarenas
Engineering Institute, Los Alamos National Laboratory, Los Alamos, NM, USA

N. Dervilis (ed.), *Special Topics in Structural Dynamics & Experimental Techniques, Volume 5*, Conference Proceedings of the Society for Experimental Mechanics Series, https://doi.org/10.1007/978-3-030-12243-0_2

forces, which can be detrimental in delicate operations. These two limitations require human operators to have extensive training and experience. By implementing AR robotic control, the amount of training required to operate robots will be reduced, and operators will be able to control applied forces more accurately. Improving human-robot control will allow for a wider utilization of robots in many fields where real-time operator control is necessary.

2.2 Background

Augmented reality (AR) is emerging as a technology that is able to connect people to a growing number of tools in a more user-friendly manner. As a result, AR applications are growing in complexity and in scope, which can be seen in the following published applications.

There are multiple AR mediums, and in [1] touch-screen devices are used to implement AR as a feature in a game that encourages exercise in young people. The authors developed their research game using Unity3D, the Vuforia AR library, and OpenGL to create an interactive AR world game-space on touch-screen devices. Ref. [2] uses spatial AR. This involves projecting AR from a projector (as opposed to the classic uses of touchscreen devices, computers, and glasses). This research sought to develop a control method to control the position and orientation of such a projector to be used for an even wider workspace. The advantage of the system proposed in this work lies in its understanding of kinematic control without knowing the exact kinematic specifications of a projector pan/tilt mount (such as an ad hoc assembly). In [3], the authors propose an AR system that generates pattern identifications by analyzing infrared optical markers projected in actual space measured by an infrared camera. For this system, the operator gives commands by projecting markers on a position on a screen. AR applications continue to be developed to assist in the daily activities of people affected by adverse health conditions [4]. This paper proposes a unique application for AR to assist people affected by paralysis in performing daily tasks. The authors use a BMI (brain-machine interface) to measure the brain-waves of the user be means of electroencephalography. They also use gaze-selection via pupil tracking. The BMI enables to denote either a "command" state or a "rest" state based on the brain activity of the user, and the gaze-selection is used to denote direction and goals for the robotic arm. The researchers found that across the board, AR assisted the user in selection and movement of obstacles using the robotic arm. The researchers propose in future work the importance of changing the interface medium from a computer screen to wearable AR glasses.

As research in the field of robotics continues to grow, AR is being used more often to reduce the complexity of controlling robots. In [5], the authors propose a mixed reality human-robot interface to assist operators in tele-operation for remote maintenance tasks. In this system, visual inspection and corrective task execution are two phases activated by the operator. For the second phase, a virtual robot helps the operator visualize the movement of the robotic arm before the task is executed. This allows the user to create/refine path movements in advance of completing them. However, this system also requires that the obstacles in the environment are known about a priori. Part of their interest in future work is to input depth sensors to detect objects in the surroundings. In [6], the authors seek to perform something similar to [5] by overlaying a computer graphic/hologram on top of the actual robot that is updated based on user input. The purpose of this graphic is to account for the time delay between when the user sends a command and when the physical robot responds. The graphic will move immediately, the actual robot will follow the graphic, and they will eventually meet. In addition, digital handles are projected over the real robotic hand which can be selected and moved to rotate the robotic arm. Another AR-based system to facilitate programming robots and planning trajectories is presented in [7]. This research incorporates robot dynamics into task-optimized executable robot paths that are also collision free. A method for assisting in robot end-effector planning using AR is presented in [8]. The authors implement a trajectory optimization scheme to assist in end-effector location and angle. This research also considers the system dynamics of the robot while checking for collision detection and simulating the torque and velocity of each robot joint. If the torque/velocity are outside the designated limits, that joint is highlighted as one that has a high probability of deviating from a planned trajectory. The package Roboop is used for robot kinematics and dynamics modeling (this package is no longer available). The authors are able to demonstrate an increased performance in a trajectory that is obtained by incorporating robot dynamics versus one that only considers kinematics.

There has been a significant amount of work in the area of providing force-feedback to the user using AR. In [9], the authors employ the principle of reverse electrovibration using AR, where a weak electrical signal is injected anywhere on the user body to create an oscillating electrical field around the user's fingers. Using this principle, the user can have a perception of texture that is physically not present. The work of the authors resulted in a device, REVEL, that allows tactile textures to be modified in real time. One of the primary benefits of this system is that it allows tactile feedback without the use of gloves, since it only requires the user to wear a small tactile passive signal generator that can be attached anywhere on the user's body. The research presented in [10] seeks to overcome the issues with time delay as it influences tele-operated systems. The authors propose a virtual model which can estimate the real-time force feedback and give visual information using AR to

the operator, which reduces the effects of time delay. In [11], the authors present an interesting method for training surgeons using force feedback in combination with AR. This gives the trainee information on collision detection and location. The motivation of this research is to reduce the risks associated with training new surgeons in an actual surgery. A method for providing haptic feedback in an AR setting is presented in [12]. The goal of this research is to enable AR users to have feedback on roughness and friction of surfaces. This paper presents a novel method of using an ungrounded haptic stylus with actuators to overlay texture vibration and friction forces when the user touches an object, which allows the virtual and real worlds to mix.

These AR methods have also been extended to control multiple robots, as shown in [13], where a method for implementing AR for single-user control of four Mindstorms NXT robots is presented. In particular, the authors use Point-and-Go and path planning as two methods to reduce mission completion times as compared to typical joystick control methods. They use a Probabilistic Roadmap Planner as their path planner for the robots to provide a higher level of automation than the Point-and-Go method. This resulted in significantly lower mission completion times and a strong preference by test users for the path-planning AR method. In [14], the authors seek to use ROS and Unity to simulate multiple UAVs to easily verify and develop flight control and navigation algorithms. They also present a layout of the system level structure that is very useful for reconstructing similar applications (including Windows, Ubuntu, ROS, and Unity connections and how it works together). The authors also discuss Unity Socket, which uses TCP/IP to exchange data between ROS and Unity3D.

There are a number of publications that detail the connection between ROS and Unity3D as a means for implementing AR in robotic systems. In [15], the authors seek to reduce the costs of developing real robots and eliminate some of the cost of performing real-world human-robot interaction experiments. They present a modified virtual reality system named SIGVerse 3.0 which includes a connection mechanism to bind Unity to the ROS environment, enabling users to develop robot software in ROS and place a virtual robot model into Unity applications. In addition, this paper includes a helpful table of functions and limitations of related systems which show the compatibility and usefulness of different platforms, showing that their new version of SIGVerse is preferable. In [16], the authors cover the main advantages of different software for visualization and robotic control, particularly using ROS and Unity3D. ROS is based on message passing while Unity3D is tied to a rendering loop where commands are made/updated each frame. Because of this difference, interaction between the two programs is difficult. ROSbridge is used to connect ROS with the outside world. ROSbridge uses WebSocket protocol to communicate JSON (JavaScript Object Notation) strings. The process of communicating between ROS and Unity3D involves transmitting JSON-encoded messages through a WebSocket. In [17], a system is presented for simulating executing the task of monitoring an industrial process, with a particular focus on shortening optimization time for a robot and make the final commissioning more efficient (this is related to the expensive process of tuning and calibrating a one-of-a-kind robot). One of the key parts of their implementation is the use of ROSbridge, which establishes a connection between Unity3D and ROS which allows for invoking ROS services.

Our application of AR is a unique system that allows the user to control a robot by means of holograms projected by a Microsoft Hololens AR headset and provides the user torque feedback that can assist in robot operation.

2.3 Methodology

2.3.1 Overview

To implement robotic control with augmented reality, a Microsoft Hololens headset, used to create an AR environment, communicates with a controllable robotic arm, the Yaskawa Motoman SIA5D. The layout of this system can be seen in Fig. 2.1. The Hololens is a Windows 10 based device that creates and modifies the AR environment. A Unity development environment, which utilizes C# in a .NET framework and scripting backend, creates the AR scene. The Hololens projects holograms into the real-world view of the user. Users interact with the holograms through hand motions. Hand motions, known as gestures, are used to make selections and move holograms. One example of a gesture is the ability to “click”, which occurs when the user moves thumb and index finger from an open position to a closed position. To interface with the SIA5D Motoman robot, we used the conventional Robot Operating System (ROS) (version indigo). The Hololens sends and receives information from the robotic arm using the ROS. The communication between the Hololens and ROS is done via a ROSbridge websocket.

Fig. 2.1 The transmitting/reception layout connecting the Hololens to the SIA5D robot

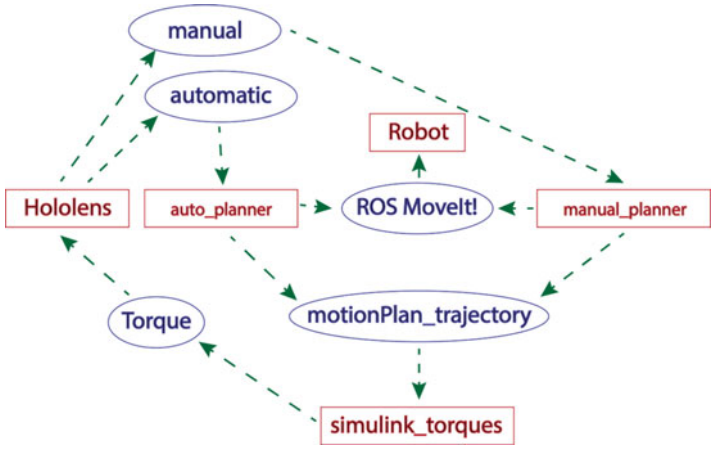

Fig. 2.2 A diagram of the ROS nodes and topics used in this application

2.3.2 System Layout and ROS

The framework of our system can be best understood after a brief discussion of how ROS works. ROS provides a useful framework for communicating with the robot using categories called nodes, topics, publishers, and subscribers. Nodes represent any entity involved in the robot's operation; examples include sensors, processors, and motors. Topics are entities that act as channels to pass information between nodes. Publishers are a type of node that publish information to topics, and subscribers are nodes that subscribe to topics. For node A to pass information to another node B, it must publish that information to a topic, and node B must subscribe to that topic.

One particularly useful feature of ROS is that it is simple to go from simulation of the robot in RViz (a visualization library in ROS) to actual control of the physical robot. The system implemented in this paper is laid out in Fig. 2.2, where the blue ovals denote topics and the red rectangles indicate nodes in ROS.

In Fig. 2.2, the Hololens node publishes desired robotic arm position information to the two robot control topics: manual and automatic depending on whether the user chooses to use manual or automatic mode. Auto_planner and manual_planner each subscribe to the automatic and manual topics, respectively. These nodes calculate and publish a motion plan to the motionPlan_trajectory topic. The simulink_torques node subscribes to the motionPlan_trajectory node and calculates a prediction for the torque exerted on the robot based on the motion plan in the motionPlan_trajectory topic. The torque calculations are then combined with the motion plan information and published to the Torque topic. The Hololens node uses the information stored in the Torque topic to preview motion plans and torque values graphically with holograms. Then the user has the option of allowing the robot to execute the plan. Pressing the "Execute" button in the user interface sends a message over the execute topic, as well as the robot's current joint states over the automatic topic; the Hololens receives the information via the websocket connection and moves the robot according to the joint states specified.

The simulink_torques node is an implementation of the Robotics System Toolbox (RST) from Mathworks that calculates the inverse dynamics (particularly joint torques) based on the motion plan. The Hololens node subscribes to the simulink_torques topic so that the user can gain additional information about the expected loadings on the robot manipulator.

2.3.3 Unity

To coordinate the holograms, user interface, and C# scripts that make up our application, we used the popular game engine Unity. Unity allows the user to place, modify, and manipulate two and three-dimensional objects, including text boxes, buttons, and CAD models in a virtual environment. In addition, scripts written in the C# programming language can be attached to such objects; their methods can be used to move and change objects, get system properties, and even communicate with websockets which are a means of two-way, message-based communication over a local network.

Unity packages add additional functionality to the editor itself. For our application, we used the ROS and HoloToolkit packages. ROS is a free library designed to improve communication between Unity and ROS; it includes scripts for communication over a websocket via ROSbridge, as well as an importer for Universal Robot Description Format (URDF) files. The HoloToolkit enables gesture recognition functionality to be used in Unity programs. To produce our application environment, we first imported the necessary packages and the URDF of our SIA5 robot into the Unity environment. We then wrote and compiled several scripts that coordinated all aspects of the application; these tasks included communicating with ROSbridge websockets, dragging and moving the robot's virtual end effector, and animating the robot's joints. We also added physical joints with rotation limits to the 3D robot model, in order to ensure that the virtual robot would only animate valid robot positions.

2.3.4 Control Modes

The AR robotic control system we implemented has multiple options for control. For this application, we have created two control modes: manual and automatic. In each mode, the motion planner library within MoveIt! is used to plan the robot trajectory. The MoveIt! library works by loading a robot package, which is created by importing a URDF file for the SIA5D robot and designating groups of joints and end effectors. The user can get a trajectory by designating a goal pose or by specifying a set of joint states to reach. The motion planning library will plan the trajectory while avoiding self-collisions but will not execute the trajectory unless given a command to execute. The implementation of MoveIt! with the robot is done using C++ code to create nodes that call this library.

2.3.5 Manual Control

When manual control is selected, the user moves the holographic robot in 3D space by clicking and dragging the virtual end effector with their hand. At a predetermined update rate, the robotic arm follows the motion of the holographic robot, making for near-real-time robot motion control. In ROS, this means that as soon as soon as the manual_planner node calculates the motion plan, the motion plan is executed. The user can move the holographic end effector by pinching their fingers over the holographic end effector and moving it where desired. The physical end effector then follows the motion of the holographic end effector.

2.3.6 Automatic Control

In automatic control, the user moves the robot using the hologram's virtual end effector in the same manner as manual control. However, the robotic arm does not automatically move to the position specified by the holographic arm; it instead waits for further user input. The Hololens can preview the motion plan by having the holographic arm follow the proposed path. If the user is satisfied with the motion plan, they can execute it; if not, they can move the arm to a new position and try again, or switch to manual control. In ROS, the motion plan is saved to a .bag file so that it can be executed when the user desires.

2.3.7 Connections

Much of the difficulties encountered in building our application lay in the various communications and connections between each component of the project. The Hololens communicates with the ROS environment to exchange information about user-selected end-effector positions and robot joint states; it does this through an intermediary websocket called ROSbridge. Both positions and states must be published to ROS topics in order for the path-planning algorithms to use them; thus, the C# scripts first format this information according to the JSON string-based file format and then add the topic to which it will be published. The Hololens must then send the resulting formatted string to the ROSbridge websocket server, which runs on the Linux system running ROS; using the Windows.Networking.Sockets library, the Hololens connects to the websocket as a client and sends the JSON-formatted string messages to it. Once received, ROSbridge parses the messages and publishes their information to the ROS topics specified in the JSON formatted strings. The motion planning algorithm nodes (auto_planner and manual_planner) then subscribe to these ROS topics, process the information to create a motion plan, and publish the plan over the motionPlan_trajectory and ROS MoveIt! topics; the MoveIt! node subscribes to the latter topic and moves the robotic arm according to the motion plan.

Receiving information on the Hololens (such as the motion plan and torque information) works much the same way as transmitting in reverse, except for a few differences on the ROSbridge server. For the Hololens to subscribe to the motionPlan_trajectory and Torque topics (which contain information about the motion plan and joint torques, respectively), it must tell ROSbridge to subscribe to those particular topics. Then, whenever a message is published on those topics, ROSbridge will format the message into a JSON string and send it to the Hololens client. The Hololens then uses the information to relay motion plan previews and torque information to the user. The various ROS components (including the path-planning algorithms, torque simulators, and robotic controllers) communicate internally with each other via ROS topics, as mentioned above.

2.3.8 Torque Simulation in ROS

One of the primary features of AR we have implemented is the ability to employ additional user feedback. One common disadvantage of conventional robot control implementations is that force/torque feedback is difficult to provide and if this feedback is available, it is difficult to give a meaningful output to the user. This means that the user is "running blind" when it comes to understanding the loading that a manipulator is undergoing.

In ROS, we can extract the motion plan for a robot trajectory in the form of waypoints and timestamps, which is effectively a time-series for each joint of the robot. Using Mathworks RST, we can import this time-series and use an inverse dynamics solver to compute the torques for each joint. For this application, we used Simulink as our primary platform because it is easier to deploy with ROS than Matlab alone. Simulink has blocks for publishing and subscribing to topics in ROS and has the ability to be deployed as a standalone application as a node in ROS as seen in Fig. 2.3.

In Fig. 2.3, the red block describes the subscription to the motion plan calculated by the MoveIt! library. The blue block calculates the inverse dynamics of the system of ordinary differential forcing equations with the state variables (position, velocity, and acceleration) along with mass and inertia information for the joints. Finally, the simulink_torques node sends the torque information back to the Hololens in the green publishing block.

Fig. 2.3 The Simulink diagram for the simulink_torques node

Fig. 2.4 A screenshot of the Hololens application in action. *Left:* The physical robotic arm has just moved to the position of the holographic arm. Notice the overlay of the hologram onto the physical robotic arm. *Right:* The hologram "Preview Motion Plan" operation, providing torque feedback via color media

2.4 Results

The final result of our project was a Hololens application and Linux terminal backend that can successfully control a SIA5D robotic arm. Designed for intuitive user control, the application features a holographic button interface that follows the user, manual and automatic control modes, motion plan previewing, torque simulation, and visual feedback.

To use the application, the ROS scripts and nodes are first initialized and run on the Linux machine. Next, the ROSbridge websocket is launched in the Linux terminal. Our application is then run on the Hololens. In the user's field of view, a single holographic button labeled "Open Menu" will appear. When clicked, a menu containing buttons for manual control, automatic control, and holographic arm initialization will appear. When the user clicks the initialization button, a holographic model of the robotic arm appears. The user then drags the hologram until it is superimposed on the physical robot.

Once the hologram has been superimposed on the robotic arm, the user selects either manual or automatic control with the appropriate buttons. For both manual and automatic control, the user drags the virtual end effector to move the holographic arm. In manual control, the joint angles of the holographic arm are transmitted to the motion planner once a second, and the robotic arm moves to mimic the hologram accordingly. This creates a near-real-time remote manipulation of the robotic arm. In automatic control, the robot moves to the position of the holographic arm only when the "Execute" button is selected by the user. In addition, automatic control has a "Preview Motion Plan" button, which when selected, displays a preview of the calculated motion plan by moving the holographic arm accordingly. Unfortunately, there was a slight issue with exactly mimicking the motion plan due to a disparity between the joint angles of the Unity model and those of the real robot. This issue only raised its head in the "Preview Motion Plan" operation, but the simulated torques for the joints were displayed through changes of color, varying from blue to red (low to high torque) and then black if the joint torque exceeds some maximum torque value (Fig. 2.4).

2.5 Conclusion

Using the above connections, interfaces, simulations, and algorithms, we were successful in controlling the SIA5D robotic arm using the Hololens. Both manual and automatic modes work as described, and the torque state algorithms, based on the robotic arm's specifications, are able to predict joint torques. While all of the aforementioned connections (e.g. websockets, ROSbridge, publishers and subscribers, MoveIt!) have been employed in previous academic and industrial work, our intuitive robotic control interface is the first application, to our knowledge, to combine all of them in a seamless pipeline from Hololens to robotic motion. This integration will help pave the way for future applications of augmented reality to practical, industrial, and other applications.

2.6 Future Work

Expanding the options and capabilities of our control modes would improve the user experience, as well as make our project more usable in a wide variety of environments and applications. For example, the ability to manipulate all of the robot linkages, instead of just the end effector, would allow for much more precise control of joint positions and robot poses, as well as adding greater obstacle avoidance capabilities.

Implementing a mesh-based obstacle avoidance functionality would aid in maneuvering the robotic arm in tight spaces, as well as areas with several obstacles to the robotic arm's motion. One of the Hololens' core attributes is the ability to detect its surroundings; by using several cameras mounted around the headset, the Hololens can generate a 2D mesh of its environment—essentially creating a map of all visible obstacles around it. One could envision feeding this mesh into the motion planning algorithm, thus giving the application the ability to plan the path of the robotic arm around any visible obstacles.

Reducing the time delay involved in our near-real-time manual control would also aid the user experience. Due to the large number of connections, algorithms, and interfacing involved, the delay between the user dragging the holographic arm and the robotic arm reaching the corresponding pose is approximately one second. This could be potentially be reduced by using lower-level interfaces with the Yaskawa Motoman robot controller itself; field programmable gate arrays (FPGA's), for example, could potentially speed up communications by eliminating the overhead of ROS.

Acknowledgements We would like to acknowledge the 2018 Dynamics Summer School at the Los Alamos National Laboratory for sponsoring this project as well as James Riback and Anita Jaramillo for their contributions to this project.

References

1. Kim, S.L., Suk, H.J., Kang, J.H., Jung, J.M., Laine, T.H., Westlin, J.: Using unity 3D to facilitate mobile augmented reality game development. In: 2014 IEEE World Forum on Internet of Things (WF-IoT), pp. 21–26. IEEE, Hoboken (2014)
2. Lee, A., Lee, J.-H., Kim, J.: Data-driven kinematic control for robotic spatial augmented reality system with loose kinematic specifications. ETRI J. **38**(2), 337–346 (2016)
3. Kuriya, R., Tsujimura, T., Izumi, K.: Augmented reality robot navigation using infrared marker. In: Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on, pp. 450–455. IEEE, Hoboken (2015)
4. Zeng, H., Wang, Y., Wu, C., Song, A., Liu, J., Ji, P., Xu, B., Zhu, L., Li, H., Wen, P.: Closed-loop hybrid gaze brain-machine interface based robotic arm control with augmented reality feedback. Front. Neurorobot. **11**, 60 (2017)
5. Yew, A.W.W., Ong, S.K., Nee, A.Y.C.: Immersive augmented reality environment for the teleoperation of maintenance robots. Procedia CIRP. **61**, 305–310 (2017)
6. Hashimoto, S., Ishida, A., Inami, M., Igarashi, T.: Touchme: an augmented reality based remote robot manipulation. In: 21st International Conference on Artificial Reality and Telexistence, Proceedings of ICAT2011. Osaka University, Osaka (2011)
7. Fang, H.C., Ong, S.K., Nee, A.Y.C.: Interactive robot trajectory planning and simulation using augmented reality. Robot. Comput. Integr. Manuf. **28**(2), 227–237 (2012)
8. Fang, H.C., Ong, S.K., Nee, A.Y.C.: Orientation planning of robot end-effector using augmented reality. Int. J. Adv. Manuf. Technol. **67**(9–12), 2033–2049 (2013)
9. Bau, O., Poupyrev, I.: REVEL. ACM Trans. Graph. **31**(4), 1–11 (2012)
10. Zhao, Z., Huang, P., Lu, Z., Liu, Z.: Augmented reality for enhancing tele-robotic system with force feedback. Robot. Auton. Syst. **96**, 93–101 (2017)
11. Chen, R.-J., Lin, H.-W., Chang, Y.-H., Wu, C.-T., Lee, S.-T.: Development of an augmented reality force feedback virtual surgery training platform. Int J. Autom. Smart Technol. **1**(1), 41–51 (2011)
12. Culbertson, H., Kuchenbecker, K.J.: Ungrounded haptic augmented reality system for displaying roughness and friction. IEEE-ASME Trans. Mech. **22**(4), 1839–1849 (2017)
13. Lee, S., Lucas, N.P., Darin Ellis, R., Pandya, A.: Development and human factors analysis of an augmented reality interface for multi-robot tele-operation and control. In: Unmanned Systems Technology XIV, vol. 8387, p. 83870N. International Society for Optics and Photonics, Baltimore, Maryland (2012)
14. Hu, Y., Meng, W.: Rosunitysim: Development and experimentation of a real-time simulator for multi-unmanned aerial vehicle local planning. Simulation. **92**(10), 931–944 (2016)
15. Mizuchi, Y., Inamura, T.: Cloud-based multimodal human-robot interaction simulator utilizing ros and unity frameworks. In: System Integration (SII), 2017 IEEE/SICE International Symposium on, pp. 948–955. IEEE, Hoboken (2017)
16. Codd-Downey, R., Mojiri Forooshani, P., Speers, A., Wang, H., Jenkin, M.: From ROS to unity: leveraging robot and virtual environment middleware for immersive teleoperation. In: Information and Automation (ICIA), 2014 IEEE International Conference on, pp. 932–936. IEEE, Hoboken (2014)
17. Sita, E., Horvath, C.M., Thomessen, T., Korondi, P., Pipe, A.G.: ROS-Unity3D based system for monitoring of an industrial robotic process. In: 2017 IEEE/SICE International Symposium on System Integration (SII). IEEE, Hoboken (2018)