Augmented Reality-Assisted Multi-Robot Programming with Collision Warning

Janet Mazur
mazurj@mail.upb.de
Paderborn University
Paderborn, Germany

Enes Yigitbas
enes@mail.upb.de
Paderborn University
Paderborn, Germany

Abstract

Delivery tasks and environmental exploration commonly utilize multi-robot systems. However, programming robots still requires a high level of expertise and knowledge. Therefore, the application of Augmented Reality (AR) has shown promise in aiding robot programming, enabling the user to operate within the robot's space and view robot data and information. Most approaches focus on programming single-robot manipulators or mobile robots. To also offer the programming of multiple collaboratively working robots, we propose EURAPS*, an extension of the existing EURAPS framework. As a feature, we integrate a collision warning to assist the programmer in collision-free multi-robot programming. We conducted a user study to evaluate our system's effectiveness in detecting collisions among mobile robots. The results show that the collision warning assists the user in avoiding robot collisions. Further research is needed to focus on more reliable robot tracking for precise recognition and collision warning with other obstacles.

CCS Concepts

• **Human-centered computing** → **Mixed / augmented reality**; *User studies*; • **Computer systems organization** → *External interfaces for robotics.*

Keywords

Augmented Reality, Multi-Robot Programming, Collision Warning

1 Introduction

In recent decades, the use of robots has increased in various sectors such as industry, healthcare [14] and education [7]. Traditional or collaborative robots (cobots) play a central role in Industry 4.0. Here, robots are used to support production assembly tasks such as screwing, welding, painting, or cutting due to their high durability, speed, and precision [15]. In this context, static robot arms and manipulators are mainly used. Mobile robots, in contrast, are more complex and thus offer a higher variability of application domains such as research and exploration [4], logistics and distribution [21], or delivery [1]. Efficiency can be increased with multi-robot systems, for example, working collaboratively in warehouses [11].

Robot programming is therefore essential, but error-prone, difficult, and requires a high level of expertise and programming knowledge [20]. To address the challenge of complex robot programming, especially for Human-Robot Collaboration (HRC) systems, Augmented Reality (AR) has been implemented in the robotics context. AR can display additional information to the user and assist during robot interaction, for example, by displaying the robot's next move or general robot information [9]. In addition, an AR projection of the robot in the real world can be displayed, for example, with a Digital Twin (DT) providing a simulation of the robot in the real environment [8]. User studies have already shown that these applications are more usable and result in higher user satisfaction [17, 22]. However, most AR robot programming systems do not consider multi-robot programming and only use a single robot. Therefore, we extended our previous work, EURAPS [17], for multi-robot programming with a collision warning. The application runs on a Hololens 2 in combination with two Lego NXT. This paper is a partial presentation of our recent results with a focus on the collision warning system. In this context, we pose the research question: "Does the AR collision warning feature aid in the prevention of robot collisions?" The goal is to analyze the necessary data visualization in AR to support multi-robot programming.

The following Section 2 outlines briefly the existing work. Afterward, Section 3 describes the conception and implementation of our multi-robot programming system in AR with collision warning. In Section 4, we present the evaluation and user study's results. Finally, Section 5 concludes the paper and examines future work.

2 Related Work

Robot programming has been applied for both static robot manipulators and mobile robots for various applications. The following presents an overview of multi-robot programming in AR and collision detection or warning approaches for multiple robots. Chandan et al. [6] proposed an algorithm and framework named *ARROCH* that provides bidirectional communication and programming of multi-robots in indoor environments. The system visualizes the mobile robot's states and planned tasks via an AR interface on a tablet, but the head-mounted display (HMD) is also usable. At the same time, the user can give the robots feedback so they can adjust their behavior. User studies with a delivery task for the robots and puzzles for humans to simulate assembly tasks showed that participants could solve the tasks faster in ARROCH compared to traditional robot control with RViz, a visualization environment for the Robot Operating System (ROS). Furthermore, the participants perceived ARROCH as less distracting while working on their own tasks, more user-friendly, and helpful in keeping track of the robot's status.

Mensch und Computer 2024 – Workshopband, Gesellschaft für Informatik e.V., 01.-04. September 2024, Karlsruhe, Germany

https://doi.org/10.18420/muc2024-mci-ws11-186

Later, Chandan et al. [5] proposed a framework to visualize data in multi-robot systems. The authors used Imitation Learning (IL) to train the robots to dynamically adapt the AR visualization. The user study tested collaboration between a human and multiple mobile robots in a warehouse environment. The results showed increased efficiency in human-multi-robot collaboration compared to their previous work, ARROCH.

As collision avoidance is often a part of robot navigation algorithms, Khan et al. [16] extended such an algorithm without communication and centralized processing to avoid collisions within mobile robots, as the previous algorithm fulfilled collision avoidance but caused deadlocks in dense environments. The idea was to adapt traffic rules when a collision situation arises, rather than choosing the optimum path. Testing the algorithm in simulation shows no appearance of deadlocks.

Others, like Asama et al. [2] propose a collision avoidance algorithm for multiple robots based on rules in combination with communication between the robots with collision warnings.

As communication between the robots is unreliable and computationally expensive, Fan et al. [10] developed a multi-robot navigation system that employs decentralized collision avoidance with Deep Reinforcement Learning (DRL). Each robot perceives its environment using LiDAR sensors, whose data is then processed to create a map and make its own decisions. Through DRL, the robots learn how to interact with their environment and avoid collisions. The learning algorithm was trained in simulation and then tested in simulations and real-life experiments. The results show good performance in avoiding collisions.

In summary, just a few studies focus on programming multiple mobile robots in AR. These studies primarily concentrate on visualizing the robots' states and tasks. These studies do not consider providing the user with additional information during programming, such as collision warnings. On the other hand, the robot's navigation algorithms primarily consider collision detection and avoidance to identify obstacles and modify the robot's path. Collision warnings during the programming of mobile robots are nevertheless not considered. As programming multiple mobile robots in AR is rarely considered and collision detection in such mobile systems is not covered, our system fills the gap by combining multi-robot programming with collision warning.

3 Concept and Implementation

This section outlines our approach to extend the functionality of EURAPS [17] to handle multiple robots with a collision warning system. We first explain the concept and then describe the implementation of the collision warning.

3.1 Concept

As our system extends the previous work EURAPS [17], the basic application components and workflow remain the same. The new components now provide multi-robot programming and introduce collision warning.

The application starts by automatically connecting to both robots via Bluetooth. The image marker defines the world coordinate space, allowing to visualize the AR interface and the Digital Twin of each robot (Fig. 1, 1). The color of the DT distinguishes the robots from each other. Two sliders on the left side of the program bar allow to select one robot for programming (Fig. 1, 2). On the right, the program bar contains the necessary functionalities for programming several robots (Fig. 1, 3). The buttons contain functions such as running the real or virtual robot or the saved program, adding program blocks, saving the program code or resetting the robots. Thus, the user can select one of the two robots, program commands, simulate the DT, and save the program when satisfied.

The saved robot paths appear in green (Fig. 1, 4), while the currently programmed paths appear in orange (Fig. 1, 5). When the user has finished programming one robot, he can move on to the second robot. If a collision within the robot paths is calculated during programming, a red sphere is generated to warn the user of such a collision (Fig. 1, 6a and 6b). Now the user can modify the robot paths until the warnings disappear, thus solving the collision. Finally, the user can simulate both virtual robots in AR or run the programs with the real robots.

To summarize, our framework enables the programming of two mobile robots. This allows two robots to work collaboratively. The collision warning helps the user quickly recognize collisions and program collision-free and safe robot paths.

3.2 Implementation

Our collision warning distinguishes between two possible collisions that can occur during robot programming:

(1) Both robot paths intersect and the robots reach that intersection point at the same time (Fig. 1, 6a)
(2) Both robot paths are too close to each other and the robots reach that close point at the same time (Fig. 1, 6b)

Since the robot path is saved and visualized as line segments with two points in Unity, a possible collision is generally estimated based on the line equations. At first, our algorithm checks for intersecting line segments between the robot paths. Based on the two points from line 1 (x_1, y_1) and (x_2, y_2) and the two points from line 2 (x_3, y_3) and (x_4, y_4), each of the line segments can be defined in a parametric equation with the scalar values t and u:

$$L_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + t \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix} \text{ and } L_2 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} + u \begin{bmatrix} x_4 - x_3 \\ y_4 - y_3 \end{bmatrix} \tag{1}$$

Using these, the intersecting point can be estimated with the following equations 2 and 3 to calculate the scalar values indicating where the intersection lies on the line segments.

$$t = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{2}$$

$$u = \frac{(x_1 - x_2)(y_1 - y_3) - (y_1 - y_2)(x_1 - x_3)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{3}$$

If $0 \le t \le 1$ and $0 \le u \le 1$, an intersection occurs. Otherwise, no intersection lies within the line segments. When an intersection occurs, we estimate the intersection point and calculate the time it takes for each robot to reach it. The time thereby considers the robot's time to drive a line segment, turnings, and claw movements based on the speed. We compare the times at the end and display a collision warning sphere if both robots reach the intersecting point at roughly the same time.

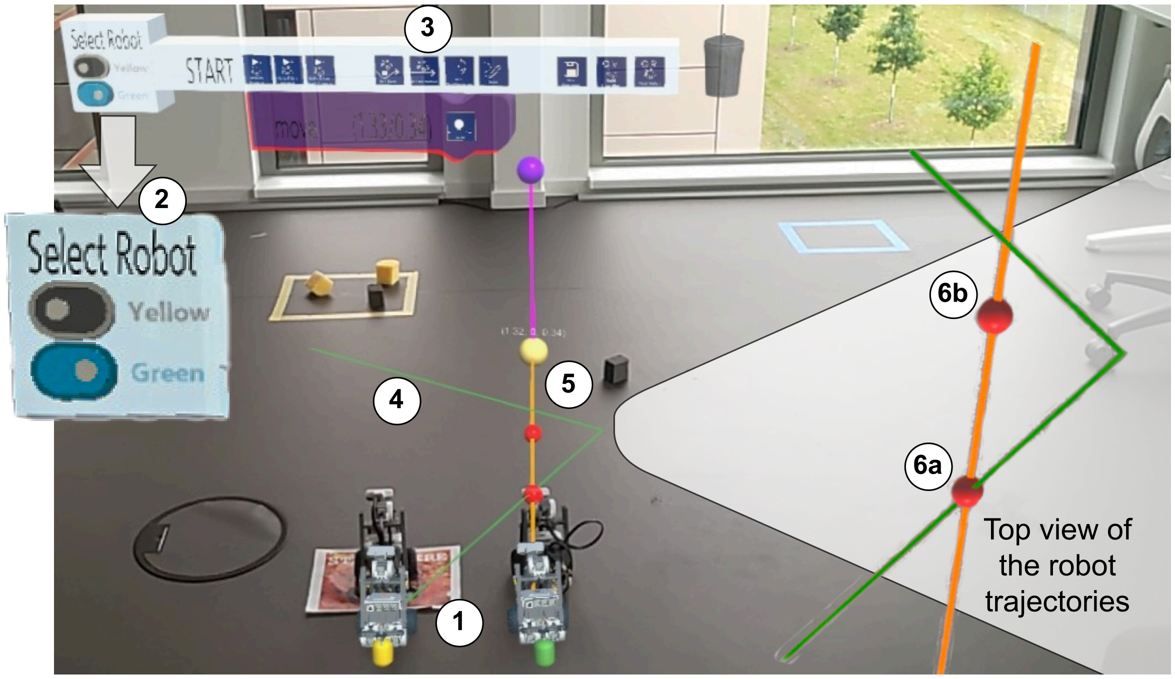

Figure 1: EURAPS* application overview in AR with the two Lego NXTs and their Digital Twins (1), program bar for multiple robots with robot selection (2 and 3), visualization of the programmed robot paths (4 and 5) and collision warning (6a and 6b)

In the second case, if the robot paths are too close, the distance between a point on one path and the line segments of the other path are compared. If the distance is too close and both robots reach this point at the same time, a collision will occur. Therefore, the calculation is based on the point $P(x_1, y_1)$ from the first path and the points $A(x_3, y_3)$ and $B(x_4, y_4)$ from a line segment from the second robot's path. The distance is calculated based on the following equation:

$$d = \frac{|(B - A) \times (P - A)|}{|B - A|} \tag{4}$$

If the distance d between the point and the line segment is too small and both robots reach it at the same time, a collision warning sphere is displayed.

4 Evaluation

A user study was conducted to evaluate the previously described system regarding the research question. In the following, first, the study setup is described followed by the results.

4.1 Study Setup

For the study, we set up two robot programming tasks. To evaluate collision warning, one task was to program two Lego NXT robots in AR without collision warning (2R/wo), and the other task was to program two Lego NXT robots with collision warning (2R/w). The general task was to program a pick-and-place application, where the robots had to pick up an object in the room and place it in an assigned deposit location. The robots were placed in such a way that a collision between the robots would occur in the event of careless robot programming, to evaluate the collision warning feature. An overview of the study setup is illustrated in Figure 1, showing the two robots, two drop-off zones in the back, and objects to be picked up.

We were able to acquire nine computer science students from the ages of 21 to 30 for the user study. No specific limitation was made to their level of experience with general programming, robot programming, or Augmented Reality. Thus, bachelor and master students were invited. Before starting with the tasks, the participants were asked to rate their experience level on a scale from *1 - None* to *5 - Expert* in **general programming**, **robot programming** and **Augmented Reality**. As all participants are computer science students, the general programming experience was rated high between 3 and 5 with a mean and average of 4. In contrast, the experience in robot programming was rather low with five participants answering that they had no previous knowledge. The rest replied with an experience between 2 and 4, resulting in a mean of 1 and an average of 2. The responses about the experience with AR are varied, but half categorize their experience between 4 and 5, resulting in a mean value of 4. Overall, the average is 3.2.

After an introduction, we asked each participant to complete the two tasks in a randomized order to minimize task completion bias. Thus, 4 participants started programming without a collision warning (2R/wo) and 5 participants started programming with a collision warning (2R/w). After each task, participants were asked to answer a study questionnaire consisting of the System Usability Scale (SUS) [3], the raw NASA Task Load Index (TLX) [13], and an additional question, as well as open text fields for positive or negative feedback. The SUS was chosen because it is a common questionnaire for evaluating usability and user satisfaction, while the TLX is widely applied for mental workload analysis. The additional question *A collision warning feature [could have] helped me avoid collisions* was asked as a five-point Likert scale [19] on a scale from *strongly disagree* to *strongly agree*. The question was adapted for the applications without collision warning to determine if participants missed such a feature.

As an additional metric, the number of robot collisions was initially included in the evaluation. Unfortunately, since the application tracking is based on image tracking only and does not include accurate tracking of each robot's position, it was not possible during the user study to estimate real robot collisions due to incorrect programming or inaccurate robot positions. Therefore, this metric is not included in the final system evaluation as it was not meaningful and significant.

4.2 Results

In the following, we will present and discuss the user study results based on usability, workload and assistance of the collision warning feature.

4.2.1 Usability. The corresponding SUS results are shown in Figure 2. The application without the collision warning feature achieved an average SUS score of 84.4, corresponding to a *"A+"* on the scale by Lewis et al. [18]. The collision warning application achieved a similarly high average SUS score of 81.8, which corresponds to a *"A"*. When examining the specific questions in the SUS responses, it is evident that participants who first used the 2R/w application rated questions related to needing help from a technical person, fast learning, and confidence rather low. However, when the task was repeated with the other application, the SUS showed an increase, suggesting that using the applications without prior knowledge is more challenging.

4.2.2 Workload. Similar results are found for the raw TLX shown in Figure 3. While the 2R/wo application has an average of 24.8, the 2R/w application has a higher average of 28.7. However, both are within the acceptable range suggested by [12]. Overall, frustration was rated highest for both applications, as most participants were unable to complete the task of picking up the objects due to inaccurate tracking. In addition, the mental effort question was rated similarly high. Again, the scores decrease when users answer the questionnaire after completing the second task, suggesting that the overall workload decreases as the user becomes more familiar with the application.

4.2.3 Assistance. The box plot in Figure 4 illustrates the distribution for the additional question regarding the assistance of the collision warning. Looking closer at the distribution for the application without the collision warning feature, 50% of participants rated the question that the collision warning feature could have helped them to avoid collisions between 3 and 4 with a median of 4. Two participants ranked this question at 1, indicating no need for a collision warning feature. On average, the participants rated this question 3.3. The application with the collision warning feature has a higher rating, with an average of 4. The interquartile range is between 4 and 5, with 5 representing the mean. Overall, most participants rated the collision warning feature for the application with more visualization between 4 and 5. Just two participants rated this question with 1 and 2.

The results show that the application with the collision warning feature has the highest rating. This indicates that most of the participants found the collision warning feature helpful in avoiding collisions between the two robots. The application without the collision warning feature has a lower rating. This indicates that the collision warning feature was not necessarily needed for the application. This was mainly because some participants had programmed two robots with the collision warning in the previous task and therefore already knew how to avoid collisions. However, some participants managed to program the robot paths without any collisions and did not experience the collision warning feature, thus scoring the question low. Nevertheless, hypothesis testing did not reveal statistically significant differences between the applications for any of the results.

4.3 Discussion

In summary, the results show that most users found the collision warning feature helpful for avoiding collisions. For the application without the feature, most participants would also have liked such a feature to prevent collisions between the robots. The outliers indicate participants who were able to implement the paths without any possible collisions, so they did not need the collision warning feature. Nevertheless, participants had difficulties with the robot's collisions with immobile objects, indicating the need for an additional collision warning feature considering obstacles. At the same time, not all collisions could have been prevented due to inaccurate tracking. In the absence of a collision warning, the lower score may be due to participants' prior familiarity with programming two robots in the previous task, which allowed them to program a collision-free path and not experience the collision warning.

Both applications show a similarly high SUS, although the 2R/wo application has a slightly higher SUS on average. This can be explained by the fact that more participants tested the collision warning application first and found it difficult to use at first. This is also related to the limited robot programming experience and mixed AR experience, so participants had to learn how to use the application. However, with practice, the application seemed to get easier. This is consistent with the text feedback from *P2*: *"Initially, it was hard to get along (no background knowledge), but it was quite easy and well integrated so not that difficult in the end"*. It shows that some practice is needed to successfully program multiple robots in AR. To improve the usability of the application and to simplify the entry point by allowing users to personalize it, future applications could offer different visualizations of collision warnings or turn them on and off.

The workload of the applications was also similar, although the application without collision warning had a slightly lower average and again performed slightly better. However, the individual responses show that users rated the application they used the second time better, which also suggests that the application has a higher workload the first time it is used, regardless of the collision warning.

Participants made suggestions to enhance the overall experience by improving the accuracy of tracking the robot's position. Since the system only sets a visual marker at the beginning of the

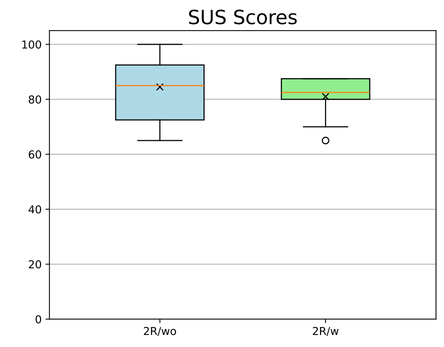

Figure 2: SUS Scores

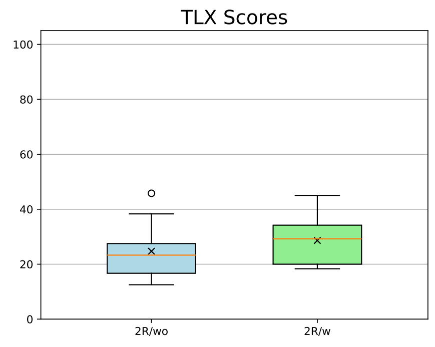

Figure 3: TLX Scores

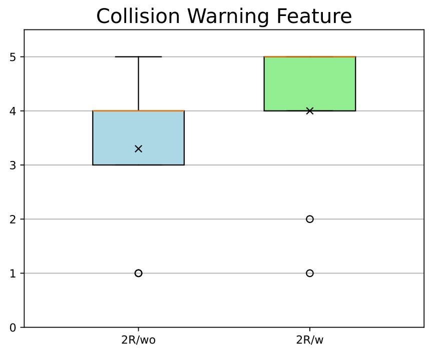

Figure 4: Collision Warning Feature

Figure 5: Box plot diagrams of the user study results for programming two robots without (2R/wo) and with (2R/w) the collision warning feature for the SUS Score, TLX Score and Collision Warning Feature

application, there is no accurate localization of the robots. This makes it difficult or impossible for the robots to interact with the environment. In addition, the AR visualization is not aligned with the environment, so *P9* has been verified that the *overlap of the real and physical world could be more precise*. Future multi-robot programming applications in AR will need to include an accurate robot tracking system. Incorporating robot sensors for environmental sensing or inter-robot communication could further enhance the user experience, improve robot collaboration, and reduce collisions between robots and the environment. In general, however, several participants rated both systems as *easy* and *intuitive*, showing that the application has potential even for inexperienced users and beginners after a short period of practice.

5 Conclusion and Future Work

In this paper, we present our extension of EURAPS* for multi-robot programming with mobile robots and collision warning in AR. Our literature overview revealed that multi-robot programming in AR has barely been covered in research so far, and collision warning has not been covered yet either. We introduced our collision warning feature, which estimates possible collisions based on intersecting or closed robot paths. In our user study, we compared our feature with multi-robot programming with and without a collision warning to estimate the necessity of such a feature. The results showed that the collision warning assisted in avoiding collisions between the robots. In applications without the feature, most participants missed the information.

However, in the future, we need to implement more accurate robot tracking for precise collision detection. It is also necessary to include sensor data to avoid collisions not only with other robots but also with other obstacles. Furthermore, more visualization techniques need to be evaluated and provided to allow for a higher level of system personalization and thus an easier start in using the application.

References

[1] Rasmus Eckholdt Andersen, Emil Blixt Hansen, David Cerny, Steffen Madsen, Biranavan Pulendralingam, Simon Bøgh, and Dimitrios Chrysostomou. 2017. Integration of a Skill-based Collaborative Mobile Robot in a Smart Cyber-physical Environment. *Procedia Manufacturing* 11 (2017), 114–123. https://doi.org/10.1016/j.promfg.2017.07.209 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[2] H. Asama, K. Ozaki, H. Itakura, A. Matsumoto, Y. Ishida, and I. Endo. 1991. Collision avoidance among multiple mobile robots based on rules and communication. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91* (Osaka, Japan). IEEE, 1215–1220 vol.3. https://doi.org/10.1109/IROS.1991.174665

[3] John Brooke. 1995. SUS: A quick and dirty usability scale. *Usability Eval. Ind.* 189 (11 1995).

[4] Wolfram Burgard, Mark Moors, and Frank Schneider. 2002. Collaborative Exploration of Unknown Environments with Teams of Mobile Robots. In *Advances in Plan-Based Control of Robotic Agents*, Michael Beetz, Joachim Hertzberg, Malik Ghallab, and Martha E. Pollack (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 52–70.

[5] Kishan Chandan, Jack Albertson, and Shiqi Zhang. 2022. Learning Visualization Policies of Augmented Reality for Human-Robot Collaboration. arXiv:2211.07028 [cs.RO]

[6] Kishan Chandan, Vidisha Kudalkar, Xiang Li, and Shiqi Zhang. 2021. ARROCH: Augmented Reality for Robots Collaborating with a Human. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3787–3793. https://doi.org/10.1109/ICRA48506.2021.9561144

[7] Kai-Yi Chin, Zeng-Wei Hong, and Yen-Lin Chen. 2014. Impact of Using an Educational Robot-Based Learning System on Students' Motivation in Elementary Education. *IEEE Transactions on Learning Technologies* 7, 4 (2014), 333–345. https://doi.org/10.1109/TLT.2014.2346756

[8] T. H. J. Collet and B. A. MacDonald. 2010. An Augmented Reality Debugging System for Mobile Robot Software Engineers. *Journal of Software Engineering for Robotics* 18-32 (2010).

[9] Morteza Dianatfar, Jyrki Latokartano, and Minna Lanz. 2021. Review on existing VR/AR solutions in human–robot collaboration. *Procedia CIRP* 97 (2021), 407–411. https://doi.org/10.1016/j.procir.2020.05.259 8th CIRP Conference of Assembly Technology and Systems.

[10] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. 2020. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research* 39, 7 (2020), 856–892. https://doi.org/10.1177/0278364920916531 arXiv:https://doi.org/10.1177/0278364920916531

[11] Zhi Feng, Guoqiang Hu, Yajuan Sun, and Jeffrey Soon. 2020. An overview of collaborative robotic manipulation in multi-robot systems. *Annual Reviews in Control* 49 (2020), 113–127. https://doi.org/10.1016/j.arcontrol.2020.02.002

[12] Rebecca Grier. 2015. How high is high? A metanalysis of NASA TLX global workload scores. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 59. https://doi.org/10.1177/1541931215591373

[13] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[14] Jane Holland, Liz Kingston, Conor McCarthy, Eddie Armstrong, Peter ODwyer, Fionn Merz, and Mark McConnell. 2021. Service Robots in the Healthcare Sector.

Robotics 10, 1 (2021). https://doi.org/10.3390/robotics10010047

[15] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. 2012. Cloud Robotics: Architecture, Challenges and Applications. *IEEE Network - NETWORK* 26 (05 2012), 21–28. https://doi.org/10.1109/MNET.2012.6201212

[16] Shehryar Khan, Yasar Ayaz, Mohsin Jamil, Syed Gilani, Naveed Muhammad, Ahmed Qureshi, and Khawaja Fahad Iqbal. 2015. Collaborative Optimal Reciprocal Collision Avoidance for Mobile Robots. *International Journal of Control and Automation* 8 (08 2015), 203–212. https://doi.org/10.14257/ijca.2015.8.8.21

[17] Sarah Claudia Krings, Enes Yigitbas, Kai Biermeier, and Gregor Engels. 2022. Design and Evaluation of AR-Assisted End-User Robot Path Planning Strategies. In *Companion of the 2022 ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Sophia Antipolis, France) *(EICS '22 Companion)*. Association for Computing Machinery, New York, NY, USA, 14–18. https://doi.org/10.1145/3531706.3536452

[18] James R. Lewis and Jeff Sauro. 2018. Item benchmarks for the system usability scale. *Journal of User Experience* 13, 3 (may 2018), 158–167.

[19] Rensis Likert. 1932. *A technique for the measurement of attitudes*. Number 140(22). Archives of Psychology. 1–55 pages.

[20] T. Pettersen, J. Pretlove, C. Skourup, T. Engedal, and T. Lokstad. 2003. Augmented reality for programming industrial robots. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.* IEEE, 319–320. https://doi.org/10.1109/ISMAR.2003.1240739

[21] Chunjie Wang and Dandan Du. 2016. Research on logistics autonomous mobile robot system. In *2016 IEEE International Conference on Mechatronics and Automation*. IEEE, 275–280. https://doi.org/10.1109/ICMA.2016.7558574

[22] Enes Yigitbas and Gregor Engels. 2023. Enhancing Robot Programming through Digital Twin and Augmented Reality. In *56th Hawaii International Conference on System Science (HICSS 2023)*. ScholarSpace.

Received 23 June 2024