Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 76 (2018) 161–166

www.elsevier.com/locate/procedia

7th CIRP Conference on Assembly Technologies and Systems

Intuitive robot programming through environment perception, augmented reality simulation and automated program verification

Jonas Wassermann[a,*], Axel Vick[b], Jörg Krüger[a,b]

[a]*Technische Universität Berlin, Pascalstrasse 8-9, 10587 Berlin, Germany*
[b]*Fraunhofer Institute for Production Systems and Design Technology IPK, Pascalstrae 8-9, 10587 Berlin, Germany*

[*] Corresponding author. Tel.: +49-30-314-28737. *E-mail address:* wassermann@iat.tu-berlin.de

Abstract

The increasing complexity of products and machines as well as short production cycles with small lot sizes present great challenges to production industry. Both, the programming of industrial robots in online mode using hand-held control devices or in offline mode using text-based programming requires specific knowledge of robotics and manufacturer-dependent robot control systems. In particular for small and medium-sized enterprises the machine control software needs to be easy, intuitive and usable without time-consuming learning steps, even for employees with no in-depth knowledge of information technology. To simplify the programming of application programs for industrial robots, we extended a cloud-based, task-oriented robot control system with environment perception and plausibility check functions. For the environment perception a depth camera and pointcloud processing hardware were installed. We detect objects located in the robot's workspace by pointcloud processing with ROS and the PCL and add them to the augmented reality user interface of the robot control. The combination of process knowledge from task-oriented application programming and information about available workpieces from automated image processing enables a plausibility check and verification of the robot program before execution. After a robot program has been approved by the plausibility check, it is tested in an augmented reality simulation for collisions with the detected objects before deployment to the physical robot hardware. Experiments were carried out to evaluate the effectiveness of the developed extensions and confirmed their functionality.

Keywords: Industrial Robotics; Robot Programming; Augmented Reality

1. Introduction

The rising complexity of products, machinery and the market is rendering great challenges for production companies. This includes high investment costs and security concerns. Especially small and medium enterprises (SME) are faced with additional obstacles like insufficient qualification of their staff.

To cope with high complexity and low qualification, SMEs are seeking simple and intuitive operation of machines and software. Everyone with and without knowledge should be able to work with machine tools and robots while avoiding long training lessons.

One approach for designing the control system for machines more intuitive is the use of Augmented Reality (AR). This technology is adding useful information to the environment to give the operator suitable assistance. In our case, the AR is giving a visual overlay to the video capture of a production cell with an industrial robot, showing e.g. its simulated representation as well as the planned path. This enables a set of possibilities for intuitive robot programming and the analysis of existing programs [1].

The cloud-based robot controller is designed to consist of several independent services that are combined together to form the final application. These services then are executed remote on a high power computing infrastructure like virtual machines in the cloud. The communication between the single services is often solved with standard network transport protocols like TCP/IP or higher level REST-ful HTTP requests. One of the more recent communication options is the OPC UA standard. This machine-to-machine protocol is well accepted in academia and industry within the Industry 4.0 initiatives.

To facilitate the migration to Industry 4.0 for SME, this paper presents the extension of a cloud-based and task oriented robot control with an environment perception service and automatic program verification. The architecture is service-oriented using OPC UA and ROS as communication frameworks.

After analyzing the actual state of the art in section 2, we present our approach and selection of methods in sections 3. The feasibility is demonstrated in experiments in section 4 and the paper closes with a conclusion in section 5.

2. Related Work

Using augmented reality in the field of industrial robots is not new and was first presented in 2002 by Marin et.al. [2]. The authors included Object Recognition and Grasp Determination as modules of their control system over the internet. Later, the demonstration of reducing programming time and increasing program quality is done in [3]. The evaluation also revealed the approach to be subjectively intuitive and easy.

Also big robot manufacturers were investigating the possibilities and challenges of augmented reality for industrial robots [4]. The authors found the training and qualification for robot operation and programming as the most promising field for this technology and forecast its integration in an intelligent teach pendant. Surprisingly, such a device was developed and presented eight years later by Abbas in [5] as implementation for smart phones.

In contrast to the common video-see-through approach, Reinhart et.al. considered a laser projector to enrich the environment with computed information in [6]. The presented spatial user interface also supports direct manipulation of augmented data like planned robot trajectories for surface tasks, e.g. cutting, welding or curing. Lambrecht et. al. adapted the idea to arbitrary task, trajectories and points while defining and manipulating them in AR through hand and finger gestures.

Human-Robot Interaction with Augmented Reality has become famous in the last years and was used in many other systems [7][8]. A more comprehensive review on programming methods including AR for SME is provided by Pan et.al. in [9].

The integration of AR-as-a-service in cloud infrastructures is investigated in [10] and [11]. The robotic tele-operation with augmented reality, like in our control system, is mentioned in [12].

3. Methods and materials

To achieve the desired intuitive programming of robot programs we extended an existing cloud based robot control system. Figure 1 shows the new architecture of the robot control.

It consists of 6 parts:

- the main control program
- 2D image capture
- 3D image capture and processing
- graphical user and programming interface
- robot interface
- Communication Server

The main control program creates the control commands for the robot interface, detects available workpieces and computes the tasks that need more processing power. These tasks are creating the augmented reality simulation and the collision detection.

To capture 3D-data of the robot environment we added a Kinect V2 camera to the robot cell. Together with an Intel NUC it realizes the environment perception. It detects objects in the workspace of the industrial robot and computes oriented bounding boxes around them. These boxes are send to the main control program and added to the augmented reality visualization and collision detection.

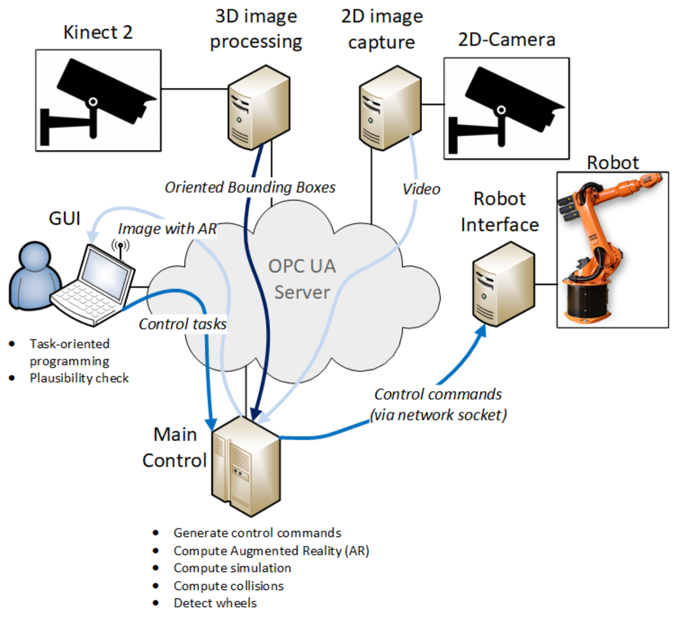


Fig. 1. System Architecture with Cloud Communication Paths

A fixed installed 2D camera in the robot cell or the camera of programing interface device send their Image to the main control program. The augmented reality visualization is rendered on top of this images. To do that a 2D-marker in the robot cell is located.

In the graphical user and programming interface the operator can see the augmented 2D image and further information. He can edit existing task-oriented programs or create a new one. After loading or creating a program the operator can then simulate it. Before simulation, the program is automatically validated and in the simulation checked for collisions.

The robot interface receives commands from the main control program and executes the movement of the Kuka KR6 industrial robot.

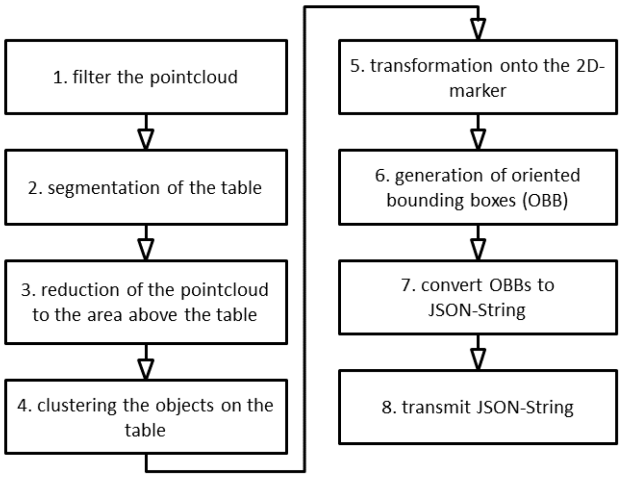
The communication Server manages the communication between the parts of the control system. All parts, except for the robot interface, have a communication client, that is connected to the communication Server. The robot interface is directly connected to the main control program via a network socket.

3.1. Communication

All communication, except between main control program and robot interface is handled by OPC UA. The communication Server is an OPC UA server. All other system parts implement an OPC UA client. They connect to the server and exchange information through it. OPC UA gives a great advantage in flexibility, scalability and migration. Any user client can access the robotic services that are published through the OPC UA Server and implement e.g. own data visualization layers. In this particular approach, the OPC UA Server itself is considered as an integral part of the cloud solution (see Fig. 1).

3.2. Environment perception

To generate data for the environment perception we installed a depth-image camera, a Microsoft Kinect 2, above the worktable of the industrial robot. The Camera is connected to small computer, an Intel NUC. The computer runs on Ubuntu 14.04 and it operates the Camera through the open source driver libfreenect2 and the ROS package iai_kinect2. Using the ROS package ar_track_alvar, a 2D marker is searched for and located in the point cloud of Kinect 2. The marker position in the camera image is provided to other ROS nodes in a topic. Another ROS node processes the point cloud with the Point Cloud Library. To do this, it requires the point cloud information from Kinect 2 and the marker position. Image processing is performed in a series of operations shown in Figure 2.


1. filter the pointcloud
2. segmentation of the table
3. reduction of the pointcloud to the area above the table
4. clustering the objects on the table
5. transformation onto the 2D-marker
6. generation of oriented bounding boxes (OBB)
7. convert OBBs to JSON-String
8. transmit JSON-String
Fig. 2. Image Processing Operations Queue

First, the point cloud is filtered to remove excess information. A Threshold filter limits the point cloud to the inside of the robotic cell. Next, the worktable gets segmented. After the surface of the table has been found, only the part of the point cloud above the table top is examined. In the next step, all Objects that still remain in the point cloud get clustered. The clusters are then transformed from the camera coordinate system onto the detected 2D markers coordinate system. In order to transmit the clusters over network, they must then be transformed into a form that can be transferred with little data traffic. For that oriented bounding boxes (OBB) are computed around the clusters. The boxes are then wrapped into a JSON string.

After the image processing is completed, the data of detected objects is transferred to another ROS node, which takes over the communication with the OPC UA Server.

In the main control program the detected objects are modeled in the augmented reality and checked for collisions with the virtual robot model during simulation. Collisions are represented by overlapping of the OBBs with the simulated 3D robot model. The overlapping is computed with the physics engine bullet physics, that is available as ROS node.

3.3. Plausibility check

Before robot programs are calculated, simulated and transmitted to the robot by the main control program, they are checked for inconsistencies and collisions. The inconsistencies are related to semantic problems like picking up two objects directly after another which is not possible as the gripper would still be busy with the first object when trying to pick the second one. Specific data from the process is required for checking the robot program. These data can be partially read out from the task-oriented robot program or they are defined when setting up the robot or they can be obtained by simple image recognition. The following data is available for plausibility checks:

- Data that can be obtained from the robot program
  - On witch specific positions does the program expect workpieces.
  - Did the robot grab a workpiece before he was supposed to place one down?
- Data that is defined during setup
  - Can any position be approached directly or must a waypoint be approached beforehand to prevent collisions with the environment?
- Data that can be acquired by image recognition
  - On which possible positions is a workpiece really present.

With the help of this information, each robot program is checked before execution and the user is informed of any errors that need to be corrected in the robot program.

This section shows how the positions are checked to see if a workpiece is present. This check takes place directly in the cloud-based control system and is based on a brightness evaluation in the 2D camera image. When the control system is set up for the first time and after changing the position of the static camera, the areas to be examined for their brightness are stored in the program. As soon as the user sends instructions to check positions for workpieces, the mean color value of the camera image is calculated in each area. After the gray value has been determined for each range, the system checks whether it is above a defined threshold. This method only checks if workpieces are present at predefined positions, it does not determine the orientation of the workpieces.

4. Experiments and results

To evaluate the results, several experiments have been carried out to check the individual components of the control system.

4.1. Environment perception

In order to verify the environment perception, we have placed objects at defined positions in the working area of the robot and started the 3d image processing. Then we compared the result with reality. For this purpose, the size and position of the objects is manually measured and compared with the size and position of the calculated oriented bounding box. In addition, the created boxes are placed over the recorded point cloud to check their orientation as a visual evaluation. The selected objects were placed in two groups in the robot's workspace. The surface area of the workspace is 110 x 82.5 cm. Group 1 consists of a Rubics cube, a keyboard and a tea box. Group 2 consists of two parcel packages.

The data was recorded for three runs of environment perception per group. Figures 3 and 4 show the results of one of the three runs per group.

The blue boxes represent the calculated position and orientation of the objects. The tire warehouse and the car are parts

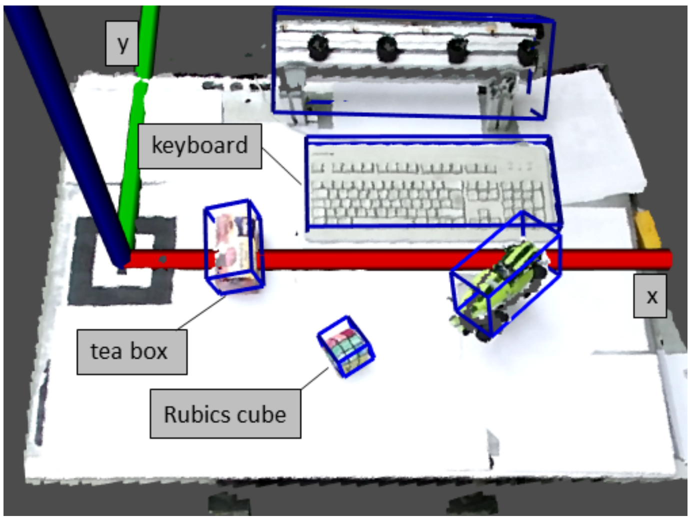


Fig. 3. Environment Perception Results of Group 1

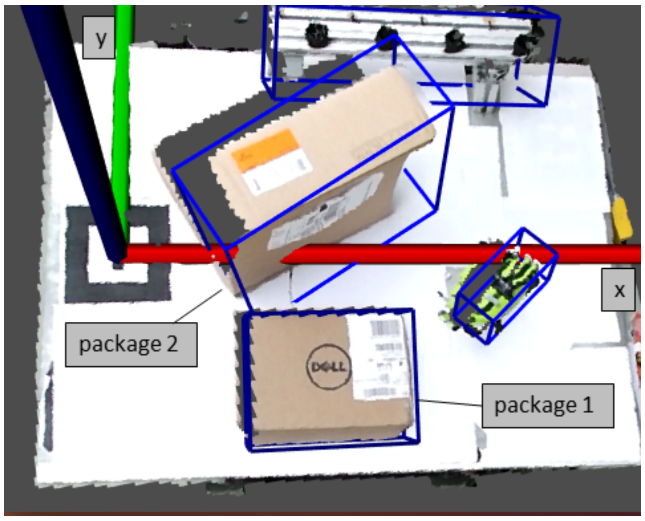


Fig. 4. Environment Perception Results of Group 2

of the working process. The figures show that the position and orientation of the computed bounding boxes matches the real objects.

If the real volumes of the objects and the mean volumes of the determined bounding boxes are compared, like presented in table 1, one can see that most objects are calculated to take up more space.

Table 1. Results Environment perception

Object	mean volume deviation (%)	mean position deviation (*cm*)
Rubics Cube	+17.4	y: 0.6 x: 1.7
keyboard	-6.3	y: 0.7 x: 2.4
tea box	+23.2	y: 0.6 x: 0.5
package 1	+10.3	y: 1.5 x: 0.5
package 2	+39.2	y: 1.8 x: 2

Since the bounding boxes do not perfectly remodel the object, the results meet expectations. The fact that the shell around the keyboard is even 6.3% smaller than the real keyboard can be explained by the fact that the keyboard is rectangular, so it can be easily reproduced with a bounding box and in image processing a part of each body is cut off together with the tabletop. In the case of long flat objects, such as the keyboard, this noticeably reduces the volume of the bounding box. Figure 5 shows the volume deviations of the three experiments.

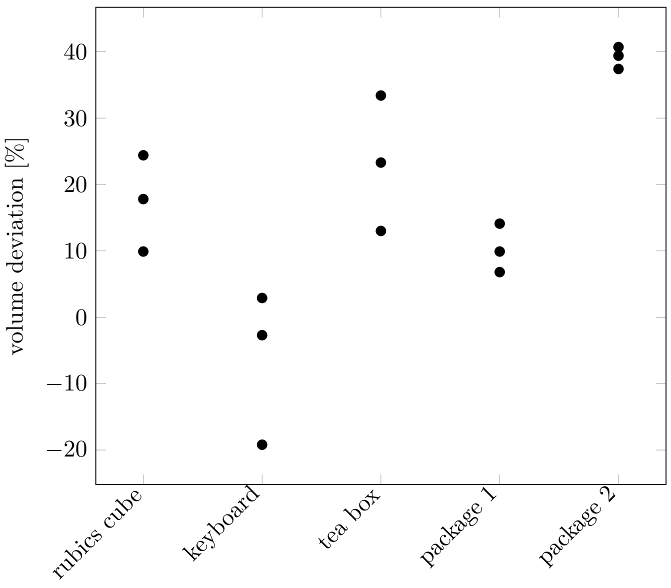


Fig. 5. Volume Deviation of Detected Objects

Also in table 1 the real positions are compared with the mean of the computed positions of the bounding boxes. The range of position deviations lies between 0.5 cm and 2.4 cm. These slight deviations are to be expected, as the bounding boxes do not perfectly replicate the objects and thus the center of gravity shifts. The position deviations of all three runs are shown in figure 6.

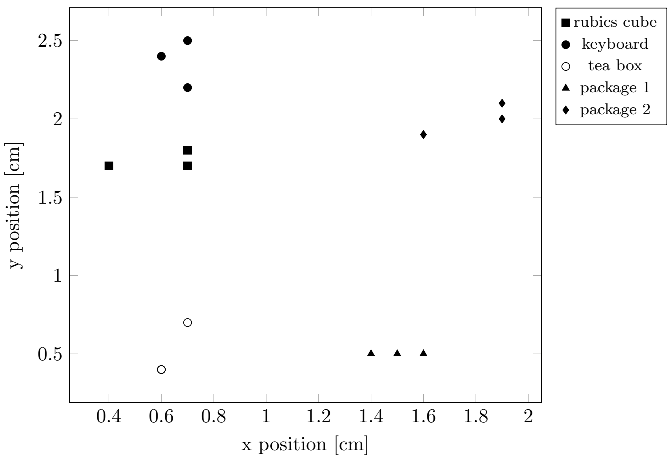


Fig. 6. Position Deviation of Detected Objects

4.2. *Collision detection*

In order to check the collision detection, an experiment is carried out in which the entire working area of the robot is filled with artificially created objects in the augemnted reality. Then the robot model is moved and the coloration of the objects in the augented reality is observed.

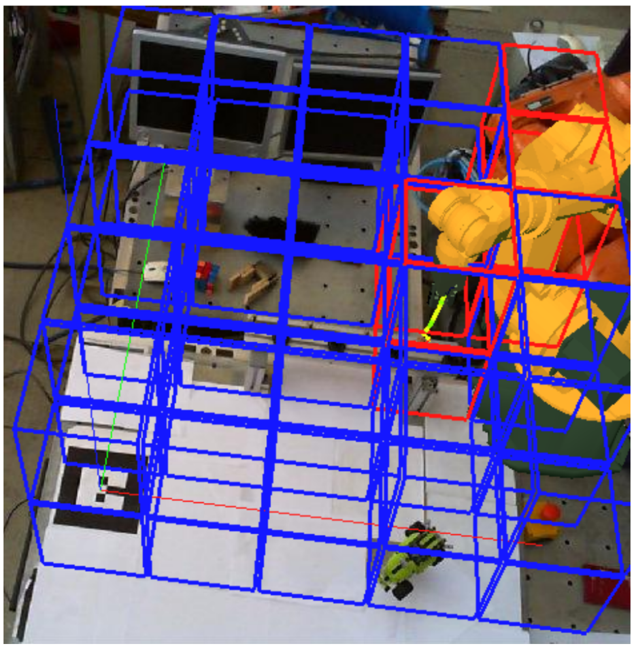


Fig. 7. Collision Detection Results

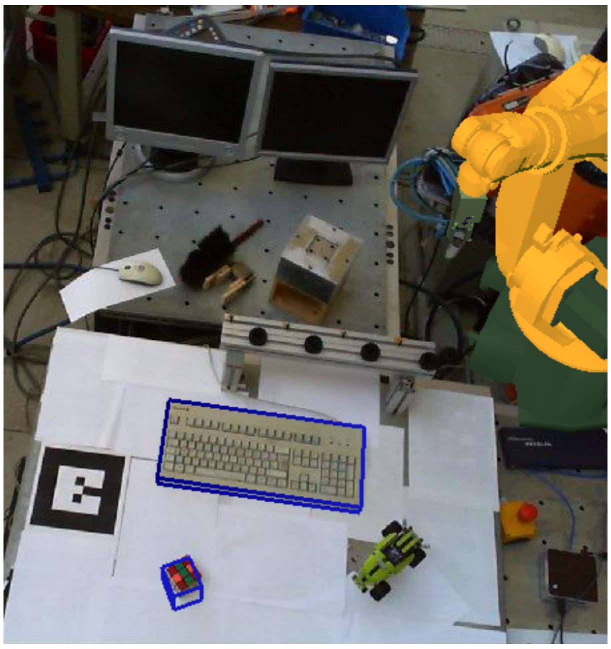


Fig. 8. 2D camera image with augmented reality

On figure 7 one can see that the collision detection has worked. The boxes with which the robot collides are detected and colored red.

4.3. Plausibility check

In order for the verification of the robot program to work, the program needs information about the current position of the workpieces.

In order to check the functioning of the workpiece detection, experiments are carried out in which working material is available at different positions in the working area. The result of the automatic workpiece detection is then compared with reality. Five experiments were carried out with different distribution of the working materials.

The system detected all workpieces in all five experiments.

Since the system only checks fixed positions in the workspace, there are problems when the position of the camera changes slightly due to vibrations or impacts. The positions that are checked for workpieces must then be readjusted.

In order to verify the verification of the robot programs with a functioning workpiecedetection, programs are deliberately written with and without errors, their execution is started and the response of the cloud-based control system is evaluated. The errors are:

- The program stats to pick up a workpiece on a position where no workpiece is present.
- The program stats to place a workpiece on a position where a workpiece is already present.
- The program stats to pick up a workpiece while there is still a workpiece in the gripper.
- The program stats to place a workpiece when no workpiece was graped before.
- The program stats to move to a position directly, when it is necessary to move to a safety point before moving to that position.

Whenever there was an error in the program, the program was stopped before execution. The error-free programs have all been executed.

5. Conclusion

The existing cloud-based control system was extended by an environment perception and collision control, as well as the automatic plausibility check and verification of robot programs. These enhancements offer employees of small and medium-sized companies the possibility to program industrial robots intuitively and without complex learning steps. The experiments carried out prove the function of the extensions created. By installing a depth camera and an Intel NUC in the working cell of the robot, objects located in the working space of the robot are transferred to the augmented reality of the cloud-based controller and can be tested there for collisions with the simulated robot. The workpiece detection automatically detects the material the robot is working with. With the conversion of the communication between the modules of the cloud-based control system to OPC UA, the technology recommended in the reference architecture model Industry 4.0 as communication technology is used. As soon as the program is started, it is automatically checked for errors and can then be tested for collisions with the robot environment in the simulation.If the operator wants to view the simulation from a different perspective, he can use the camera of his operating device to draw the augmented reality into the image he has taken. Figure 8 shows the augmented reality. The blue bordered objects were detected by the environment detection and are examined during the simulation for collisions with the robot. The toy car and the tire storage are part of the robot's working process and are not checked for collisions.

There are several approaches to the further development of our cloud-based control.

In the current version, the cloudbased control uses a single

2D marker-based method for drawing the augmented reality. The augmented reality cannot be displayed if the required 2D marker is hidden or outside the camera's field of view. Replacing this procedure with a procedure that works with multiple 2D markers or without markers at all could solve this problem.

Our workpiece detection is able to detect black workpieces in the warehouse and on the car. In order for the workpiece detection system to be able to recognize material with different colors or at positions other than the previously defined ones, the gray scale detection could be replaced by an object recognition method that can recognize the tires at any position in the camera image. The robot could then pick up tires from any position in the working area.

The collision detection detects collisions, visualizes them and stops programs with collisions before excution. A further improvement would be to compute new robot paths to avoid detected objects.

References

[1] Jan Guhl, Son Tung Nguyen, and Jörg Krüger. Concept and architecture for programming industrial robots using augmented reality with mobile devices like microsoft hololens. In *Emerging Technologies And Factory Automation*, USA, 2017. IEEE.

[2] R. Marin, P. J. Sanz, and J. S. Sanchez. A very high level interface to teleoperate a robot via web including augmented reality. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 3, pages 2725–2730, 2002.

[3] T. Pettersen, J. Pretlove, C. Skourup, T. Engedal, and T. Lokstad. Augmented reality for programming industrial robots. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 319–320, Oct 2003.

[4] R. Bischoff and A. Kazi. Perspectives on augmented reality based human-robot interaction with industrial robots. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 4, pages 3226–3231 vol.4, Sept 2004.

[5] S. M. Abbas, S. Hassan, and J. Yun. Augmented reality based teaching pendant for industrial robot. In *2012 12th International Conference on Control, Automation and Systems*, pages 2210–2213, Oct 2012.

[6] G. Reinhart, W. Vogl, and I. Kresse. A projection-based user interface for industrial robots. In *2007 IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, pages 67–71, June 2007.

[7] H. Fang, S. K. Ong, and A. Y. C. Nee. Robot programming using augmented reality. In *2009 International Conference on CyberWorlds*, pages 13–20, Sept 2009.

[8] C. L. Ng, T. C. Ng, T. A. N. Nguyen, G. Yang, and W. Chen. Intuitive robot tool path teaching using laser and camera in augmented reality environment. In *2010 11th International Conference on Control Automation Robotics Vision*, pages 114–119, Dec 2010.

[9] Z. Pan, J. Polden, N. Larkin, S. V. Duin, and J. Norrish. Recent progress on programming methods for industrial robots. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8, June 2010.

[10] P. H. Chiu, P. H. Tseng, and K. T. Feng. Cloud computing based mobile augmented reality interactive system. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 3320–3325, April 2014.

[11] I. Mal, D. Sedlek, and P. Leit£o. Augmented reality experiments with industrial robot in industry 4.0 environment. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 176–181, July 2016.

[12] Y. Lin, S. Song, and M. Q. H. Meng. The implementation of augmented reality in a robotic teleoperation system. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 134–139, June 2016.