Development of Robot Programming System through the use of Augmented Reality for Assembly Tasks

Wenchao Zou, Chair of Automation Technology, BTU Cottbus-Senftenberg, Cottbus, Germany
Mayur Andulkar, Chair of Automation Technology, BTU Cottbus-Senftenberg, Cottbus, Germany
Prof. Dr.-Ing. Ulrich Berger, Chair of Automation Technology, BTU Cottbus-Senftenberg, Cottbus, Germany

Abstract

The development of compliant robots has enabled automation of tasks such as assembly, but their programming is still a tedious and time-consuming task performed by skilled personnel. To address this problem, a novel method for robot programming through demonstration of tasks using augmented reality is presented in this work. The method involves extracting assembly relationships from CAD models between parts to be assembled and assembly sequence demonstrations through augmented reality. An assembly tree and an action sequence are generated based on human demonstration of the task to be performed. The developed method was implemented in a simulation environment and experimentally validated using a compliant robot. The results indicate that the developed method is intuitive and easy to program the robot by an unskilled operator.

1 Introduction

The use of robots in industries to perform repetitive or tedious tasks is common in industry. However, their application in Small and Medium-sized Enterprises (SME) is still not preferred [1]. This is mainly due to the manufacturing cycles in SMEs which are usually managed in small batches with several variants [2]. This has resulted in handling operations such as pick and place, bin-picking or kitting being performed manually by operators [3]. But the rising labor costs and shorter life cycles have increased the demand for automation in SMEs to remain competitive [4]. The development of compliant robots has enabled automation of such tasks with the flexibility of relocating the robot from one task to another. However, the programming of these robots is still a problem as it is usually time consuming, complex and requires skilled personnel [5].
The conventional offline programming methods such as text based or graphics, although being safe and efficient [6], require an experienced programmer with knowledge of robotic systems. Moreover, in many cases these methods are non-intuitive and cannot be easily applied for complex scenarios [7]. One the other hand the conventional online programming methods such as lead through which although being intuitive has concerns regarding the safety of humans while working together with the robot [8]. Moreover, these methods require that the robot be programmed online by the operator taking considerable amount of time and dependency on the operator skills.
To overcome the problems with the requirements of skilled personnel for robot programming, several approaches have been developed. One such approach is the so called Programming by Demonstration (PbD) method where robots can be programmed in an implicit and intuitive way so that the programmer can get rid of the explicit tedious programming [9]. However, the PbD approaches are usually online where the physical robot learns the demonstrated task, and in many cases, a single demonstration is usually not sufficient to provide information for generating the robot program. This may take considerable amount of time thus making the online approaches not applicable for batch production systems.
To address the above issues, one solution is the use of augmented reality (AR) for robot programming as it is intuitive and easy for humans without requiring any domain specific knowledge. Through the use of AR, the required task to be performed by the robot and objects in the robot workspace can be virtually rendered in real time [10]. Moreover, the program can be generated offline without requiring a physical robot [11] thus relaxing the time requirements for the number of demonstrations to be conducted until the desired task description is achieved.
Several attempts to use AR for robot programming have been done in the past. Initial attempts to use AR were based on visualization of task-specific information during the programming. For example, a monitoring system was developed in [12] to display the magnitude of applied force on the robot gripper. Another system applied AR to display the path planning and gripper actions of the robot [13]. This type of AR application helped the programmer to monitor the task information, but the robot programming was essentially performed using the conventional methods. Another typical application of AR is to perform path planning of the tasks in simulation which are then performed by the real robot in a virtual environment. In [14], a virtual robot was controlled with a fiducial marker for path planning of the physical robot on real parts. In [15], the spatial path planning of the virtual robot was conducted using AR with gestures. These applications had an easy and intuitive way to program robot, but they are not scalable for application to complex tasks such as as-

sembly. AR has also been applied to program robots for part handling tasks. In [16], a pick & place task was implemented using AR with gestures. However, only objects with simple shapes were handled. Another system had also implemented similar task using AR with haptic feedback [17]. However, it involved manipulation of virtual objects via teleoperator is less intuitive and complex.
To overcome the problems with the conventional robot programming methods using AR, in this paper a novel CAD and AR based programming method for simple pick and place and applicable to assembly tasks is proposed. The tasks were performed using CAD models of the objects where the task sequencing was generated by human demonstrations performed using AR. The generated program sequence was tested in simulation and executed on a real robot.

2 Proposed approach

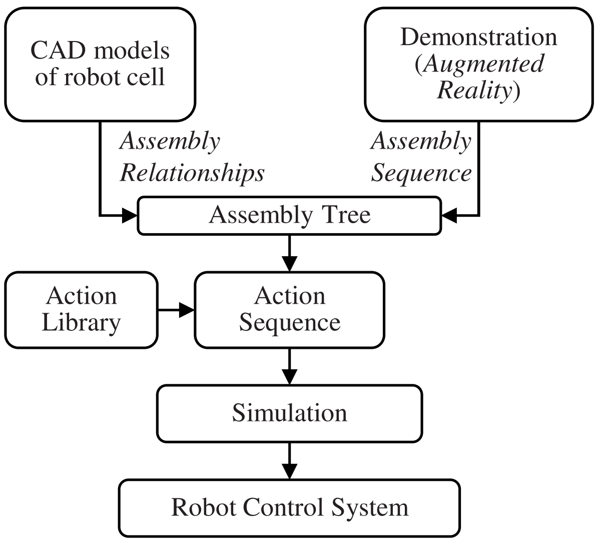

Figure 1 Overview of the proposed approach

The main idea behind developing the proposed method is to extract information from CAD models along with human demonstrations to generate the robot program. Unlike pick and place tasks which involve limited physical interaction between parts, assembly tasks are more complex as they involve physical interaction between two components. To describe the assembly task, in this work an assembly relationship between the parts and assembly sequence has been used. The assembly relationships were extracted from CAD models while the assembly sequence was generated through human demonstration using AR instead of conventional text-based programming. To convert the AR demonstrations into robot program, initially an assembly tree was generated using the assembly relationships and AR demonstrated task sequence. Next, an action sequence was generated using the assembly tree and a pre-defined action library from which the robot program was computed autonomously. The program generated could then be deployed for execution on the robot control system. An overview of the proposed approach is presented (see **Figure 1**). The individual components in the proposed approach are explained next.

2.1 Extraction of assembly relationship

A typical industrial robot application implementation process begins with CAD design and layout [18]. Therefore, the use of CAD models for robot programming is a natural extension of the process and better at connection with design and planning process. However, to generate robot program for assembly task from the CAD model, it is necessary to get the information regarding assembly constraints. But the standard exchange file formats for CAD (such as STL and OBJ file) contain only polygons and the non-geometric information such as assembly relationship are not available. This creates a problem when demonstrating the task sequencing using these models in AR. One solution is to import the parts along with the non-geometric information, but this is not possible with the existing AR environments which support only the standard formats. Moreover, the AR environments have been optimized to work with these polygon-based formats for fast rendering and easy usage. To overcome this problem of importing non-geometric information of the parts in the AR environment, an XML file was generated using a macro created in CAD software SolidWorks.

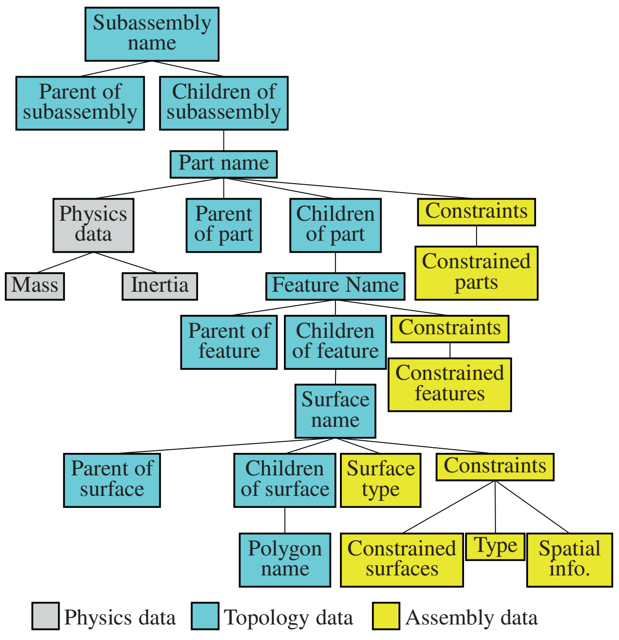

Figure 2 Hierarchy of non-geometric data

The XML file consisted of non-geometric information about the topology, assembly and physics data [19] where topology data represents the hierarchical relationship of the objects that are contained in the CAD models. Physics data represents the physical properties such as mass and inertia of the parts. Assembly data consists of constraints and their spatial information (see **Figure 2**).

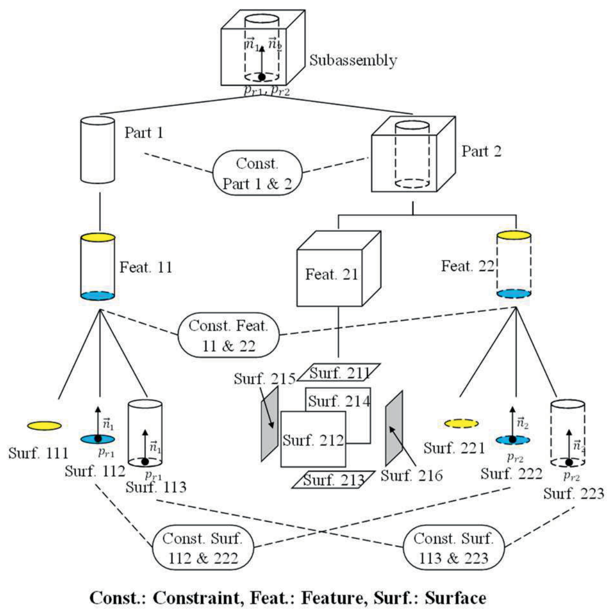

Figure 3 Example of non-geometric data

The file is defined in a hierarchical manner and consists of five levels namely subassembly, part, feature, surface and polygon that together form one product in CAD software. To indicate the hierarchical relationship between the level, parent and child level are defined. Each level, except for product and polygon level, has only one parent but several child levels. A product has no parent level and a polygon has no child levels. The constraints are applied between parts, features and surfaces on the same level. The type and spatial information of the constraints are stored on the level of surfaces and they serve together with the surface type for demonstrating the virtual assembly. An example presented in the above-mentioned hierarchy for the non-geometric data is shown with a subassembly of a cylinder (part 1) and a cuboid with a cylindric hole (part 2) (see **Figure 3**). The right branch of the assembly relationship is as follows: The part 2 has two child features namely cube (feat. 21) and cylinder (feat. 22) in which the cylinder is subtracted from the cube. The feat. 22 has three child surfaces namely two circles (surf. 221 and 222) and one cylinder (surf. 223). Constraints are applied between part 1 and 2, feature 11 and 22, surface 112 and 222, surface 113 and 223.

2.2 Demonstration of assembly sequence

A product can be assembled with same parts in different sequences (see **Figure 4**). Although automatic generation of assembly sequence is possible, its optimization has limitations for example long computational time and human intervention [20]. Therefore, generation of assembly sequence is performed in this work through human demonstration to transfer human process knowledge into robot programming. Based on the non-geometric data as mentioned above, virtual assembly is demonstrated by the human to generate assembly sequence. The main idea of the demonstration is that the human moves the virtual parts directly and assembles them together where the human motion indicates the motion of robot end-effector.

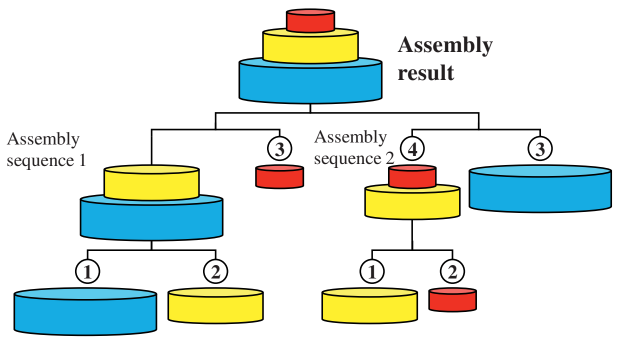

Figure 4 Example of assembly tree

The virtual parts were rendered in 3D space in which their spatial positions were assumed to be extracted from layout of the production process or computed using vision systems [21]. To render the virtual parts in 3D space, a reference frame is required for positioning the virtual objects in the AR environment which could be done using several ways for establishing and tracking the reference frame [10]. Once the reference frame is established, the virtual parts can be rendered in 3D space through transformation with respect to the reference frame.

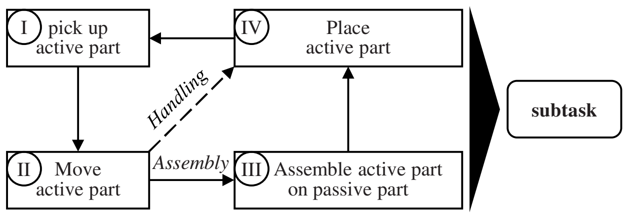

Figure 5 General assembly subtask

The virtual parts and subassemblies are indicated as active or passive during the demonstration. A part or subassembly is indicated as active when human picks it up and manipulates it. A part or subassembly is indicated as passive when it's the target to be assembled with another part or subassembly. When human picks up the active part or subassembly, it moves along with the human motion. When the active part or subassembly is mounted on the

passive part or subassembly or placed on a given position, its motion stops. Based on active and passive part, the assembly process can be generally summarized as a cycle of several steps (see **Figure 5**). One set of the steps is recorded as a subtask.
To avoid the potential collision during the assembly, two motion planning approaches are used for different steps in a subtask. For step I, II and IV of a subtask, the active part is picked up, moved spatially and placed to a given position. During these steps, contact is not necessary between the parts and thus there is less risk of collision. Therefore, human motion is directly used for motion planning of the robot end-effector. For step III, the active part is assembled onto the passive part and thus contact is necessary between the parts. Additionally, relative movement between parts is relatively fine during assembling, but human motion is relatively coarse. Therefore, the extracted assembly relationship between the parts instead of human motion is used for motion planning of the robot end-effector as that the target position on passive part is not changed due to human motion. The switch between the two motion planning approaches is based on the distance between active and passive parts.
Once a subtask is done, the subtask is recorded and updated into the assembly sequence. If an active part (subassembly) is assembled with a passive part (subassembly) in the subtask, the new subassembly from them is stored into this subtask. Once the assembly sequence is completely demonstrated by the user, an assembly tree is generated automatically using the recorded subtasks According to different assembly sequences, the subtasks change respectively. An example can be made with an assembly tree that was presented in two assembly sequences (**Figure 4**). The shown Assembly tree has three subtasks in first assembly sequence and four subtasks in second assembly sequence. In addition, the objects of each subtasks are different.

2.3 Action sequence

To conduct the robot programming using the assembly tree, an action library was developed for expressing the above-mentioned subtasks as a combination of multiple generic robot actions. Using the assembly tree and the library of actions, an action sequence was automatically generated.
The action sequence consists of three levels: high-level, middle-level and low-level (see **Figure 6**). The high-level action sequence consists of the complete sequence subtasks to be executed in performing a given task. Each component (or block) in the high-level sequence consists of robot actions which are taken from action library. These robot actions constitute the middle level and are basically skills such as pick, place, move from one pose to another, task execution, etc. The details of these robot actions fall in the low-level action sequence which is basically the robot program (in robot programming language) executed by the robot controller. For example, a screwing task (high-level) consists of the actions (middle-level): grasping a screw, moving the screw to a nut, screwing the screw into the nut, check the tightness of screwing and release the screw. The grasping action consists of the programs (low-level): approaching the part, opening the gripper, moving to the grasping position, closing the gripper and moving to a given position.

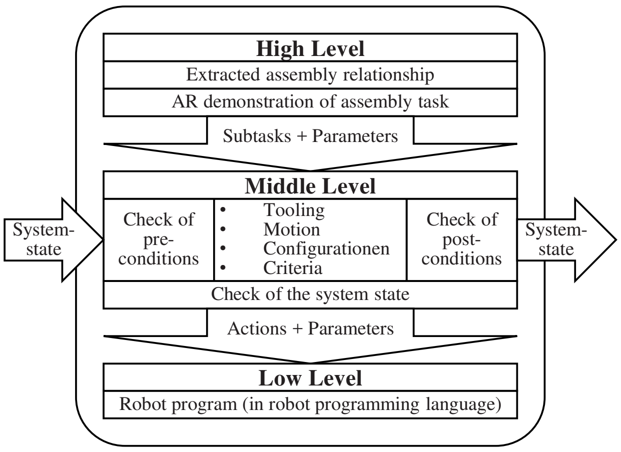

Figure 6 Structure of action sequence

To execute the robot actions, the necessary parameters are derived from the human demonstration. The parameters of the robot actions consist of tooling information, motion data, robot configuration and criteria to build the pre- and post-conditions as well as system state checking. The tooling information consists of the tools that execute the robot actions. The motion data consists the spatial positions and orientations as well as distance of movements that are used for calculation of robot motion. The robot configuration consists the speed, acceleration, force, torque, stiffness and damping that are empirically defined by human. The criteria consist of the key values to be compared when checking of pre- and post-conditions as well as the system state such as positions, force, number for count down, etc.
To check the quality of execution of the robot actions, pre- and post-conditions are defined to indicate the desired system state before and after execution of the robot action. In a generated action sequence, the pre-condition of a robot action matches the post-condition of the previous robot action and its post-condition matches the precondition of the next robot action. The check of system state is throughout the execution to monitor the undesired states such as over force or torque. If the conditions are not fulfilled, the respective predefined reaction or error message will be triggered.

2.4 Simulation and execution of assembly

The generated action sequence is executed first in a simulation. The simulation is built from the layout of the production process and CAD models of robots and end-effectors. In the simulation, the reachability and payload of the selected robot and end-effector is inspected. If the reachability and payload is not enough for the executed tasks, error message will be triggered. In this case, two options are given that are reselecting the robot or end-effector or reporting the error to process planning. If the simulation is performed successfully, the verified action sequence is transferred to the robot controller for actual task execution.

3 Implementation

The implementation of the proposed approach for robot programming through AR demonstration was done. An overview of the implemented scenario is presented (see **Figure 7**). The demonstration was performed by the user with his bare hand to manipulate the displayed virtual parts and assemble them together. After the demonstration, the assembly tasks were executed in a simulation environment to validate them and a physical robot performed the assembly tasks automatically based on successful execution of the task in simulation. The field of demonstration was built on an arbitrary plane that was remotely located from the physical robot and the work station. A fiducial marker was placed arbitrarily on the plane to build the reference frame for AR. An RGB-D camera (Kinect one) was used for hand tracking and positioning of the fiducial marker in 3D space while the AR environment was built using ARToolkit5 [22] with C++.

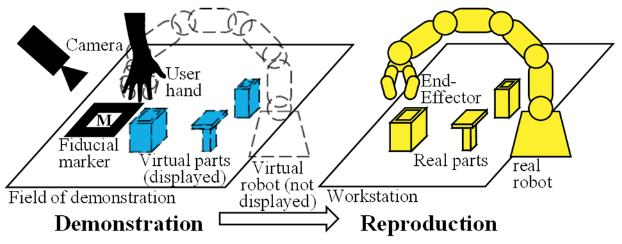

Figure 7 Overview of implementation

The virtual parts were built with the CAD assembly models and the layout of the physical work station was built in advance with SolidWorks. The assembly data including the hierarchy and constraints of the parts and the spatial positions of robot basis, fixtures and parts were exported to an XML file from the CAD models. The XML file served for the location and assembly of the virtual parts in the AR demonstration. The geometric data of each part was exported to an STL file. The STL files served for the rendering of the virtual parts in the AR demonstration. After importing the above-mentioned data in AR environment, the virtual parts were spatially rendered with respect to the fiducial marker. Through a display, the user could see the virtual parts being rendered over the real environment (see **Figure 8** A). The user moved the virtual parts to assemble them together by his hand. As mentioned above, the virtual parts differ into active and passive part. The criteria for deciding of the active part and the passive part was defined using the positions of thumb fingertip, index fingertip and midpoint between them. The assembly constraints were automatically invoked when the active part was moved near the passive part.

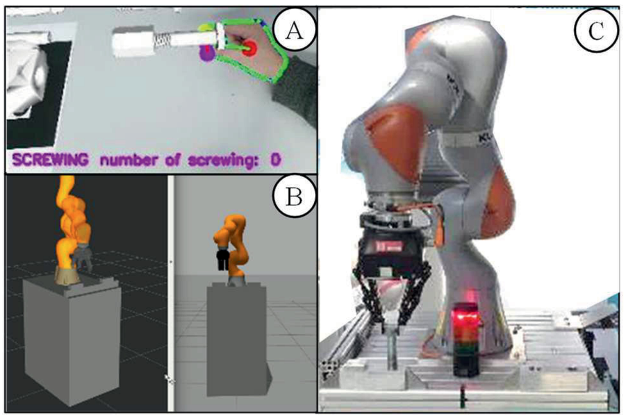

Figure 8 Experiment, A: AR demonstration, B: simulation, C: execution on real robot

An assembly task of screwing was implemented with three assembly constraints between a screw and a nut namely the alignment of their central axes, rotation angle between their central axes and matching of their bottom planes. The hand motion and actions were fully recorded according to the sequence of happening. Once the demonstration was conducted, the action sequence was generated to invoke the pre-defined robot actions in the simulation and the robot controller. The screwing task was simulated based on the generated action sequence using gazebo simulator [23] and later executed on a KUKA iiwa robot (see **Figure 8** B and C).

4 Results and discussion

The implemented screwing task was demonstrated with 10 trials. The average demonstration time was approx. 50 seconds which was relatively short for single task. The robot programming was conducted without textual input or coding but only with demonstration of the given task. The success rate was 90% with 1 failure that was caused by three problems. First, because the camera measuring range is limited, the demonstration will be interrupted when hand moves out of the measuring range. A solution to this problem could be highlighting the measuring range to mention the user about the range for hand movement. Second, because virtual and physical parts differ in looking and other physics properties such as shadow, the user cannot precisely perceive the distance between the virtual parts to notice if collision occurred between the virtual parts. A solution to that could be displaying the distance and collision warning directly during the demonstration. Third, there was only one camera used and thus the tracking and positioning of fingertips could be conducted only from limited view. Therefore, the demonstration would be interrupted when the tracking of fingertips is lost. A solution for this could be the use of multiple cameras from different directions to cover the space of hand movement. In addition, the accuracy of human motion capture is usually insufficient for robot programming because human motion is not as accurate as robot and deviation of measuring occurs during the motion capture. A solution could be a better system that keeps the detection and positioning of human motion stable. It's assumed that layout of the production process is known. However, the layout can have deviations with the real production process and better methods are required to compensate for the uncertainty arising due to the deviations in spatial positions of objects/ obstacles.

5 Conclusion and future work

This paper presents a novel method for programming a robot using AR. The method has been developed to be intuitive and easy to demonstrate in an offline manner. A task of screwing was validated in simulation and experimentally conducted using a KUKA iiwa robot. The presented method inherited the benefits of both online and offline robot programming such as intuitiveness and efficiency. This method could be used for diverse types of collaborative robots. The future work will address the above-mentioned problems and implementation for different assembly tasks.

6 Literature

[1] T. Dietz, U. Schneider, and M. Barho, et al., "Programming system for efficient use of industrial robots for deburring in SME environments," in Proceedings for the conference of ROBOTIK 2012, 7th German Conference on Robotics, VDE Verlag, Berlin, 2012.

[2] D. Grube Hansen, A. A. Malik, and A. Bilberg, "Generic Challenges and Automation Solutions in Manufacturing SMEs," B. Katalinic (Hrsg.), in Proceedings of the 28th International DAAAM Symposium 2017, DAAAM Proceedings, vol. 1, DAAAM International Vienna, 2017.

[3] A. G. Banerjee, A. Barnes, and K. N. Kaipa, et al., "An ontology to enable optimized task partitioning in human-robot collaboration for warehouse kitting operations," D. Popa, M. B. J. Wijesundara, and M. Blowers (Hrsg.), in SPIE Sensing Technology + Applications, SPIE Proceedings, SPIE, 2015.

[4] M. Stenmark, J. Malec, and K. Nilsson, et al., "On Distributed Knowledge Bases for Robotized Small-Batch Assembly," in IEEE Transactions on Automation Science and Engineering, vol. 12, no. 2, IEEE, Piscataway, NJ, 2015.

[5] J. Zhang, Y. Wang, and R. Xiong, "Industrial robot programming by demonstration," in 2016 International Conference on Advanced Robotics and Mechatronics (ICARM), IEEE, Piscataway, NJ, 2016.

[6] R. Holubek, D. R. Delgado Sobrino, and P. Košťál, et al., "Offline Programming of an ABB Robot Using Imported CAD Models in the RobotStudio Software Environment," in Applied Mechanics and Materials, vol. 693, Trans Tech Publications, Switzerland, 2014.

[7] U. Herbst, "Gestaltung eines ergonomischen Interaktionskonzeptes für flexibel einsetzbare und transportable Roboterzellen," Dissertation, 2015.

[8] A. Brunete, C. Mateo, and E. Gambao, et al., "User-friendly task level programming based on an online walk-through teaching approach," in Industrial Robot, vol. 43, no. 2, 2016.

[9] A. Billard, S. Calinon, and R. Dillmann, et al., "Robot Programming by Demonstration," B. Siciliano and O. Khatib (Hrsg.), in Springer Handbook of Robotics, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-540-30301-5_60, 2008.

[10] M. Billinghurst, A. Clark, and G. Lee, et al., "A survey of augmented reality," in Foundations and Trends \ text registered Human-Computer Interaction, vol. 8, no. 2-3, Now Publishers, Inc, 2015.

[11] J. Pan, L. Zhang, and D. Manocha, "Collision-free and smooth trajectory computation in cluttered environments," in The International Journal of Robotics Research, vol. 31, no. 10, SAGE Publications Sage UK: London, England, 2012.

[12] C. Mateo, A. Brunete, and E. Gambao, et al., "Hammer: An Android based application for end-user industrial robot programming," in 2014 IEEE/ASME 10th International Conference 2014, 2014.

[13] B. Akan, A. Ameri, and B. Curuklu, et al., "Intuitive industrial robot programming through incremental multimodal language and augmented reality," in 2011 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2011.

[14] S. K. Ong, J. W. Chong, and A. Y. Nee, "A novel AR-based robot programming and path planning methodology," in Robotics and Computer-Integrated Manufacturing, vol. 26, no. 3, Elsevier, 2010.

[15] J. Krüger and J. Lambrecht, "Spatial Programming for Industrial Robots: Efficient, Effective and User-Optimised through Natural Communication and Augmented Reality," in WGP Congress 2014, vol. 1018, 2014.

[16] J. Lambrecht, M. Kleinsorge, and M. Rosenstrauch, et al., "Spatial programming for industrial robots through task demonstration," in International Journal of Advanced Robotic Systems, vol. 10, no. 5, SAGE Publications Sage UK: London, England, 2013.

[17] J. Aleotti, G. Micconi, and S. Caselli, “Object interaction and task programming by demonstration in visuo-haptic augmented reality,” in Multimedia Systems, vol. 22, no. 6, 2016.

[18] X. Fang and J. Zhang, “A collaborative method between CAD and CAR software for robotic cells design,” in Proceedings of 2nd International Conference on Information Technology and Electronic Commerce, IEEE, 2014.

[19] P. Xia, A. M. Lopes, and M. T. Restivo, et al., “A new type haptics-based virtual environment system for assembly training of complex products,” in The International Journal of Advanced Manufacturing Technology, vol. 58, no. 1-4, 2012.

[20] M. R. Bahubalendruni and B. B. Biswal, “A review on assembly sequence generation and its automation,” in Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, vol. 230, no. 5, 2015.

[21] D. Le Tho, M. Andulkar, and W. Zou, et al., “Self adaptive system for flexible robot assembly operation,” in 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2016.

[22] DAQRI, ARToolKit. [Online] Available: http://artoolkit.org/. Accessed on: Mar. 05 2018.

[23] Open Source Robotics Foundation, Gazebo. [Online] Available: http://gazebosim.org/. Accessed on: Mar. 06 2018.