ROS-Unity3D Based System for Monitoring of an Industrial Robotic Process

Enrico Sita[1], Csongor Márk Horváth[2], Trygve Thomessen[3], Péter Korondi[2], Anthony G. Pipe[1]

***Abstract*— Planning and monitoring the manufacturing of high quality one-of-a-kind products are challenging tasks. In the implementation of an industrial system, the commissioning phase is typically comprised of a programming phase and an optimization phase. Most of the resources are commonly invested in the optimization of the process. The time and cost of the implementation can be reduced if the monitoring system is not embedded in the industrial process, but kept instead as a decoupled task. In this paper we present a framework to simulate and execute the monitoring task of an industrial process in Unity3D, without interfering with the original system. The monitoring system is made of external additional equipment and is decoupled from the industrial task. The monitoring robot's path is subject to multiple constraints to track the original process without affecting its execution. Moreover, the framework is flexible thanks to the Unity-ROS communication so that the monitoring task can be carried on by any ROS-compatible device. The monitoring system has been applied to a robotic system for heavy, multi-pass TIG welding of voluminous work-pieces. The results of the implementation show that the constraints for monitoring were satisfactory in the 3D environment and capable for real robot application.**

I. INTRODUCTION

For complex industrial processes that produce high quality, one-of-a-kind products, planning is one of the most time-consuming phases. However, during the execution of the process constant and accurate monitoring is necessary to ensure that what has been planned, simulated and tested is accurately reproduced on the real workpiece. The methods of classical automation are therefore not suitable for this kind of production. Processes are difficult to adapt, and the complex commissioning phase prevents companies to react to these market demands in time. Automation and industrial robotics however allow to automate such complex processes while maintaining a high level of flexibility. While there are many different industrial applications that reflect this scheme, [1], [2], [3], in this paper we will focus on the particular processes of heavy grinding and welding as they have been used as test cases for our implementation.

In the industry, for both rigid and flexible automation solutions, the commissioning phase remains one of the most expensive parts of the process [4], [5]. During the commissioning phase the optimization is the most time expensive task, where the process is tuned and calibrated with respect to all the controllable parameters. If the optimization phase was too much time consuming, it would be hardly justified for *one-of-a-kind* types of production since the investment wouldn't be supported by a high-volume production. This is one of the reasons why it is not feasible to embed the monitoring system in the industrial process itself. Moreover the monitoring process is mainly used to optimize the industrial system, and with an independent monitoring solution it can be re-used on other applications once it fulfils its purpose.

The framework described in this paper aims at providing the tools to simulate and test the monitoring strategy with an industrial robot for heavy welding and grinding task, in order to shorten the time of the optimization phase and make the commissioning phase more efficient. Figure 1 shows the current setup for the welding task in our lab.

It is also important to notice that the remote monitoring system serves also as a means to provide external assistance on systems which are fully operational. Therefore, after the commissioning phase a system might benefit from external monitoring to evaluate the product quality even though it was not originally designed to include such a system.

Nonetheless, the framework and the results presented in this work are not to be seen as constrained to heavy welding and grinding processes. In fact, the monitoring system described in this paper is independent of the robotic

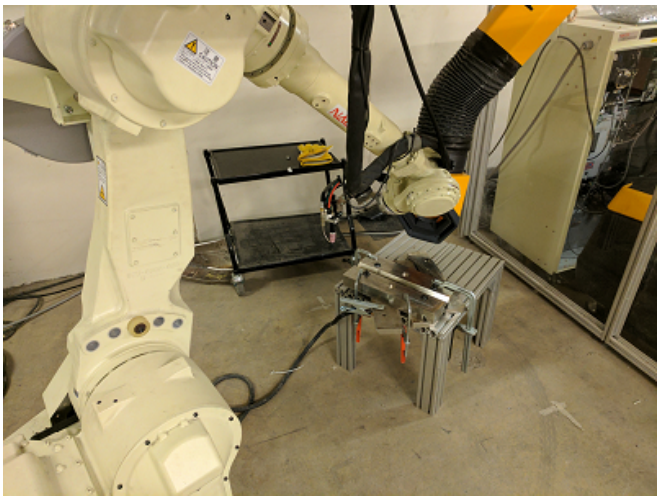

Fig. 1. Lab set-up of the welding task. The welding robot is a NACHI MC70. The actual monitoring robot is not shown in this set-up.

This work was supported by PPM AS and The Research Council of Norway.

[1]E. Sita and A. G. Pipe are with Department of Engineering Design and Mathematics at Bristol Robotics Laboratory, University of the West of England, BS16 1 QY, UK enrico2.sita@live.uwe.ac.uk, tony.pipe@brl.ac.uk

[2]C. M. Horváth and P. Korondi is with Department of Mechatronics, Optics and Mechanical Engineering Informatics, Budapest University of Technology and Economics, 1111 Budapest, Hungary hcsongorm@mogi.bme.hu, korondi@mogi.bme.hu

[3]T. Thomessen is with PPM AS, Leirfossvegen 27, 7038 Trondheim, Norway trygve.thomessen@ppm.no

hardware and also independent of the industrial process being monitored.

Moreover, the recent progress made in the fields of Augmented Reality (AR) and Virtual Reality (VR) is reshaping the future of human robot interactions ([6], [7], [8], [9]).

Multi-modal feedbacks can now be more easily combined to communicate much more information about the real system and also to provide more immersive experiences when we interact with the digital environment [10]. In fact, the gap that exists between the user's actions in the digital world and their effect on the real system is slowly but steadily narrowing. Concretely, the intention to later integrate in our framework multi-modal interactions for man-machine communication is reflected in the choice of Unity [11] as the component in charge of displaying the digital models and interface with the user. Although not the only possible option, the game engine Unity is supported by an active community and a well grounded ecosystem, besides being among the top choices for AR/VR development [11]. In Unity we design the user interface and the monitoring process in a 3D environment before having the task performed by a real robot. In contrast to a simulation software (e.g. Gazebo [12]) Unity3D is not conceived to accurately reproduce real world scenarios in terms of dynamics and physics, but rather to visualize models in a 3D environment and interact with them just as it is the case in games development. The main advantage thus is in the flexibility to design the user interface and the user's interactions with the virtual world.

The second core component is the well known Robot Operating System (ROS), [13] which is used to directly communicate with all the hardware components which are not directly involved in the UI design, such as robots and sensors. ROS allows the framework to interface with any ROS-compatible robot without the need to change the control strategy or the monitoring task.

In this paper we present a framework to simulate the monitoring task of an industrial process in Unity3D. The monitoring process is subject to multiple constraints in order not to interfere with the industrial task. Moreover, we discuss the flexibility of the framework that allows for easy deployment of the monitoring task on a real robot.

II. RELATED WORK

Due to the flexibility we intend to give to our framework, there are many different research contributions that need to be acknowledged and approaches that have to be mentioned.

As previously stated, the industrial process of welding treated in this paper should not pose a limitation in the conclusions we want to draw. The framework is to be seen as independent of the particular robotic task being treated here, but we shall mention related work for similar processes in order to provide better context to the need of a monitoring system for the entire process.

In contrast to high-volume fully automated production systems where monitoring is mainly used for quality control, in smaller batches production systems there is often the need for continuous monitoring. In [14] Pfeifer et al. describe the advantages of an inspection system along a micro-assembly line. Buschhaus et al. present in [15] monitoring of a robot-based process for the metallization of three dimensional molded interconnected devices. The monitoring is crucial in this processes as it serves as in-line correction method in order ensure high quality results. The implementation of a monitoring task for a robot welding application is a continuation of the concept presented in [16]. In their work, Zimber et al. discuss how an autonomous industrial manipulator (AIMM) can be used for monitoring an industrial process. Ultimately, due to the increasing demand from SMEs of remote support solutions for their industrial robotic systems, a monitoring AIMM can significantly improve remote maintenance and assistance. The separate solution brings significant advantages, since it doesn't interfere with the industrial process. In fact, if additional sensors have to be integrated in the industrial process they have to be included in the design process and their presence can redefine the optimization process of the task itself. If additional equipment or additional support tasks are not included in the design process, it is possible that they cannot be integrated at all without an immense investment of resources. The monitoring robot can be programmed also *after* an industrial process optimization phase and it doesn't require modifications in the original system. It serves as a less expensive, quicker-to-integrate external equipment.

For what regards the ROS-Unity communication and its advantages, there are several papers that explored the potential of such connection and that in general investigated the potential of Unity for designing Human Robot Interface (HRI). The work of Bartneck et al. in [17] is one of the first papers advocating the user friendliness of Unity for the design of human robot interaction. One of the main arguments of the paper is that programming robot behaviours and interactions is easier in Unity due to the presence of a set of tools for animations and visual programming used in game development. However, in their work they decide not to involve any middle-ware solution for robotic hardware and implement all the communication logic and HRI within the Unity environment. Other works ([18] [19]) that followed explored further the possibility of using ROS as middle-ware solution, while still managing the HRI in Unity or similar software. The research works mentioned here are in the field of telepresence and teleoperation where multi-modal user interactions are essential. Nonetheless we mainly cite such contributions to highlight the flexibility and modularity allowed by the connection between Unity and ROS. In [20] Codd-Downey et al. proposed an architecture Unity-ROS to control a mobile robot in virtual reality.

Furthermore, Pan et al. [21] proposed an approach for simulating a robotic welding task in Unity. As previously mentioned, Unity is a game engine and not originally thought as a simulation software and therefore lacking proper tools to include the dynamics of the system and accurate hardware parameters. As the authors point out, such system could be beneficial for educational purposes and training.

From a slightly different perspective, we see the Unity

environment as where, besides the user interactions, some of the higher level logic is processed and then communicated to the hardware through ROS or other dedicated channels. Concretely, by receiving real-time information about the state of the system we can animate the 3D environment accordingly and apply constraints on the monitoring robot's motion.

III. SYSTEM DESCRIPTION

The framework can be divided into two parts:

- The industrial process (i.e. the robotic welding task)
- The monitoring process

The system in charge of performing the welding task is based on the work presented by Horvath et al. in [22] and shown in Fig. 2. Furthermore, the welding process data are communicated to ROS and thus made "accessible" to the monitoring process, which is based in Unity and communicating with hardware through ROS.

The general architecture of the whole framework is shown in Fig.3, where the connection ROS-Unity is the main bridge between the different subparts of the system. The monitoring process is directly linked to Unity as it is designed and implemented with the game engine. It is also worth mentioning that the monitoring task is linked to multi-modal man-machine communication to reflect the HMI design process that takes place in Unity. In fact, it is possible to integrate different types of feedbacks (tactile, audio, visual) into the same digital environment without the need of additional software. The monitoring task should be only partly automated, in the sense that the user should have the freedom to adjust the view to his/her needs without having to worry not to interfere with the welding process currently ongoing. In this context, multi-modal feedbacks can increase the user's comfort when he/she takes control over the monitoring robot. Furthermore, the connection with multi-modal communication also reflects our intent to eventually interface the system with VR/AR equipment to investigate immersive telepresence applications.

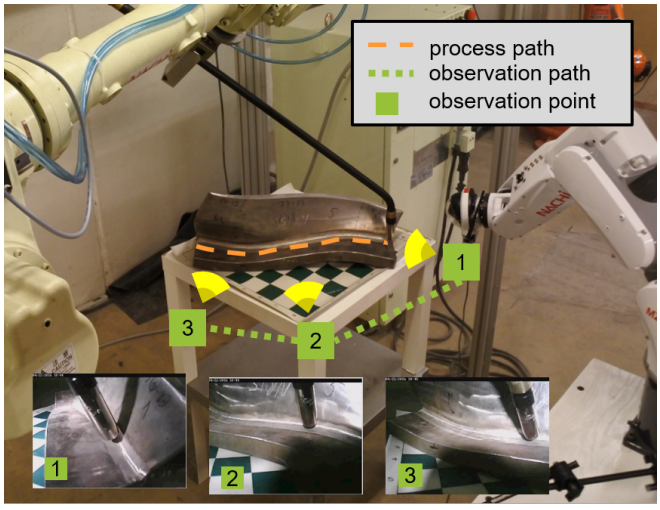

Fig. 2. The set-up of our lab with a welding process (orange dashed line) and the associated monitoring task with way points.

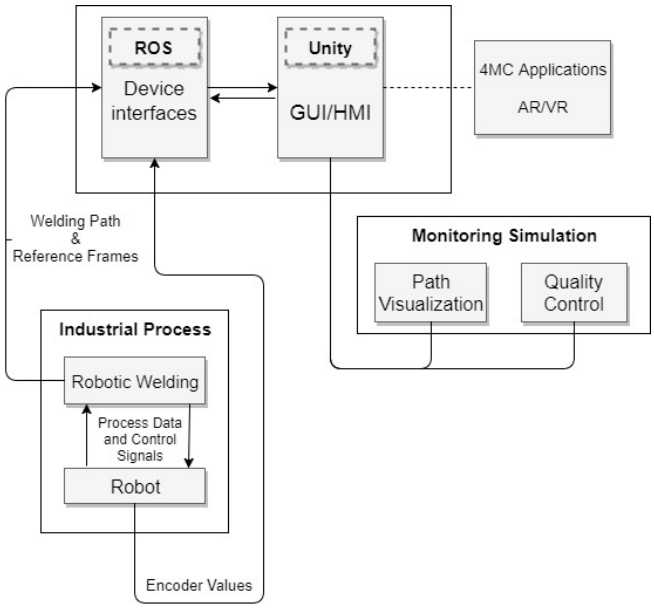

Fig. 3. Layout of the framework presented.

The 3D environment use for the simulation is created by importing the CAD models of the robot and the workpiece involved in the welding process. Such models are then placed in Unity along with the 3D model of the robot used for the monitoring task, as it is shown in Fig.4.

The welding application provides the data about the welding path, which can be displayed into Unity and visualized together with the workpiece. The reference frames of the welding robot, the torch and the workpiece are also communicated by the welding application to Unity through ROS. Once the welding path is available in Unity, the monitoring program calculates the path for the observation taking into account the torch orientation and the welding robot configuration in order to avoid collisions with the system.

The default monitoring strategy consists of simply following the welding process while keeping the welding torch and the part of the workpiece being machined inside the camera's field of view. However, sometimes this strategy is not the most desired one by the user, which should be then allowed to tweak and adjust the camera position if necessary. Therefore the Unity scene allows for user commands that modify the camera position and orientation while tracking the ongoing process.

It is then possible to observe in the 3D environment the welding robot moving according to the real-time joint values provided by the industrial process, furthermore the user can see through the camera view of the simulated monitoring robot and observe how the process while it's performed. The details of the actual implementation of the framework and the monitoring scene in Unity will be discussed in the following section.

IV. IMPLEMENTATION

This section treats the implementation of some sub-parts of the system, mainly regarding the communication between all elements. The last part of this section describes how the animation/control of the 3D model is carried out in Unity.

A. ROS

In the implementation of the system, the communication between Unity, ROS and the welding process is based on the following elements:

- C# Rosbridge for Unity-ROS. This script establishes the connection and allows for invoking Ros services
- Rosbbridge script for Welding process and ROS. This script allows the welding system software to publish data onto topics.
- Ros topics of the welding process. Currently the data published are welding path positions, workpiece reference frame and robot reference frame

Regarding this specific welding process the industrial system doesn't allow for external control commands, meaning that Unity can only fetch the real-time information to synchronize the monitoring task but cannot interfere with the ongoing welding operation.

B. Joint Reading

It is thus important for the monitoring robot to receive at runtime the welding robot configuration, that is its joint values. The welding robot used in our system is a NACHI MC-50, while the robot model used for simulating the monitoring is a NACHI MZ-04 and each robot has 6 DOF. Although the welding robot is ROS-compatible, in our implementation we exploit a different communication channel to receive the robot's encoder values at runtime. In fact, we use a Rasperry-Pi connected to the robot to read the encoder values through UDP communication. The implementation details of such device are not the subject of this paper, but for the sake of clarity Unity receives precise encoder values of the welding robot through UDP communication. Even though the device allows for joint control, as previously mentioned this capability is disabled for Unity since the monitoring process is not allowed to control the welding task. The encoder values received in Unity are then converted into joint angles ([rad]) with double float precision. The conversion is defined by the formula:

$$\theta_i = \Theta_i^{offset} + (Enc_i - Enc_i^{offset})/\pi_i \qquad (1)$$

where i is the joint index, Enc_i is the encoder value of joint i received through UDP connection and the parameters Θ_i^{offset}, Enc_i^{offset} and π_i are constants obtained from the robot controller beforehand. The update frequency of the encoder values has a limit of 5ms (both for writing and reading), which limits the Unity maximum frame rate to 200 fps. However, such a limit is more than enough for real time application and does not constitutes a bottle neck in our system.

In the 3D simulation we display the robotic cell thanks to the CAD models obtained from the industrial process. The digital welding robot is synchronized with the joint values coming from the real robot. We assume that the monitoring robot is equipped with a camera mounted on the end effector and we use the end effector's reference frame for the camera's orientation.

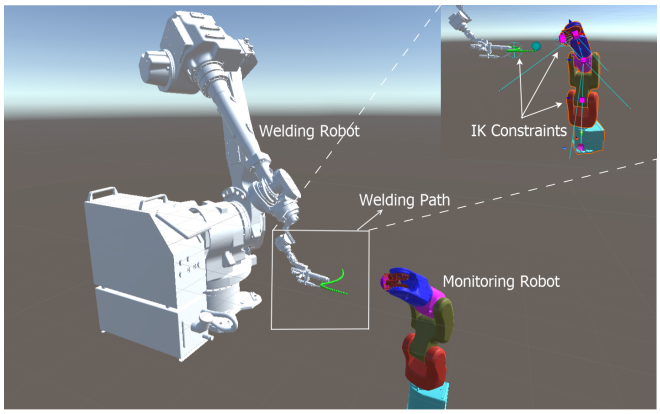

Fig. 4. The Unity environment and the 3D visualization of a welding path. The 3D model of the object has been hidden to better visualize the path. In the detail it is shown the monitoring robot with the visualization of the IK solver constraints.

C. Animation in the 3D Environment

When the welding starts, the monitoring robot starts tracking the welding torch by keeping it in the camera's field of view. The robot is animated via a c# IK solver based on the work presented in [23] and made available as a Unity-plugin.

In addition to the field-of-view constraint, the monitoring robot needs to take into account the following constraints:

- The distance from the torch must not be lower than a certain threshold;
- Avoid collision with the welding robot;
- Keep the welding torch in the centre of the camera;

Each frame, the encoder values received from the Raspberry-Pi are converted into radians and used to update the position of the 3D model. Then, the monitoring algorithm enforces the constraints on the 3D model of the MZ-04 and then compares the newly calculated joint values with the ones of the previous frame. If there is a difference between the two frames it proceeds by performing the inverse conversion to obtain the corresponding encoder values that the actual robot should reach. Finally, the new encoder values are sent through UDP to the Raspberry-Pi connected to the MZ-04. This process is summarized in Algorithm 1.

In Algorithm 1, n_{dof} is the number of joints of the monitoring robot. Moreover, it is worth observing how Step 6 of the algorithm implies that the IK-solver modifies the pose of the 3D model, according to the objectives that are active in that frame.

V. RESULTS AND DISCUSSION

The system has been evaluated in a simulation conducted entirely in Unity. The model of the welding robot was programmed to move along a test path (see green line in figure

Algorithm 1 Joint conversion and update in Unity

Input: MZ-04 Encoder Values
Output: Unity-generated Encoder Values
Encoder Reading and Conversion
for $i = 1$ to n_{dof} **do**
  Obtain θ_i
  Convert θ_i from radians to degrees
  Update the 3D model of the i-th joint
end for
Enforce the IK-solver constraints
for $i = 1$ to n_{dof} **do**
  Obtain the new θ_i in degrees
  Convert θ_i from degrees to radians
  Obtain ENC_i with the inverse conversion
  Store ENC_i in the array ENC_{new}
end for
return ENC_{new}

4, and three main objectives were set on the model of the monitoring robot: welding torch had to remain in focus (look-at constraint); maximum distance between the welding torch and the monitoring robot's end effector; collision avoidance with the welding robot. Every run consisted of the welding robot performing the path once (back and forth), while we observed the behaviour of the monitoring robot.

During the monitoring simulation the robot's configuration could occasionally fluctuate due to the multiple objective optimization. In fact, since the optimization algorithm is based on GA, the robot might move from its current configuration to one with a higher fitness. However, a monitoring simulation is considered successful when the main objectives presented in section IV are satisfied. This means that two successful simulations may have slightly different monitoring paths, but they both accomplish collision avoidance while keeping track of the welding torch and the workpiece. Given a specific instance of the objective, we are not interested in the global optimum within the given search space, but rather a sub-optimal solution in a limited time frame (since the search is computed at run-time).

The main reason why we considered different solutions acceptable is due to our intention to include also commands given by the user to control the monitoring view. Therefore, since in the future the monitoring path will be modified at runtime by the user's actions, the system must allow for some flexibility in the robot configurations.

In the welding task considered in this paper we did not incur situations were one or more of the objectives could not be satisfied. However, it is important to consider such cases to prevent unexpected behaviours from the monitoring robot. In fact, when not all objectives can be satisfied the robot might jump between configuration that optimize different objective that however share similar weights. In order to prevent these fluctuations, we decided to implement an agent in charge of supervising the IK solver at runtime. Concretely, in the event of configurations which do not fulfil one or more constraints this agent will add a special constraint to the optimization function of the IK solver.

The additional objective is called "displacement objective" and its sole purpose is to punish all new configurations found which are "distant" from the current one in terms of joint space. It is important to observe that the agent is also ensuring that the objective are satisfied with the same priority with which they have been listed in the previous section. This is achieved by changing the weights at runtime in a fashion that consistently reflects the aforementioned order.

Thus, with the assumption that the priorities are kept intact, the displacement objective ensures that robot is not "jumping" to a new configuration which is significantly different from the current one, even if the overall fitness of the solution would improve.

The experiments in simulation show that the monitoring robot is capable of tracking the welding torch without specific knowledge of the welding path (the trajectory was only known by the welding robot model). In the bigger framework, it helps proving that such a model-based approach is suitable for remote monitoring of an industrial task.

In this work, the monitoring task has been implemented entirely in simulation, checking that the constraints were satisfied in the 3D environment. However, it is possible to implement the very same simulation on a real robot and this will be part of our future works. The intention is to exploit the UDP communication that has been used to synchronize the system with the industrial process, and use it this time for joint control of the monitoring robot. In this context we will conduct tests to assess the capability of the system to decrease the time for troubleshooting compared to a situation where monitoring was absent.

VI. CONCLUSION AND FUTURE WORK

In this paper we presented a framework for robotic monitoring of an industrial process. The key achievements of this work are the following:

- Remote monitoring system for an industrial robotic process.
- Flexibility of the system due to ROS-Unity communication. The monitoring can be executed with any ROS-compatible hardware.
- Non-invasiveness of the remote monitoring. The parameters of the industrial process remain unmodified and the monitoring equipment can be introduced without compromising the welding task.
- Compact solution to set up a monitoring strategy. The monitoring robot is controlled in the same framework that provides the camera view.

The framework has been used for the planning and evaluation of the monitoring strategy on the welding application. One of the objective is to move toward a shorter set up time thanks to the decoupling from the original process. We are currently running tests in our lab in order to collect more data. The system provides a more flexible compared to an embedded monitoring solution that would have to be designed taking into consideration the welding path and the

welding equipment, and that couldn't be re-used on different installations.

Finally, we aim at extending our framework for multi-modal man-machine communication (4MC) and VR/AR devices for remote monitoring.

VII. ACKNOWLEDGMENT

This research is supported by The Research Council of Norway through the project 245691 Cognitive robot welding system (CoRoWeld), the Industrial PhD project 244972/O30 Virtual presence in remote operation of industrial robot and the Industrial PhD project 264145/O30 Interactive robot operation using multi-modal man-machine communication"

REFERENCES

[1] D. Meike and L. Ribickis, "Energy efficient use of robotics in the automobile industry," in *2011 15th International Conference on Advanced Robotics (ICAR)*, June 2011, pp. 507–511.

[2] A. K. Sethi and S. P. Sethi, "Flexibility in manufacturing: A survey," *International Journal of Flexible Manufacturing Systems*, vol. 2, no. 4, pp. 289–328, Jul 1990. [Online]. Available: https://doi.org/10.1007/BF00186471

[3] F. Jovane, Y. Koren, and C. Bor, "Present and future of flexible automation: Towards new paradigms," *CIRP Annals - Manufacturing Technology*, vol. 52, no. 2, pp. 543 – 560, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0007850607602030

[4] H. Gattringer, R. Riepl, and M. Neubauer, "Optimizing industrial robots for accurate high-speed applications," *Journal of Industrial Engineering*, vol. 2013, 2013.

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

[6] Digi-Capital. (2017) Augmented/virtual reality report q3 2017. Http://www.digi-capital.com/news/2017/01/after-mixed-year-mobile-ar-to-drive-108-billion-vrar-market-by-2021/#.WXYHFYiGNPb. [Online]. Available: http://www.digi-capital.com

[7] S. Tachi, "Embodied media: Expanding human capacity via virtual reality and telexistence (keynote)," in *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ser. ICMI 2016. New York, NY, USA: ACM, 2016, pp. 3–3. [Online]. Available: http://doi.acm.org/10.1145/2993148.3011261

[8] A. Israr, Z. Schwemler, J. Mars, and B. Krainer, "Vr360hd: A vr360° player with enhanced haptic feedback," in *Proceedings of the 22Nd ACM Conference on Virtual Reality Software and Technology*, ser. VRST '16. New York, NY, USA: ACM, 2016, pp. 183–186. [Online]. Available: http://doi.acm.org/10.1145/2993369.2993404

[9] J.-L. Lugrin, D. Obremski, D. Roth, and M. E. Latoschik, "Audio feedback and illusion of virtual body ownership in mixed reality," in *Proceedings of the 22Nd ACM Conference on Virtual Reality Software and Technology*, ser. VRST '16. New York, NY, USA: ACM, 2016, pp. 309–310. [Online]. Available: http://doi.acm.org/10.1145/2993369.2996319

[10] T. Thomessen, M. Niitsuma, K. Suzuki, T. Hatano, and H. Hashimoto, "Towards Virtual Presence Based on Multimodal Man-Machine Communication: A Remote Operation Support System for Industrial Robots," *IFAC-PapersOnLine*, vol. 48, no. 19, pp. 172–177, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896315026531

[11] Unity3d.com. (2017) Unity 3d: Game engine. https://unity3d.com/ and https://unity3d.com/public-relations. [Online]. Available: https://unity3d.com/

[12] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, pp. 2149–2154.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[14] T. Pfeifer and G. Dussler, "Process observation for the assembly of hybrid micro systems," in *Proceedings of 2002 International Symposium on Micromechatronics and Human Science*, 2002, pp. 117–123.

[15] A. Buschhaus and J. Franke, "Industrial robots accuracy optimization in the area of structuring and metallization of three dimensional molded interconnect devices," in *Robotics in Smart Manufacturing: International Workshop, WRSM 2013, Co-located with FAIM 2013, Porto, Portugal, June 26-28, 2013. Proceedings*, 2013, pp. 179–190.

[16] F. Zimber, C. M. Horváth, T. Thomessen, and J. Franke, "Control strategy for an industrial process monitoring robot," in *2016 IEEE/SICE International Symposium on System Integration (SII)*, pp. 706–710.

[17] C. Bartneck, M. Soucy, K. Fleuret, and E. B. Sandoval, "The robot engine making the unity 3d game engine work for hri," in *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2015, pp. 431–437, iD: 1.

[18] D. Krupke, L. Einig, E. Langbehn, J. Zhang, and F. Steinicke, "Immersive remote grasping: Realtime gripper control by a heterogenous robot control system," in *Proceedings of the 22Nd ACM Conference on Virtual Reality Software and Technology*, ser. VRST '16. New York, NY, USA: ACM, 2016, pp. 337–338. [Online]. Available: http://doi.acm.org/10.1145/2993369.2996345

[19] Y. Hu and W. Meng, "Rosunitysim," *Simulation*, vol. 92, no. 10, pp. 931–944, Oct. 2016. [Online]. Available: http://dx.doi.org.ezproxy.uwe.ac.uk/10.1177/0037549716666683

[20] R. Codd-Downey, P. M. Forooshani, A. Speers, H. Wang, and M. Jenkin, "From ros to unity: Leveraging robot and virtual environment middleware for immersive teleoperation," in *2014 IEEE International Conference on Information and Automation (ICIA)*, 2014, pp. 932–936, iD: 1.

[21] J. Pan, Y. Zhuo, L. Hou, and X. Bu, "Research on simulation system of welding robot in unity3d," in *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*, ser. VRCAI '16. New York, NY, USA: ACM, 2016, pp. 107–110. [Online]. Available: http://doi.acm.org.ezproxy.uwe.ac.uk/10.1145/3013971.3013982

[22] C. M. Horvath, T. Thomessen, and P. Korondi, "Robotized Multi-Pass Tungsten Inner Gas Welding of Francis Hydro Power Turbines." Edinburgh, Scotland, United Kingdom: IEEE IES, June 2017, pp. 1759–1765.

[23] S. Starke, N. Hendrich, S. Magg, and J. Zhang, "An efficient hybridization of genetic algorithms and particle swarm optimization for inverse kinematics," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2016, pp. 1782–1789, iD: 1.