Exploring the VR-based PBD programming approach to teach industrial robotics in manufacturing education

Dr. Yi-hsiang Isaac Chang, Illinois State University

Dr. Yi-hsiang Chang is an assistant professor in the Department of Technology at Illinois State University. He received a MSME degree from Carnegie Mellon University, a MSIE degree and a PhD degree in Technology, both from Purdue University. Dr. Chang has worked with several international corporations including Boeing, DaimlerChrysler, and Dassault Systemes, as well as multiple small to medium size companies in both US and Asia. His current field of focus is in product innovation, process improvement, and technology diffusion.

Dr. Kevin L Devine, Illinois State University

Kevin is the Program Coordinator for the Engineering Technology major at Illinois State University. His primary teaching assignments are in engineering graphics, industrial robotics, and CNC programming/machining.

Mr. Gunnar Keith Klitzing, Illinois State University

Exploring the VR-Based Programming-by-demonstration Approach to Teach Industrial Robotics in Manufacturing Education

Abstract

Traditionally industrial robots have been programmed on-line using a teach pendant or off-line using programming software. For novice users, using a teach pendant is ideal because the user moves the robot to specific points along the desired tool path, recording key locations in the robot program during the process. Once novice users grasp the basics of robot programming, off-line software is often used to program and simulate more complex robot motion without sacrificing the up time of the robot; points in space are specified and the software generates the tool path automatically.

With the advent of collaborative robots, a third programming approach is now gaining attention. Programming by demonstration (PbD) is often used to program collaborative robots. Using PbD, the user programs the robot by physically holding the robot end effector to move it along the intended path, and the computer records the tool path. Accuracy of the program can be enhanced by revising the location of certain points afterward. While the PbD approach seems to be more intuitive, it requires a collaborative robot. Since collaborative robots are relatively new, few academic institutions have them available for students to use.

In this work in progress paper we describe a study to explore methods of using the virtual reality (VR) add-on for ABB's RobotStudio to allow students to program traditional industrial robots using the PbD approach. Key features of the VR add-on, and how they allow the user to move a virtual robot end effector to generate a tool path is described. A comparison between the conventional approach of robot programming using the teach pedant and the VR-based PbD approach is then presented. The quality of the resulting robot programs and advantages/ limitations of the VR-based PbD approach are then discussed. We will conclude the paper with practical implications in instructional practice.

Introduction

To program an industrial robot, the programmer can describe its movement by using a teach pendant. This on-line teach method allows the programmer to interact with the robot, manually jogging the end effector to specific locations, recording the location changes in time, and assigning corresponding non-motion instructions. Since the robot's physical operations are visible in the entire process, the experienced programmers can notice the existence of coding errors or movement anomaly. However, the teach pendant method could be relatively slow, especially when dealing with complex workpiece geometry [1].

Another way to program an industrial robot is through a computer-aided programming environment such as ABB's RobotStudio [2]. This method can be done off-line, e.g. without the presence of the physical robot. The CAD-like software environment allows the programmer to specify the end effector's spatial positions by referring directly to the work object's CAD model [3] and letting the software determine the rest of the path. While this approach seems to be less

time consuming, the interface could induce a relatively high mental workload [4] and be quite daunting to novice learners. They might focus only on minor issues but overlook the big picture.

Yet another method for industrial robot programming is through programming by demonstration (PbD) [5]. By tracing the programmer's trajectory, the robot controller can further analyze and reduce the noise from human demonstration [6] to improve the robustness of the resulting path. Although this method is intuitive, it is not as popular as the aforementioned methods due to its need of special hardware to detect human movement through camera or force sensors and optimize the robot path. Additionally, the safety of the programmer or the operator in such a closed-range, collaborative environment has been a concern of the manufacturing industry [7].

Recent breakthroughs in visual computing technology, especially in the virtual reality (VR) area, might help address these known concerns and improve the novice programmer's learning experience. The VR add-on for ABB RobotStudio provides an immersive virtual environment that allows its user to interact with the virtual robot directly and thus approach robot programming in a holistic way. In a manner similar to PbD, the user can use the VR headset's controllers to grab the virtual robot's end effector and drop it at intended locations. The path is updated instantly along with the user's adding, removing or changing locations in the virtual space, achieving what-you-see-is-what-you-get (WYSIWYG). As the robot's movement is verified through inverse kinematics, the path from a VR session may be used with minor modification in the conventional desktop environment (in this case, ABB's RobotStudio).

While using the VR add-on for learning industrial robot programming seems promising, there is a lack of how-to documents. A team of faculty and student researchers at Illinois State University have been testing the functionality of the VR add-on since 2018. The purpose of this paper is to share our experience on the setup of the VR environment in RobotStudio, the evolution of the add-on's functions and limitation, different applications of the VR add-on in the classroom settings, and ongoing in-house testing of new features.

VR for Industrial Robot Programming

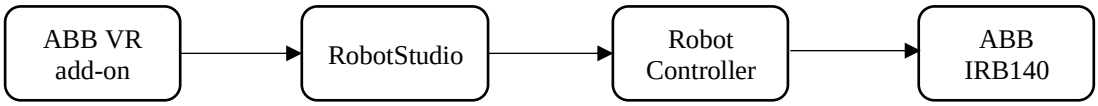

Figure 1. The VR environment for ABB's RobotStudio.

Figure 1 highlights the setup of the VR environment. Being part of ABB RobotStudio, the VR add-on serves as the human-computer interface and provides the immersive environment for the user to interact with the virtual robot. The compatible VR headsets include HTC VIVE, Oculus Rift, or Microsoft HoloLens 1. The former two systems are tethered to the computer for rendering and require two infrared sensors (e.g. lighthouses) to determine the user's spatial location, movement, body posture, and controller gesture. A physical space free of obstacles is needed to ensure natural yet safe navigation in the virtual space. HoloLens 1, on the contrary, is an inside-out device, meaning that it does not require additional spatial sensors. It is untethered and the user's interaction with the virtual object is through finger gestures. Since the virtual

object is projected to the glass hull in front of the user's eyes, the field of view is somewhat limited and more suitable for simple tasks such as selecting and moving virtual objects.

We have tested three computer systems with both HTC VIVE and Oculus Rift. Two of these systems are an AMD-based Dell desktop, Inspiron 5676, and a Dell all-in-one computer, Inspiron 27-7775, both equipped with a Ryzen 7 CPU and an 8GB AMD Radeon RX 580 GPU. The third system is a Lenovo Legion Y720 laptop, equipped with an i7 CPU and a 6GB nVidia GTX 1060 GPU. With the included 16GB RAM and a VR-ready graphic card, all three systems work with the latest version of ABB RobotStudio (2019) in the Windows 10 Enterprise environment without any issue. However, a computer system with less RAM and Video capacity might not be able to provide a similar VR experience in terms of detail and fluency.

Figure 2. The VR test environment: ABB robot work cell (left); the VR add-on in action (right).

Figure 2 shows the robot work cell used for this study, and a faculty member was testing the VR add-on in Oculus Rift. A lighthouse was placed on the left of the screen, which showed what the user was seeing in the VR headset. It is worth noting that only a portion of image was displayed on the screen; the actual field of view within the headset is much broader. Besides, neither the current VR add-on nor the current headset provide eye movement detection. The image on the screen is rendered based on the headset's orientation.

The VR add-on in RobotStudio 2019, the latest edition of this software, allows the user to drag-and-drop the virtual end effector in target locations, specifying traveling speed and zone size (target tolerance). The resulting path from a VR session can recorded in RAPID, ABB Robotics's programming language, and the end effector's non-motion operations (for example, open gripper, turn on the drill, etc.) can be added in the desktop environment. The end program can be simulated in the desktop environment with the collision detection function turns on or off. The presence of a physical robot is not needed up to this point. The end program can be loaded to the work cell reported by Devine and Reifschneider [8]. A dry run of a physical robot is still required to ensure the proper mapping from the virtual work cell to the physical work cell and accessories (conveyor belt, jig and fixture, etc.).

Figure 3 depicts the screenshots of the user interface within the VR add-on. As shown in Figure 2, both VR headset controllers are utilized by the user. By default, the right-hand controller is used for drag-and-dropping the virtual end effector to the desired location, and the left-hand controller is used for carrying the "command cube", a cube that the user can flip along the controller axis and choose buttons at three sides of the cube for different functions (programming, simulation, navigation, axis lock, annotation, etc.). Once the end effector is selected, the user raises his or her left-hand controller and use the right-hand controller to rotate to the cube to where the desired function is and point-click the button to assign non-motion operations. In addition, the user can navigate the work cell by simply "walking around", changing his or her body posture to get closer or stay back from the object to get a better understanding.

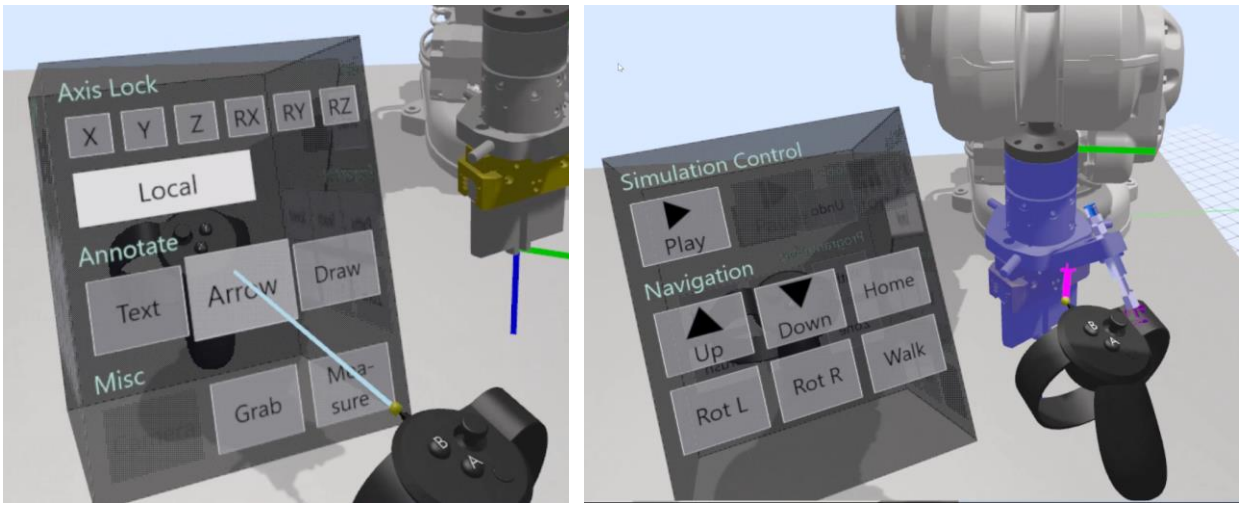

Figure 3. The controller-based interface in ABB's VR add-on: Selecting the "Arrow" button during the creation of an annotation (left); and changing the user's viewpoint, the magenta line linked to the highlighted end effector (right).

Figure 4 shows an example path of the robot's end effector: Starting from its home position on the top, the robot moved toward the left to grab the yellow disk, turned the bottom of the disk toward the scanner attached to the end of the workstation, and placed the disk down to the right side before moving back to its home position. The tolerance of target locations, whether in the conventional desktop setting or in the setting assisted by the VR add-on, can be set as "fine" so that the resulting path will pass through the exact spot instead of "sweeping by".

The biggest difference between the conventional desktop setting and the VR-assisted setting is that the updated path is displayed immediately in the VR-assisted setting whenever a target location is added, deleted, or altered. The user is informed by sight and less guesswork from him or her is needed. Comparatively, the path modification done in the desktop setting will not be displayed until the user commands the system to do so. The WYSIWYG nature in the VR setting is less demanding during the programming process, therefore the user can pay attention to the task and other details and complete the work faster. In our preliminary study with nine

participants in 2019 [9] comparing user performance and experience in the desktop setting and the VR-assisted setting, a substantial reduction in terms of time taken to complete the task and a noticeable improvement of user experience were observed in the VR-assisted setting, while the program quality in both settings was alike.

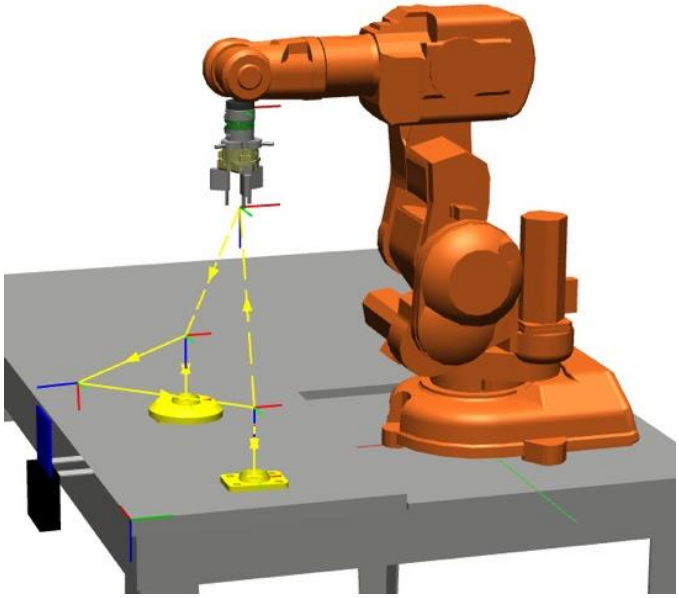

Figure 4. An example path of the end effector (reprinted with permission, [9]).

The VR add-on's Functions and Limitations

Since its first release in 2017, ABB RobotStudio's VR add-on has been refined and updated along with each new release. The function of the initial release in RobotStudio 6.07 was somewhat limited, only allowing the user to navigate the virtual space and jog the virtual end effector around; the trajectory could not be recorded, and the VR add-on was purely for creating the immersive experience. The 6.08 edition released in 2018 came with the basic function to record the trajectory, speed, and zone size. The latest edition of the VR add-on available in 2019 brought the ability to lock the axis movement or rotation, addressing the biggest concern in the 2018 release. The user now can move the end effector with more spatial constraints, reflecting the best programming practice.

Probably the only key function missing in this release is the ability to snap to desired geometric entities (points, lines, surfaces, etc.) The pointer from the controller does recognize and point to some entities such as center points, midpoints, and end points. However, the information of these entities has not been utilized in the VR add-on for easier manipulation of objects. In its current release, the user might find it difficult to create tasks with desired accuracy due to the limitation of the VR headset [10], especially when dealing with tasks consisting a pattern (e.g. repeating the same movement at a set of offset locations as shown in Figure 5).

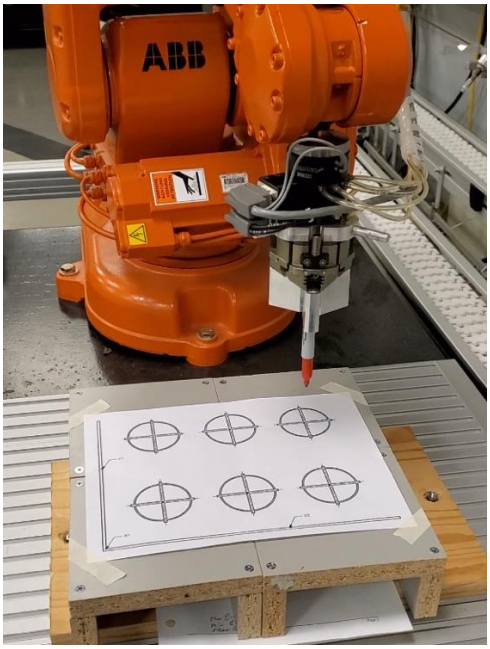

Figure 5. A linear pattern of six target points with the same movement.

One practical matter regarding the VR system's accuracy is that the controller's gesture might not properly represent the actual hand movement, as the "grabbing" of the end effector was calculated with a point on its surface (the magenta pointer in Figure 3) instead of a three-point plane has a normal direction; consequently the end effector might move linearly while the user intends to create a curve path to avoid an obstacle. Similarly, depending on how steady of the user's hand and its orientation during the process of teaching target locations, the local coordinate system of the end effector might not align with the world coordinate system, causing a discrepancy that would be magnified with the distance from the point of reference.

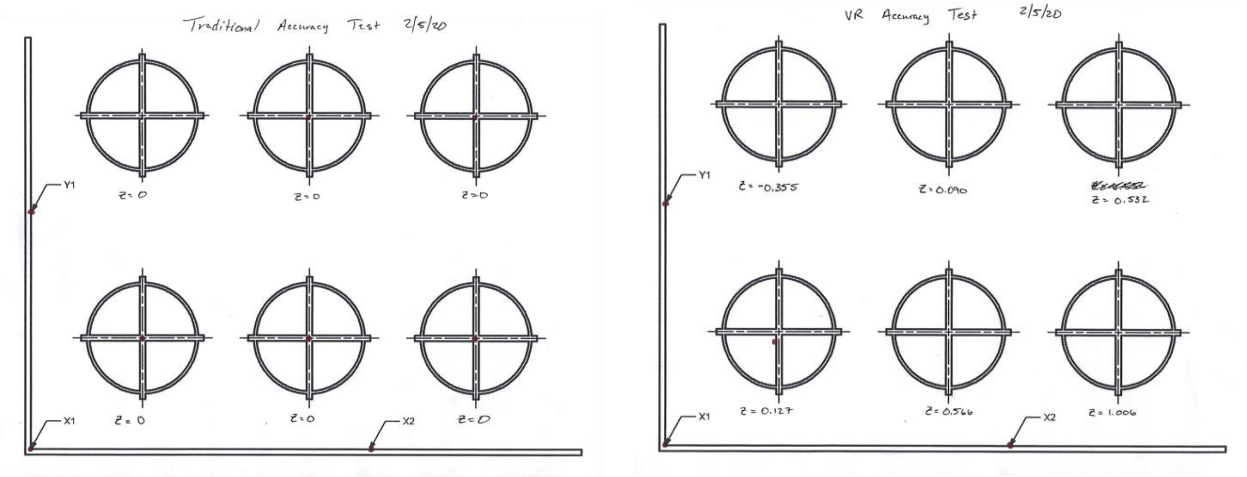

Figure 6. Test outcome on the physical robot with the programs generated in the desktop setting (left) and the VR-assisted setting (right).

Figure 6 shows the test results of teaching six target points in the desktop setting and the VR-assisted setting; these target points were for the red sharpie in Figure 5 to mark on the paper. The resulting programs from both settings were tested with the physical robot. The robot was able to mark all six targets taught in the desktop setting, with a target z value of zero. However, for those targets taught in the VR-assisted setting, only the first point closest to the zero point of the world coordinate system (the lower left corner) was marked successfully. After examining the program created in the VR-assisted setting, the corresponding z values of all six targets, starting from the lower left corner in a counterclockwise order, were 0.127, 0.566, 1.006, 0.532. 0.090, and – 0.355, meaning the other five target locations were hanging in the air. This indicates that the end effector's axes were not fully aligned with those in the world coordinate system, as shown in Figure 7. A further research is needed to address this issue.

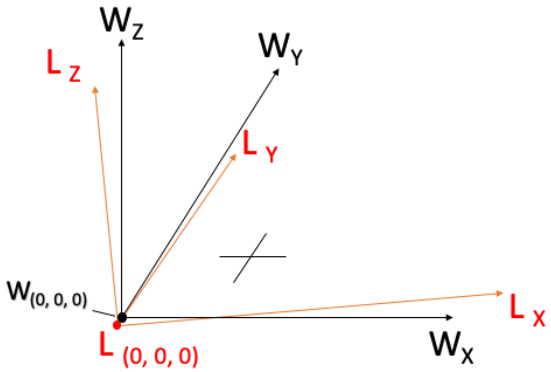

Figure 7. The tilted local coordinate system of the end effector; the origins and axes were not aligned.

Implication of Instructional Practices

With our experiences of using the VR add-on and knowledge of its limitations, we have more confidence now to suggest applications of using VR for the existing TEC234 Robotic Systems Integration course at Illinois State University. In this course students learn about the operation and programming of industrial robots. Students start the semester creating programs using the handheld teach pendant. As the semester progresses, students use RobotStudio to create and simulate programs using the off-line graphics-based environment. The following are three possible scenarios and example exercises.

First, the VR add-on can be used simply for the purpose of *visualization*: Students enrolled in this course can start learning about the robot's movement, end effector operations, and its interaction with the rest of the work cell in the virtual environment. As the virtual robot poses minimum to none safety threat and the simulation can be repeated for multiple times, novice learners will have opportunities to approach the robot in a closed range and observe from multiple view angles. A properly designed activity could guide the students to pick up key concepts or clarify misconception, such as the issues in creating the world coordinate system, specifying zone size, and assigning motion parameters reported by Devine [11].

Secondly, the VR add-on in its current state can be used for *design verification*. Before executing the program on the physical robot, students are required to evaluate their programs by computer simulation with collision detection, identifying any possible issues. A dry run at the physical robot work cell is also required before engaging the end effector with the workpiece. In our experience most of the programming problems were identified during the dry run of the physical robot instead of during the computer simulation stage, as an anomaly in a 3D environment was less likely to be detected through a 2D screen. With the VR add-on, the students can load their program created in the desktop setting and evaluate the simulation in the immersive 3D environment. Such an approach has minimum safety concern and the simulation can be executed easily without affecting the physical robot. It is especially useful for the classroom or lab environment equipped with limited numbers of physical robots.

Last but not the least, the VR add-on can be used for *testing the conceptual design* of the course's final project in a quick turnaround manner. In this project student teams of two are asked to synthesize what they have learned in the entire semester to create a robot work cell for an assigned task. The instruction of the project is implicit and open-ended; student teams will need to consider the task objective and propose design alternatives before settling down with a specific one. In the conventional setting, the most frequently used strategy by student teams is to plan the project by laying down the project components at the work cell and move the robot with the teach pendant to study the layout's feasibility. A program will then be developed from scratch and verified in computer simulation as well as the physical robot's dry run.

If any mistake happens during this process, student teams will need to identify the issue and might need to revise their work cell design or the robot program. The first iteration could take a while to develop and student teams could run out of the time for revision if the front-end planning was not done properly. With the VR add-on, student teams may create a virtual prototype quickly and test out different design alternatives before developing a work cell design in detail. Such a quick-and-dirty approach may greatly reduce the frustration and anxiety. Consequently, the quality of the resulting program can be improved, and student learning can be further reinforced.

Discussion and Future Work

We reported our two-year experience with the VR add-on for ABB's RobotStudio and discussed how ABB's approach was similar to the programming by demonstration (PbD) approach used for programming a physical robot. The appearance and functionality of the VR add-on was described, along with its impact on the user's performance by providing him or her the WYSIWYG ability. The limitation of the VR add-on was presented, and the accuracy issue caused by the VR hardware and human error was deliberated. We concluded our discussion with three possible applications of the VR add-on for instructional considerations.

More work is required in the future before the VR add-on can be fully incorporated into the existing curriculum at Illinois State University. Experiments are needed to better understand the causes of programming accuracy issues, so strategies to minimize or bypass those causes can

be developed. The next generation of consumer-grade VR hardware has been rolled out in the second half of 2019. The design of both HTC Cosmos and Oculus Quest takes the inside-out approach, meaning that the infrared cameras now are attached to these headsets and no lighthouse is needed to detect the user's motion and controller gestures. While this might solve the issues of interference caused by another set of lighthouses that was placed nearby (for example in the classroom setting), how such a new design could affect the already not so good accuracy is to be determined.

The recent update of Oculus drivers allows the in-the-loop user to switch between his or her VR environment and the computer desktop without taking off the head mount device. This could help to solve or bypass certain accuracy issues of the VR add-on, if the user knows how to combine the best practice in both settings (e.g. desktop versus VR add-on) for specific programming tasks. For example, instead of creating the local coordinate system with the VR controller, the user can switch to the desktop mode and his or her VR controller becomes a mouse, which can snap and pick three points on the workpiece edge in the RobotStudio desktop environment instead.

Another upcoming feature important for classroom learning is RobotStudio's ability to hold online conference calls. One user can initiate the conference call and send the meeting code to the other attendee, who will then join the conference by providing the meeting code at the beginning. A VR collaboration session can be held and both parties has the ability to use the controller to point and add annotation. Since the user or the student in the session is fully enclosed by the head mount device, his or her colleagues (or the instructor) can only interpret what is shown on the 2D screen and provide feedback verbally. The conference call function could allow a second individual to participate the discussion in the immersive environment. Nevertheless, whether this function is only limited to one-to-one sessions, or how the bandwidth will be hit if one-to-many sessions are allowed (e.g. a virtual classroom) is still under investigation.

Acknowledgement

The authors would like to express their sincere gratitude for the financial support from the College of Applied Science and Technology and from the SoTL Center, both at Illinois State University.

Reference

[1] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, 2010, pp. 1–8.

[2] C. Connolly, "Technology and applications of ABB RobotStudio," *Ind. Robot Int. J.*, 2009.

[3] R. Holubek, D. R. Delgado Sobrino, P. Košt'ál, and R. Ružarovskỳ, "Offline programming of an ABB robot using imported CAD models in the RobotStudio software environment," in *Applied Mechanics and Materials*, 2014, vol. 693, pp. 62–67.

[4] Y. Chang and K. Devine, "A Tale of the Robot: Will Virtual Reality Enhance Student Learning of Industrial Robotics?" in *2018 ASEE Annual Conference & Exposition*, 2018.

[5] H. Friedrich, S. Mnch, R. Dillmann, S. Bocionek, and M. Sassin, "Robot programming by demonstration (RPD): Supporting the induction by human interaction," *Mach. Learn.*, vol. 23, no. 2–3, pp. 163–189, 1996.

[6] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," *Robot. Auton. Syst.*, vol. 54, no. 5, pp. 409–413, 2006.

[7] B. Matthias, S. Kock, H. Jerregard, M. Kallman, I. Lundberg, and R. Mellander, "Safety of collaborative industrial robots: Certification possibilities for a collaborative assembly robot concept," in *2011 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, 2011, pp. 1–6.

[8] K. Devine and L. Reifschneider, "Agile robotic work cells for teaching manufacturing engineering," in *Proceedings of the 116th American Society for Engineering Education Conference and Exposition*, Austin, Texas, 2009.

[9] Y. Chang, K. Devine, and G. Klitzing, "Can virtual reality enhance user performance and experience by reducing an individual's cognitive workload?" presented at the ASEE EDGD 74th Midyear Conference Proceedings, Norfolk, VA, 2019.

[10] D. C. Niehorster, L. Li, and M. Lappe, "The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research," *-Percept.*, vol. 8, no. 3, p. 2041669517708205, 2017.

[11] K. Devine, "Integrating robot simulation and off-line programming into an industrial robotics course," in *Proceedings of the 116th American Society for Engineering Education Conference and Exposition*, Austin, Texas, 2009.