2019 IEEE/ACM 2nd International Workshop on Robotics Software Engineering (RoSE)

Visualizing the "Hidden" Variables in Robot Programs

David C. Shepherd, Nicholas A. Kraft, Patrick Francis
ABB Corporate Research
Raleigh, NC 27606, USA
{david.shepherd, nicholas.a.kraft, patrick.francis}@us.abb.com

***Abstract*— Programs for one-armed industrial robots include many location-centric statements, such as *Move to location AboveSurface*. Unfortunately, developers struggle to understand static location variables like *AboveSurface*, as mapping their seven-coordinate definitions to the real world is time consuming and error prone. Thus these locations, as well as the paths between them, which constitute the heart of any robot program, are effectively *invisible* to the programmer. To enable effective robot programming this core data must be visualized. There are two potential approaches to visualizing it. The first, visualizing locations one by one (e.g., by automatically moving the robot to each location) is limited to one location at a time. The second, visualizing locations in virtual reality *can* show all locations and paths at one time, but eliminates the ability to interact with the real world, which has many drawbacks. To avoid these drawbacks we propose using high-precision mixed reality to visualize program locations and paths, all while preserving the natural interaction with the real world workspace and robot. In this paper we demonstrate the feasibility of such an approach, sketch a solution, and discuss the advantages and disadvantages.**

I. INTRODUCTION

The cost of purchasing one-armed industrial robots has plummeted in the past decade. Unfortunately, the cost of programming has not, largely due to the fact that experts (i.e., system integrators) must be hired to handle the complexity of programming an industrial robot. To reduce these programming costs, many new and established robot vendors are creating alternative programming environments designed for end users [9]. While these approaches have helped, the vast majority of projects are still implemented by professionals.

What keeps end users from using these environments? During internal user testing of one such easy programming environment at ABB [12], we discovered that locations, specifically the reading, naming, and tracking of locations, posed significant challenges for users. It is clear from the following example location definition why anyone but an expert user would have difficulty reading and understanding the raw definition:

```
CONST robtarget AboveBlock:=[[464.839,
  448.687, 253.033], [0.000,-0.009,
  -0.999, -0.004], [0, 0, 0, 0],
  [9000000000, 9000000000, 9000000000,
  9000000000, 9000000000, 9000000000]];
```

Because of this, when creating small programs users take great care in specifying meaningful names. Yet this solution has clear limits. While location naming works well for toy-sized tasks, where variables like *AboveBlock* are unique, in even slightly larger programs users struggled to create meaningful names. This is not surprising, as variable naming is known to be difficult [2]. Of course, even a well-named location variable does not necessarily help a developer to remember which precise point in 3D space that the location variable represents or help the developer to visualize the paths between the different locations.

What happens when users do not clearly understand the locations and paths in a program? Users create incorrect paths that may damage the workspace, work objects, or robot when tested. Take for instance the simple repeating task of picking up an object from a feeder and placing it in a tray. Imagine that a programmer has just finished a program that places one object in the tray. Because the paths are not visualized, he/she is likely to forget about the final path, the path from the first cycle's end to the second cycle's beginning. Thus, during the beginning of the second cycle, when moving along the path from the tray back to the feeder, the robot collides with the edge of the tray, potentially causing damage. If this path was visualized the user would be more likely to notice the impending collision.

There are two obvious solutions to this issue. First, the developer can issue a command to the robot, instructing it to move to any predefined location. As a result, the physical robot can serve as an oracle for the user. Unfortunately, this approach has several limitations, the most important of which is that only a single location can be shown at a time, which makes reasoning about paths difficult.

We also considered a virtual reality (VR) solution (e.g., [11], [6], [13]). In this approach the entire environment — including the robot, work surface, work objects — is modeled in VR. Locations and paths can be shown in place, eliminating much of the confusion robot programmers usually face. Unfortunately, this solution has a high cost. Modeling real world fixtures, work objects, and work spaces requires significant effort. Moreover, this solution duplicates testing effort, as real-world testing must be completed even after the solution is created and tested in the VR environment. Unless high-fidelity physics models are created, real world factors such as friction between work objects and the robot gripper can have a huge impact on the design of a program. In addition to duplicating effort, this solution requires duplicated physical space, as both the work space and the VR work space require a dedicated physical space.

To avoid the drawbacks of these solutions while achieving

978-1-7281-2249-6/19/$31.00 ©2019 IEEE
DOI 10.1109/RoSE.2019.00007

many of the same benefits, we propose a hybrid solution: location and path visualization in mixed reality (MR). With the recent advances in MR technology, which allow near-millimeter tracking accuracy, we propose overlaying locations and paths onto the user's view of the actual robot station. This approach visualizes much of the hidden information in robot programs while still allowing users to directly manipulate the physical robot to define locations and physical objects to test programs.

II. Feasibility of Proposed Solution

VR technology has advanced sufficiently such that a consumer-grade VR solution can provide near-millimeter accurate placement of objects in 3D space. With a recent VR solution such as the HTC Vive Pro[1] — which includes goggles to view 3D content, tracking beacons to adjust the view according to your position, and a computer with advanced graphics card to ensure near real-time updates — robotic paths can be visualized with high precision.

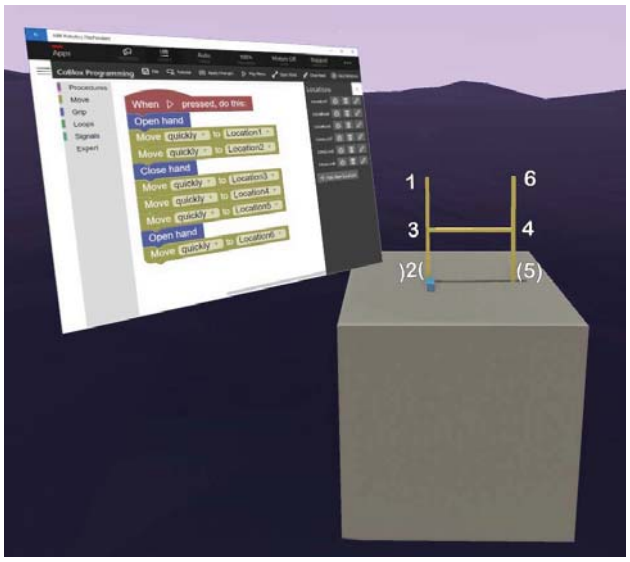

Fig. 1. VR visualization of a robot program, its locations, and its paths

In Figure 1 we provide a mockup that shows two key VR elements of our proposed solution. First, the program itself is shown on the left. Based on CoBlox, a block-based programming language for ABB robots [12], it defines a simple pick and place routine. Second, to its right we show a worktable and visualize the path represented by the program. This VR view has many advantages. It allows a user to create and modify a traditional program such that the control flow information is visible (left) and, at the same time, to visualize the mapping between locations defined/used in the program and their definitions in 3D space. It allows users to view the path from any angle[2], ensuring its correctness and adjusting it naturally via 3D drag and drop. Unfortunately, this view has limitations, such as obscuring the real world objects — i.e., the robot itself and the work objects.

Fig. 2. Viewing a robotic workspace in AR

One potential way of overcoming this drawback of a VR-based solution is to use augmented reality (AR) features to create a mixed reality (MR) solution.[3] In recent VR headsets, cameras have been placed on the front of the goggles. This allows developers to intermingle live video with virtual reality content when re-broadcasting content to the user. In Figure 2 we show an image captured from live video view broadcast from such cameras.[4] As you can tell from this view the fidelity is high enough to clearly view the robot, the workspace, the work object (i.e., the block), and the surrounding environment. This allows the user to interact with the real world as before, positioning the robot and moving work objects with his/her hands.

The next step is to combine these techniques into a single approach, which we have mocked up in Figure 3 and used to illustrate the usage scenario in the next section. This approach allows for physical interaction with the workspace, work objects, and robot arm all while annotating the 3D space with previously hidden information.

III. Example Usage Scenario

Imagine using our proposed solution, shown as a mockup in Figure 3, to perform a single cycle (i.e., no repetition) task where the goal is to pick up an object from the left side of the workspace and then place it on the right side of the workspace. In this example you see the physical robot, workspace, and work objects through the mixed reality view with location and path information on top. On the left, you see a traditional end-user program representation.

To create this program the user would start with an empty program canvas (left). They would add an "Open Hand"

[1] https://www.vive.com/us/product/vive-pro/

[2] See https://youtu.be/lvo71ojOiPE for a brief demonstration.

[3] Augmented reality overlays virtual objects on the physical world, whereas mixed reality also allows the user to interact with the virtual objects.

[4] See https://youtu.be/O74fnL4yEa8 for a brief demonstration.

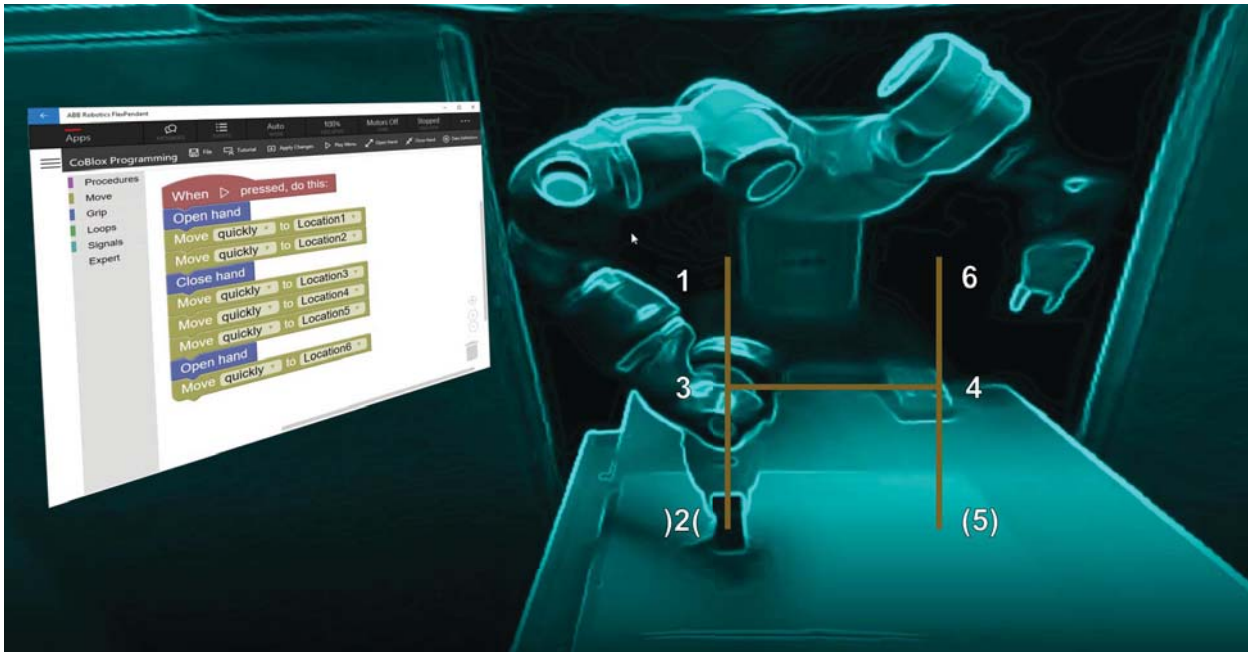

Fig. 3. Using AR to display the existing workspace while overlaying VR content such as locations and paths

statement to this canvas then a "Move to <Location>" statement. At this point the location is undefined. To define the location the user first names his/her new location *Location1* then grabs the robot arm and moves it to the desired location. Once the arm is in place, in this case directly above the block it is going to pick up, the user saves the location and the "1" appears in mid-air directly beside the robot arm. Next the user adds a second move statement, defines *Location2* by moving the arm down onto the block (as shown in the picture), and saves the location. A "2" appears beside the arm and a line is drawn between the two locations to indicate the path. Next the user adds a "Close Hand" statement, which changes "2" to ")2(" to indicate there is a gripper close at that location.

Once the user has completed the entire program it would exactly match the picture that is shown. The program is currently a single cycle program (i.e., it does not repeat), but what if it did, would it be safe? If the robot were set to be in continuous (i.e., repeating) mode, then an additional path would be drawn between the ending position, "6", and the starting position of the next cycle, "1". The programmer could quickly discern that in this program the path between "6" and "1" is clear, and so continuous mode should be safe.

IV. Advantages

A. Liveness

The primary benefit of visualizing locations and paths in 3D space is that it dramatically increases the *liveness* of the programming environment.

> Liveness means the user interface is always active and reactive — objects respond to user actions, animations run, layout happens, and information displays update continuously. [7]

In an integrated development environment, liveness often means variables whose values update constantly, in-line with code, even as the developer is changing it, or a video-game-like visualization that updates as variables are changed.

While there have been many claims that liveness aids programmers, the truth is that most liveness demonstrations target overly specific problems (e.g., changing a programmatically drawn graphic) or awkwardly try to provide liveness for non-visual data, such as variable values. In the first case, while updating a graphic programmatically and seeing results is compelling, the potential use cases are so small in scope that enthusiasm becomes dampened. The second case, providing values for a set of variables, is always beset by issues on what initial values to choose for variables that are not initialized locally, and becomes even more complicated when considering non-primitive variables/values.

In contrast, robot programming provides an obvious place for liveness to succeed. The data is visual, locations and paths, and visualizing them helps programmers to reason about their code. Because we target only paths and locations — a subset of the overall program — we avoid issues with visualizing caused by complex data, such as objects. As those that have used modern debuggers know, objects are notoriously difficult to visualize.

B. Depth

When running robot programming studies with simulated robots (i.e., 3D modeled robots represented on a 2D screen,

such as in the RobotStudio IDE[5]) we consistently received feedback that positioning the robot correctly was one of the biggest challenges. Most of the problems the users encountered were due to depth. While they would position the arm so that it appeared to pick up an object from one side, when moving the viewpoint it became clear that the arm was actually a few inches in front of the object. This type of optical illusion, while common in 3D projection views, is uncommon in real life. Thus, when utilizing an AR view, which provides depth information via its stereoscopic projection of the world, we expect users to avoid these types of issues.

V. Challenges

The biggest challenge to using augmented or mixed reality currently is the quality of the AR content. While the screen size and resolution of VR headsets is now good enough to support detailed visualizations, the cameras we are using to gather AR content (i.e., the cameras on the front of the headset) are not ultra-high resolution. Thus, while VR content appears crisp inside of the goggles the AR content is less crisp. While we are currently working with the best available AR headset for our requirements, we expect that in the next 1-2 years there will be hardware improvements that eliminate this drawback.

Another challenge is that, in order to interact with the VR content (e.g., the program blocks) a handheld controller is required. This makes for awkward switches between grabbing the controller and the robot arm or work objects. Fortunately, this problem is also likely to become obsolete in the next few years as technologies like LeapMotion[6] make gesture-based controls available in AR.

VI. Related Work

We are not the first to propose the use of VR, AR, or mixed reality for robot programming. However, our two key goals in using mixed reality are unique: to visualize location and path variables and to allow the user to interact with the same physical workspace in which the final robot program will be executed.

Foit [3] describes a prototype tool for defining a path in augmented reality and for saving that path definition for later processing by a robot programming environment. The paper raises the issue of poor tracking accuracy of VR/AR headsets at the time (2014) as the major disadvantage of the proposal. Recent advances in consumer grade technology allow for near-millimeter tracking accuracy.

Hoenig et al. [5] provide an overview of mixed reality and list four advantages of mixed reality for robotics: elimination of safety risks, simplification of debugging, addition of (virtual) features to robots, and simplification of experiments with robot swarms. However, they do not elaborate on the advantages that mixed reality can bring to the debugging process, and they are primarily concerned with mobile robots and UAVs.

Guhl et al. [4] propose the use of VR and AR to allow users to plan and test the paths of unsafe industrial robots without being in harm's way. Their focus is on an architecture to permit the use of a Microsoft Hololens to interact with a variety of industrial robots from a remote location. That is, the system proposed by Guhl et al. has the goal of removing the operator from the physical workspace of an unsafe robot, whereas our focus is on planning and testing paths in the same physical workspace in which a collaborative robot is to be located.

Pires et al. [10] focus on allowing the user to create and modify paths using a mixed reality view. Like our technique, this work focuses on visualizing paths, but unlike our work there is no coordination with the complete program. Ostanin and Klimchik [8] focus on using mixed reality to speed up the programming of complex paths, such as those used in welding and painting. This path planning is focused on advanced users, not enabling novice users. Blankemeyer et al. [1] focus on programming by demonstration, where the operator performs the operations in a mixed reality setting, and the operations are linked to a virtual CAD model of the robot and workspace.

[5] https://new.abb.com/products/robotics/robotstudio

[6] https://www.leapmotion.com

References

[1] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer, A. Raatz, Intuitive Robot Programming Using Augmented Reality, Procedia CIRP, vol. 76, pp. 155–160, 2018, doi:10.1016/j.procir.2018.02.028.

[2] F. Deissenboeck, M. Pizka, Concise and consistent naming, Software Quality Journal, vol. 14, no. 3, pp. 261–282, September 2006.

[3] K. Foit, Mixed Reality as a Tool Supporting Programming of the Robot, Advanced Materials Research, vol. 1036, pp. 737–742, 2014, doi:10.4028/www.scientific.net/AMR.1036.737.

[4] J. Guhl, J. Huegle, J. Krueger, Enabling Human-Robot Interaction via Virtual and Augmented Reality in Distributed Control Systems, Procedia CIRP, vol. 76, pp. 167–170, 2018.

[5] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, N. Ayanian, Mixed reality for robots, Proc. of the 2015 IEEE/RSJ International Conf. on Intelligent Robots and Systems, 2015, doi:10.1109/IROS.2015.7354138.

[6] D. Krupke, F. Steinicke, P. Lubos, Y. Jonetzko, M. Goerner, J. Zhang, Co-Located Mixed Reality Human-Robot Interaction, https://youtu.be/X9kGWa5ey7Q.

[7] J.H. Maloney, R.B. Smith, Directness and liveness in the morphic user interface construction environment, Proc. 8th ACM Sym. on User Interface and Software Technology, pp. 21–28, 1995.

[8] M. Ostanin, A. Klimchik, Interactive Robot Programming Using Mixed Reality, International Federation of Automatic Control, 2018, doi:10.1016/j.ifacol.2018.11.517.

[9] Z. Pan, J. Polden, N. Larkin, S. Van Duin, J. Norrish, Review: Recent progress on programming methods for industrial robots, Robotics and Computer-Integrated Manufacturing, vol. 28, no. 2, pp. 87–94, 2012.

[10] J.N. Pires, J. Neves, D. Serrario, Application of Mixed Reality in Robot Manipulator Programming, Industrial Robot, vol. 45, no. 5, 2018, doi:10.1108/IR-06-2018-0120.

[11] RobNor VR robot programming with HTC Vive and ABB RobotStudio, https://youtu.be/WSjgxzgOLLw.

[12] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. Shepherd, D. Franklin, Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices, Proc. of the 2018 CHI Conf. on Human Factors in Computing Systems, April 2018, doi:10.1145/3173574.3173940.

[13] H.J. Yap, Z. Taha, L.J. Vui, VR-Based Robot Programming and Simulation System for an Industrial Robot, International Journal of Industrial Engineering: Theory, Applications, and Practice, vol. 15, no. 3, pp. 314–322, 2008.