Contextual visualizations for debugging collaborative robots

Emil Stubbe Kolvig-Raun
eskr@universal-robots.com
Universal Robots
SDU Software Engineering, MMMI,
University of Southern Denmark
Odense, Denmark

Thor Malmby Jørgin
tjoer21@student.sdu.dk
SDU Software Engineering, MMMI,
University of Southern Denmark
Odense, Denmark

Miguel Campusano
mica@mmmi.sdu.dk
SDU Software Engineering, MMMI,
University of Southern Denmark
Odense, Denmark

ABSTRACT

Collaborative robots, commonly known as lightweight industrial robots, have become indispensable in manufacturing environments. The growing complexity of work cells necessitates the development of improved techniques, methodologies, and tools for their creation, optimization, and debugging. This tool paper explores the application of dynamic visualizations for the debugging process using domain-specific knowledge. These visualizations are tailored for debugging collaborative robots, focusing on pick-and-place applications, and are integrated into a proof-of-concept tool. Demonstrating the tool's utility, we showcase its ability to enable operators to assert the correctness of the robot's behavior and identify program failures using several case studies.

KEYWORDS

Collaborative Robots, Robotic Arm, Visualization, Debugging

ACM Reference Format:
Emil Stubbe Kolvig-Raun, Thor Malmby Jørgin, and Miguel Campusano. 2024. Contextual visualizations for debugging collaborative robots. In *2024 ACM/IEEE 6th International Workshop on Robotics Software Engineering (RoSE '24), April 15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3643663.3643965

1 INTRODUCTION

Collaborative robots (cobots) are lightweight industrial robots with integrated safety mechanisms [12]. These are programmed to operate within a work cell, and manufacturers commonly provide their own proprietary programming languages, making their integration rely heavily on resourceful software strategies. These languages include, among others, URScript, KRL, PDL, RAPID, AS, and KAREL. Generally, software maintenance is recognized as a critical facet of costs. While the figure tends to vary, a general agreement exists that 60% to 80% of a system's budget is spent on software maintenance [4].

Furthermore, previous studies [1, 2] reinforce the need for enhanced debugging tools within the robotics domain and point out that the meshing between software and the physical world necessitates a comprehensive understanding of system behavior. However, despite the established significance of debugging practices in software engineering, their integration into the robotics domain still needs to be explored. This gap arises from the unique requirements imposed by the cyber-physical systems paradigm of robot systems.

Visualization tools provide a graphical representation of code execution, state information data, and the system responses and have been used in research to visualize domain-specific robot data and for debugging complex cyber-physical systems [3, 5, 6, 11]. In the context of debugging, these tools offer a visual roadmap of the system's functioning, facilitating the identification and resolution of issues that may arise during the operation of cobot applications.

Contributions. In this tool paper, we introduce a visualization tool designed for debugging collaborative robots, specifically addressing common challenges in pick-and-place applications involving robotic arms. Our tool utilizes a series of visualizations derived from post-execution data of cobot operations, which are extracted and processed within our tool. Using various visualizations, operators can swiftly assess the accuracy of robotic arm behavior during pick-and-place tasks. We validate our visualizations through diverse pick-and-place scenarios, demonstrating the tool's effectiveness in enabling operators to: (i) assert the correct execution of behaviors and (ii) identify the specific program segments where behavior failures occur.

2 RELATED WORK

The tool presented in this paper differs from previous approaches by presenting runtime information in state resolution, intricately linking the program and its contents to execution. Consequently, this facilitates the presentation of visualizations and pertinent debugging metrics within a more contextually enriched execution framework.

After pivotal contributions like the Robot Operating System (ROS) [10], the adoption of decentralized service-oriented architectures has gained momentum in deploying cobot applications via modular interfaces [9]. Noteworthy tools, exemplified by RViz[1] and Foxglove[2], have emerged within this framework, facilitating the visualization of diverse sensor and state data from robots, commonly used in the pre-deployment phase. Nevertheless, the visualizations crafted in our tool apply to cobots operating outside of the ROS domain.

This work is supported by the Innovation Fund Denmark for the project DIREC (9142-00001B).

RoSE '24, April 15, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0566-3/24/04
https://doi.org/10.1145/3643663.3643965

[1] RViz: http://wiki.ros.org/rviz (Accessed: November 30, 2023)
[2] Foxglove Studio: https://foxglove.dev/ (Accessed: November 30, 2023)

Additional industrial contributions such as Fanuc's Zero Down Time[3] (ZDT) and Universal Robots' (UR) Robot Dashboard[4] software takes this concept a step further by enabling monitoring strategies, i.e., configuring alarm systems to reduce unforeseen halts. However, these do not operate as debugging tools since the objective is not to enhance cobot and program longevity or prevent malfunctions due to flawed implementations, instead centering on streamlining production lines, i.e., defect-free product output, production target, etc.

On the other hand, research contributions like VizRob [3] aspire to comprehend robotic behaviors and offer visualizations rooted in the paradigm of state machines, potentially serving as a debugging tool. However, despite utilizing log traces, VizRob lacks the seamless integration between event information and runtime. This limitation results in a reduced emphasis on domain knowledge.

Furthermore, various initiatives explore the application of augmented or virtual reality (AR or VR) for visualizing data to comprehend robot behaviors [5, 6, 11]. These endeavors concentrate on specific case studies, employing contextual information to debug different applications. However, their primary focus is on the development process, offering limited ways to utilize these visualizations after runtime execution. Leveraging AR or VR for debugging applications at any phase is a promising avenue.

Importantly, with appropriate data conversion, these visualizations have the potential to be integrated into existing tools like RViz, Foxglove, and VizRob. Additionally, integrating with AR or VR could bring significant value to enhance the debugging process further.

3 CONTEXT: EXPERIMENTAL SETUP

Cobot applications encompass diverse types, with material handling constituting 31% and assembly, positioning, and pick-and-place collectively covering 39% in 2022[5]. Our study centers on accommodating the common case, which is picking objects.

The experimental setup, depicted in Figure 1, features a UR5 e-series[6] (UR5e) robot arm, an OnRobot VGC10[7] vacuum gripper, and five 10 by 10 cm tiles with a smooth surface. The UR5e is programmed in URScript to pick up the objects and move them to a designated location.

We tested three execution cases: (i) Successful picking of every object, (ii) Failure to pick two of the objects, and (iii) Successful picking with partial grasp. Figure 1 illustrates instances of successful and partially grasped objects. Even with partial grasping, the behavior is correct, though potentially less desirable for operators due to reductions in the vacuum gripper's effectiveness.

Factors causing a robotic arm to fail in picking objects include incorrect movements, misaligned positions, gripper malfunctions,

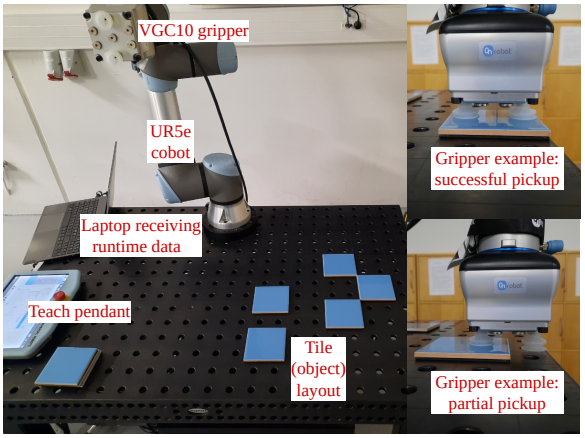

Figure 1: Experimental setup including a UR5e cobot from UR equipped with an OnRobot VGC10 vacuum gripper, displayed with the tiles acting as the objects to be moved around. Explanatory text in red.

and external impacts. Operators armed with comprehensive information can understand and adjust the arm's behavior. However, current cobot application evaluations rely on continuous visual inspection, and new deployments are typically tested across an entire work shift. Our visualizations utilize data from the cobot and its environment, enabling operators to assess failures remotely.

The tool receives runtime data through the push-based Event Driven Data Exchange (EDDE) [8], a research initiative by UR aimed at facilitating event-based logging instead of frequency-based logging. The interface accurately synchronizes runtime data with robot system events, enabling the implementation of introspection strategies across the entire cobot application.

4 IMPLEMENTATION: VISUALIZATION TOOL

The debugging tool is web-based, and we primarily implemented it in JavaScript. The tool and the data used in this study are accessible online[8] [7]. Additionally, we offer a video demonstrating visualizations and interactions with the tool[9]. The focus on the experimental setup drove the development of specialized visualizations tailored exclusively for this context.

Figure 2 depicts the debugging tool's main view, which necessitates an EDDE runtime recording and the executed URScript as input. This section describes the tool's features and visualizations.

4.1 Data resolution and user selection

The EDDE exports an event-based time series [8], associating each entry with a computation step in the robot operating system, providing detailed information at any robot operating state.

Users can select a recorded program cycle, representing a complete program iteration, including locating, gripping, moving, and placing an object. A slider in the bottom left facilitates scrolling through the recorded runtime execution and synchronizes all visualizations upon selection to display the same runtime state.

[3]Fanuc's ZDT: https://www.fanuc.eu/uk/en/iot-solutions/zdt (Accessed: November 30, 2023)
[4]UR+ Robot Dashboard: https://www.universal-robots.com/plus/products/kim-nyholm/robot-dashboard/ (Accessed: November 30, 2023)
[5]Market share by application type:https://www.statista.com/statistics/1044767/collaborative-robots-market-by-application/ (Accessed: November 30, 2023)
[6]UR5 e-series: https://www.universal-robots.com/products/ur5-robot/ (Accessed: November 30, 2023)
[7]OnRobot VGC10 vacuum gripper: https://onrobot.com/en/products/vgc10 (Accessed: November 30, 2023)
[8]The code is available in: https://github.com/thor2304/Robotic_Visualizations
[9]We provide a video showing the visualization in: https://youtu.be/FffedGoe-fU

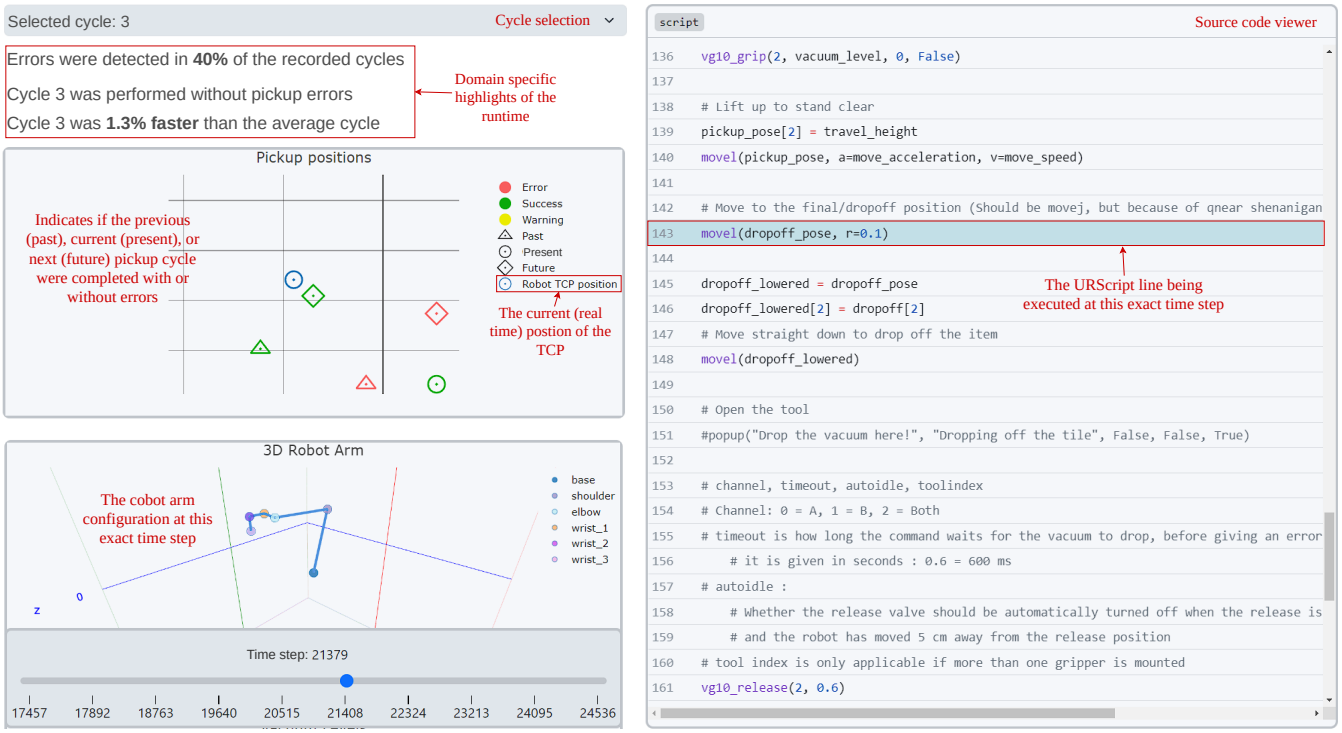

Figure 2: The view of our debugging visualization tool. Explanatory text in red.

4.2 Object interactions and arm configuration

The two visualizations seen on the left side of Figure 2 depict (i) the configuration of the UR5e arm at the selected execution step and (ii) the positions of the objects designated for pickup and whether the run was successful.

(i) The first plot is a 3D visualization highlighting each joint and link and allows the user to interact with and inspect the arm configuration.

(ii) In the second plot, triangular figures represent the positions of the objects in the previous (past) cycles, circles denote the current (present) cycle, and squares present the following (future) cycles. Objects successfully picked are colored green, failed attempts are red and partial grasps are indicated in yellow. The gripper's current position is marked with a blue circular figure.

4.3 Gripper vacuum levels

The OnRobot VGC10 vacuum gripper has two suction cups and is regulated based on a variable specifying a target suction level. The tool facilitates dynamic filter adjustment to assess the suction level against a configurable target. As shown in Figure 3, the tool utilizes this information to indicate whether a grasp was successful (a), partial (b), or a failure (c). Visualizations are annotated with red or yellow overlays for failed or partially successful grasp. The target vacuum level is configured as a program variable in the URScript, and used in line 136 to grasp the object (as seen in Figure 2). As such, the vacuum level is extracted from the event data and incorporated into the graphs (target horizontal line, legend) during runtime.

4.4 URScript line and variable values

The EDDE also facilitates the association of the currently executed script line with each data entry. As shown in Figure 1, the debugging tool displays the URScript source code to the right. The tool highlights the line in blue, indicating the line currently executed in the selected robot operating system step. Additionally, it provides an overview of the values associated with each variable within the program, encompassing configured variables and those imposed on the system from a URCap or end-effector. This is depicted in Figure 3-d.

By associating runtime data with specific script lines and highlighting variable values, the tool helps users quickly identify errors or issues in the program. This empowers them to make targeted amendments and optimize the cobot's performance.

Example. This information, coupled with the vacuum-level data, offers additional insights to users. For instance, if line 136 in the URScript commands the vacuum gripper to grasp an object, and line 161 commands the vacuum gripper to release the object, the data between these two lines should indicate that an object is being picked up. Any deviation from this expected behavior suggests an error or issue in the program execution (*e.g.*, partial grasping), and the visualizations are accordingly marked with red or yellow overlays, indicating problems or cycles with issues.

5 CONCLUSION AND FUTURE WORK

In this work, we introduced an initial iteration of a contextual visualization tool designed for debugging collaborative robots, specifically

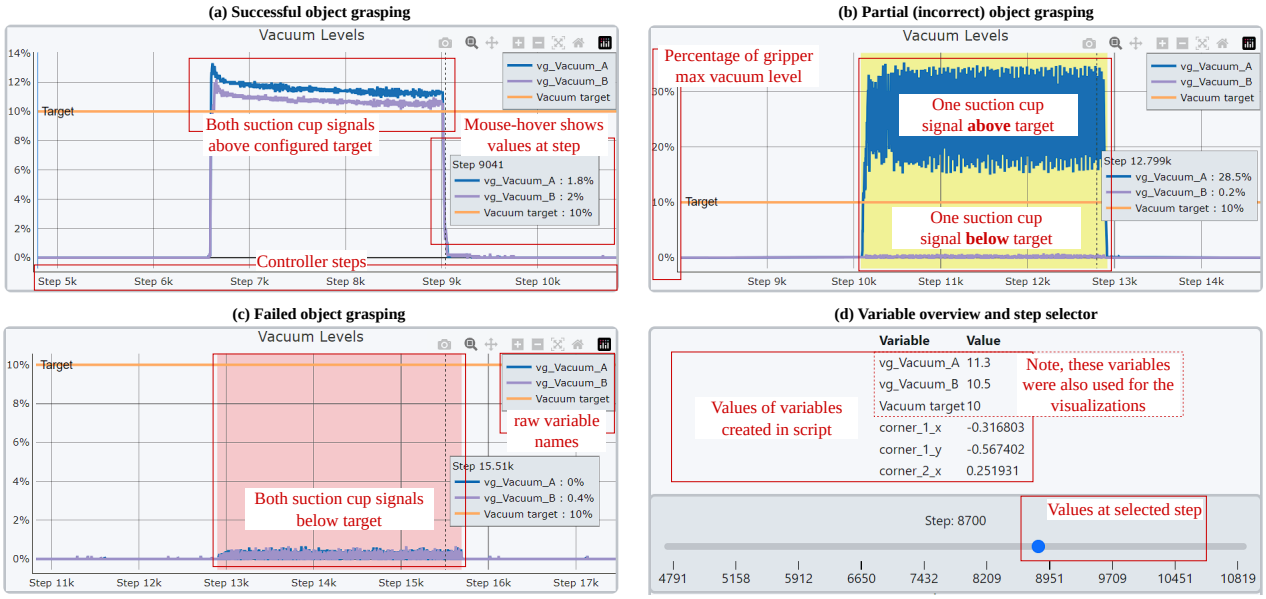

Figure 3: Vacuum level visualization for success, fail and partial grasping. Explanatory text in red.

emphasizing pick-and-place applications. The tool incorporates multiple visualizations that aid users in discerning the success or failure of the robotic arm in picking up objects.

Through an experimental setup, we demonstrate the tool's efficacy in enabling users to comprehend the behavior of robotic arms. Users can swiftly assess whether the robotic arm aligns with the intended behavior specified in the source code. However, it is essential to note that our findings are only indicative, and further validation through user involvement is imperative to substantiate the tool's effectiveness in cobot application debugging.

We aim to explore avenues for extending this tool to debug a broader spectrum of cobot applications effectively. Therefore, in future work, our focus will entail a comprehensive analysis of diverse cobot applications, encompassing their contextual intricacies and runtime data.

Additionally, we envisage delving into the evaluation of script lines by introducing metrics that assess robustness and wear in relation to speed and cycle time. Also, recognizing the potential of debugging tools in continuous monitoring and maintenance, similar to UR+ Robot Dashboard and Fanuc's ZDT, we intend to develop visualizations highlighting temporal trends, offering valuable insights into the system's dynamic behavior, essentially enabling users to craft programs optimized for the robotic system and the work cell to acquire greater reliability in deployments.

We plan to explore methodologies for efficient data management and analysis to address the challenge of handling substantial volumes of data generated during application runs. This is particularly relevant given the considerable data size, exemplified by the pick-and-place scenario, where approximately 74 MB of data is generated in just one minute of execution time.

REFERENCES

[1] Nuño Basurto and Álvaro Herrero. 2020. Data selection to improve anomaly detection in a component-based robot. In *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019) Seville, Spain, May 13–15, 2019, Proceedings 14*. Springer, 241–250.

[2] Tawfik Borgi, Adel Hidri, Benjamin Neef, and Mohamed Saber Naceur. 2017. Data analytics for predictive maintenance of industrial robots. In *2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*. IEEE, 412–417.

[3] Miguel Campusano and Alexandre Bergel. 2019. VizRob: Effective visualizations to debug robotic behaviors. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE, 86–93.

[4] Gerardo Canfora and Aniello Cimtitle. 2001. *Handbook of Software Engineering and Knowledge Engineering: Software Maintenance, pp. 91-120.* World Scientific Publishing Co. Pre. Ltd.

[5] Toby Hartnoll Joshua Collett and Bruce A MacDonald. 2010. An augmented reality debugging system for mobile robot software engineers. *Journal of Software Engineering for Robotics* 1, 1 (2010), 18–32.

[6] Bryce Ikeda and Daniel Szafir. 2022. Advancing the design of visual debugging tools for roboticists. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 195–204.

[7] Thor Malmby Jørgin, Emil Stubbe Kolvig-Raun, and Miguel Campusano. 2023. Contextual Debugging: initial validation. https://doi.org/10.5281/zenodo.10277257

[8] Emil Stubbe Kolvig-Raun, Mikkel Baun Kjærgaard, and Ralph Brorsen. 2023. EDDE: An Event-Driven Data Exchange to Accurately Introspect Cobot Applications. In *2023 IEEE/ACM 5th International Workshop on Robotics Software Engineering (RoSE)*. IEEE, 25–30.

[9] Riccardo Pinciroli and Catia Trubiani. 2021. Model-based performance analysis for architecting cyber-physical dynamic spaces. In *2021 IEEE 18th International Conference on Software Architecture (ICSA)*. IEEE, 104–114.

[10] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, Vol. 3. Kobe, Japan, 5.

[11] David C Shepherd, Nicholas A Kraft, and Patrick Francis. 2019. Visualizing the" hidden" variables in robot programs. In *2019 IEEE/ACM 2nd International Workshop on Robotics Software Engineering (RoSE)*. IEEE, 13–16.

[12] ICS:25.040.30 Technical Committee:ISO/TC 299. 2016. *ISO/TS 15066:2016(en)*. Technical Report. International Organization for Standardization.