Robotics and Computer–Integrated Manufacturing 71 (2021) 102178

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcm

Review

Towards commissioning, resilience and added value of Augmented Reality in robotics: Overcoming technical obstacles to industrial applicability

Jens Lambrecht [a,*], Linh Kästner [a], Jan Guhl [b], Jörg Krüger [b]

[a] Chair Industry Grade Networks and Clouds, Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
[b] Chair Industrial Automation Technology, Technische Universität Berlin, Pascalstr. 8-9, 10587 Berlin, Germany

ARTICLE INFO

Keywords:
Augmented Reality
Robotics
Industrial integration
Distributed automation architectures
Spatial referencing
Robot programming

ABSTRACT

Augmented Reality (AR) has the potential for facilitating the interaction with robots by enhancing the operator's spatial understanding as well as providing further cognitive support, e.g. in order to make manual programming processes more efficient and provide on-site simulation. However, we consider major issues that prevent a widespread use of AR towards robotics in industries. Initially, the commissioning of AR devices for robotic applications demands spatial referencing of robots and AR devices. Fiducial markers are a popular artificial aid, but hard to implement in industrial use cases because of additional efforts and a lack of robustness. A further bottleneck for developing AR applications in robotics is the restricted technical and ergonomic maturity of AR devices, e.g. limited local computation and short product life-cycles. Furthermore, benefit through AR in comparison to classic online and offline programming techniques is still unclear for most applicators. In this paper, we present approaches towards a distributed, hardware-agnostic microservice architecture with standard interfaces for an interoperable usage of AR in robotics. Furthermore, we present marker-less pose estimation methods for articulated robot arms as well as mobile robots based on RGB and depth images that serve automatic referencing. Finally, we reveal further application potential of AR in robotics in terms of combining advantages from online and offline programming.

1. Introduction

In the general context of information sharing of Industry 4.0, Augmented Reality (AR) bares the potential to support processes of human robot interaction in terms of augmenting real industrial environments through the visualization of spatial-related virtual information [1]. In high-wage countries, the production and manufacturing industry faces a lack of skilled labour and a general need for flexibility of production systems. Because of human's exceptional flexibility, adaptability and cognitive skills, human labour is and will be still essential within production environments [2]. AR bridges the gap between real and virtual world with the potential of supporting manual work and leveraging new technology domains like the Digital Twin [3,4] and natural interaction, e.g. haptic and non-haptic gestures [5]. Especially the trend for human robot collaboration (HRC) provokes the usage of new interaction technologies in robotics as it turned out that most of the current HRC reference applications lag behind efficiency compared to pure manual task execution [6]. In fact, there is an increased demand for human robot interaction in HRC due to a strong need for communication in terms of coordination and synchronization between robot and human worker. This applies to stationary robots [7] as well as mobile robots and transport systems [8]. Hence, AR can fill this gap providing novel mobile interfaces through the use of wearables and hand-held device (s. Fig. 1). Consequently, AR is more flexible and covers a novel range of interaction opportunities compared to conventional display-based output technologies. From our perspective, AR is a key technology for hybrid work systems of the future as well as for Digital Twins.

Despite the technology potential and early adoption in general industries [9], the usage of AR within robotic applications on the industrial shop floor lags behind, as there are a lot of promising approaches in the research domain since years, but there are no respective products on the market or practical industrial use cases beyond proof of concepts yet. In parallel, there are new opportunities towards enhancing the function range and overall performance through reliably offloading computation following principles of service-oriented distributed software architecture involving cloud and edge computing as well as 4G or 5G wireless radio communication [10]. Within this publication, we aspire to address major lacks of applicability of AR in robotics,

* Corresponding author.
E-mail addresses: lambrecht@tu-berlin.de (J. Lambrecht), doan.hl.kaestner@campus.tu-berlin.de (L. Kästner), guhl@iat.tu-berlin.de (J. Guhl), joerg.krueger@tu-berlin.de (J. Krüger).

https://doi.org/10.1016/j.rcim.2021.102178
Received 12 March 2020; Received in revised form 10 April 2021; Accepted 12 April 2021
Available online 26 April 2021

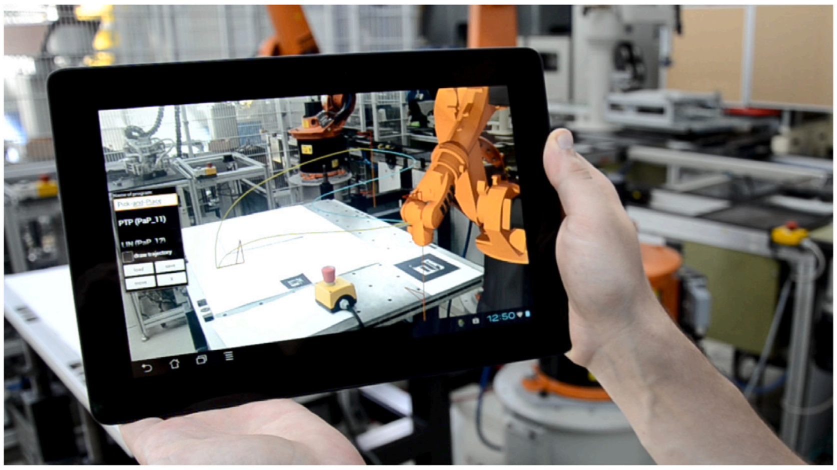

Fig. 1. Mobile marker-based Augmented Reality kinematics overlay simulation on a hand-held device.

but limit ourselves to software-related issues, e.g. not dealing with hardware-related ergonomic issues, e.g. the weight of AR devices. In addition to stationary robotic manipulators, we are comprehensively considering mobile robots and autonomous transport systems due to many similarities in control, interfaces and interaction demands.

From our knowledge and broad discussion with potential applicators in the manufacturing industry, we are facing the following main obstacles that prevent the practical usage of AR in robotics:

- resilience of software due to short product life-cycles and restricted AR hardware performance,
- complicated and time-consuming commissioning, i.e. referencing, and
- unclear added value in comparison to classic online and offline programming methods.

This paper is organized as follows. Section 2 breaks down the boundaries for applicability, discusses the state of the art and clarifies the technological maturity. Section 3 introduces standardized software architectures, communication and interfaces and Section 4 describes methods for automatic commissioning. In Section 5, we introduce approaches for added value combining methods from online and offline programming through AR. Finally, we summarize and present future directions in Section 6.

2. Fundamentals and related work

2.1. Resilience

As most AR devices, especially head-mounted displays, still lack in computation power, usability and ergonomics, e.g. because of limited coverage of the user's field of view as well as heavy weight [11], the AR hardware is subject to short innovation cycles. In order to provide AR solutions for the robotics domain that can still be used for the next hardware generations, a monolithic software approach that is tailored to a specific hardware or development environment should be avoided. Where this is not the case, one has to face a non-transferability or need for major adaptions towards new hardware.

The Unity software is evolving more and more towards a standard for AR development, but obliges the developers to the C# programming language and non-standardized interface, e.g. a connection of service of the Robot Operating System (ROS) is not easily possible [12,13]. Kousia et al. [8] introduce an AR software suite for interaction with mobile robots in collaborative work environments. The overall software is bridging the gap between a HoloLens AR application and the ROS-based navigation stack, but is not taking into account to build up the application level device agnostic and interoperable. Consequently, interoperability issues also apply to the integration of AR devices into companies' IT structures and connecting wide-spread machine interfaces, e.g. OPC UA. Peppoloni et al. [14] propose a framework for ROS-integrated immersive robot teleoperation, but are not considering distributed computation approaches.

Consequently, our architecture concept introduced in this publication is AR device agnostic through offloading modular software functions towards external cloud and edge infrastructure following a modern microservice approach [15] and unified interfaces.

2.2. Commissioning

Addressing the commissioning of AR assistance in robotics comprises the spatial alignment of coordinate systems of AR device, robot and environment. Thus, the virtual objects displayed at desired positions can augment the users view. This applies to video-see-through devices, e.g. tablet-PCs as well as to optical-see-through (OST), e.g. head-mounted displays with transparent screen. Within this publication, we neglecting the additional 3D transformation between sensor and display of the AR device as well as the parallax effect of OST devices [16]. Eq. (1) shows how to transform an object position vector $\vec{p}_r$ from the robot coordinate system towards the AR coordinate system $\vec{p}_a$ in terms of visualization with the help of the homogeneous coordinate transformation between AR device and robot T^r_a.

$$\vec{p}_a = T^r_a \cdot \vec{p}_r \tag{1}$$

As most AR devices are typically mobile, T^r_a needs to be determined for every frame or respectively every visualization cycle in order to adapt the visualization according to the movement of the AR device. If the AR device has an integrated Simultaneous Localization and Mapping (SLAM) function, it is basically sufficient to determine the robots pose within the global map T^r_m once and subsequently hand over to the SLAM following Eq. (2) continuously calculating the pose of the map within the AR coordinated system represented by T^m_a. In regards to SLAM, it makes sense to re-locate the robot from time to time in terms of drift compensation or loop closing.

$$T^r_a = T^m_a \cdot T^r_m \tag{2}$$

As a direct six degrees of freedom (DoF) localization of the robot is quite challenging as robotic manipulators are articulated objects and partially symmetrical, nearly all early applications of AR in robotics using RGB images consider the usage of artificial aids, i.e. fiducial markers, for pose estimation of the robot. Examples for using markers can be found in [17–20]. Nowadays, fiducial makers are still a popular aid for prototyping AR applications in robotics [21,22]. Basically, the accuracies of the marker detection approach depend on various parameters, e.g. image resolution, distance and angle to marker as well as the size of the marker [23]. The drawback of using markers is the lacking applicability towards real industrial scenarios, because of the efforts for the application of the markers in the environment, inflexibility and lacks of recognition towards staining or occlusion of the markers. Furthermore, the marker itself must be aligned once to the robot coordinate system, which is practically mostly done precisely through a complex manual determination of point correspondences between marker and robot.

The application of artificial 3D reference objects is also well widespread for AR devices additionally using providing point clouds from depth cameras [24]. However, [25] is proposing the usage of dynamically 3D generated models and the usage of the Iterative Closest Point (ICP) algorithm in order to match the models into point clouds generated from depth images. In practice, this method is quite computationally intensive because of the iterative approach of ICP and not applicable in cluttered scenes and partly occluded robots. We need to conclude, that the usage of markers is not appropriate to leverage industrial applications of AR in robotics. As a first step towards automatizing the referencing, Puljiz and Hein [26] are proposing a semi-automatic referencing method using user input for a rough initial guess and standard ICP registration algorithm for refinement.

Within this publication, we are introducing novel methods for (real-time) robot pose estimation based on Convolutional Neural Networks (CNNs) that does not rely on fiducial markers, scene knowledge, specific cameras or configurations, initialization or 3D volume and shape models. In comparison to classic computer vision approaches, the presented methods also cope with partial occlusions.

2.3. Added value

In terms of industrial acceptance of novel technologies, striking benefits regarding time, quality and cost must be obvious. In contrast to pure virtual offline programming (OLP) systems [27], AR-based simulation and programming systems bare the general potential of transferring automatic programme and motion generation, validation and optimization to the shop floor [28]. For this purpose, AR is typically combined with new user input technologies such as bare-hand gesture [29,30]. In [31], we demonstrated that the combination of AR and gesture-based programming has major advancements in comparison to classic teach-in and OLP in terms of efficiency, effectiveness as well as subjective assessment.

Unlike OLP, using modern depth-sensor-equipped AR devices, there is no need for virtually modelling the robot cell beforehand [32]. Moreover, components and collision objects can be detected and localized online by using the AR device's sensor.

Within this publication, we focus on benefits from merging functions from online and offline programming within AR applications for programming of robotics manipulators by enriching online programming processes with results from computational heavy tasks like mapping, path planning and object recognition.

3. Communication and computing architecture

This section describes a cloud- and edge-enabled software and communication architecture for mobile AR simulation in robotics. Hardware, such as the HoloLens and tablet-PCs as well as the edge server are interconnected following a distributed computation principle (s. Fig. 2). As a basic software architecture, we are following the microservice paradigm [15] providing modularized software functions that:

- run independently,
- are covering small functions with a specific focus and
- are supporting decentralization through easy and automatic orchestration, monitoring and transfer between different infrastructures.

Our microservices are encapsulated in Docker containers that are integrated in a DevOps pipeline orchestrated through the Kubernetes framework.

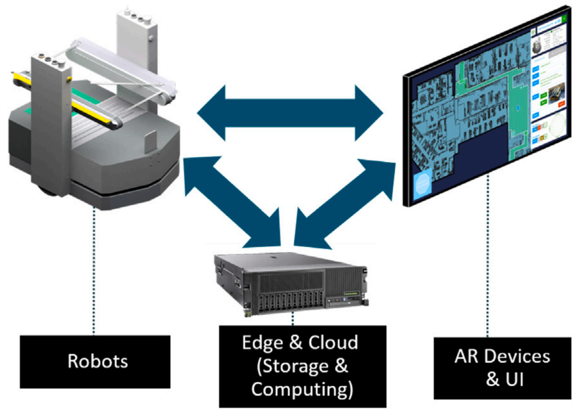

Fig. 2. Connected devices within our distributed control architecture: autonomous robots, edge computing devices for storage and computing as well as AR devices.

3.1. Communication middleware

We use the ROS framework as basic communication middleware with its XML-RPC-based service protocol supporting distributed synchronous (ROS services) and asynchronous communication (ROS topics) as well as data storage. The ROS framework was enhanced by security functions in order to secure communication on the network and transport layer. The usage of ROS simplifies the declaration and implementation of transmitted messages between units and manages them using a ROS master node. The transmission of sensor data to another device is done through reading raw data in binary from the sensor and converting and transferring it to a ROS topic. In order to provide further connectivity to non-ROS modules, the *rosbridge* package is used to address WebSocket interfaces. In addition, an interface towards the industrial automation protocol OPC UA is considered.

3.2. Service distribution

We consider a flexible distribution of specific AR interaction and robot control services towards cloud and edge computing infrastructure depending on specific service requirements regarding latency, jitter and bandwidth. In [10], we introduced an approach towards offloading real-time applicable control services for mobile robots with the help of edge computing and 4G campus networks. Within this scenario, an AR simulation could benefit from using the same software service for (motion) visualization as used for control of the robot reducing the simulation-to-reality gap. Furthermore, additional interaction services, e.g. gesture-based interaction, could be connected seamlessly. Besides flexibilization and hardware-independence, offloading software functions towards virtualized infrastructures enables scaling the solutions as well as an improved software life-cycle management.

Regarding AR-based simulation, the only software modules left onboard (on-premises) of the AR device are device-specific localization and visualization functions and our basic ROS nodes for publishing sensor data, e.g. RGB and depth images, and subscribing robot-specific visualization objects. Hereby, we shortly introduce our main microservices enabling AR simulation:

- Kinematics: providing calculation for forward and inverse kinematics as well as management of coordinate systems.
- Path Planner: local and global trajectory planning and roll-out in 2D and 3D, also applicable for the motion control real robots.
- Collision Avoidance: collision detection and re-planning in 2D and 3D.

Further services are managing maps, robot programmes and connecting user input.

3.3. Focus: HoloLens — ROS bridge

In case of the HoloLens, the AR device is programmed with C# as.NET application in Unity. Thus, there is no option to deploy ROS nodes directly on-premises. To bridge this gap, we considered a communication based on the ROS# framework [33]. ROS# is forwarding asynchronous communication between Unity applications and ROS through WebSockets. The message exchange between the two entities is done through publisher and subscriber nodes encoding and decoding the data into the common *rosbridge* protocol. Unfortunately, WebSocket communication is not supported by the HoloLens. Hence, we extended the C# framework by adding windows networking sockets for asynchronous communication to enable a HoloLens integration into ROS. A similar approach using the ZeroMQ protocol instead of WebSockets can be found in [13].

4. Commissioning

In this section, we present our recent approaches towards markerless spatial referencing of robot and AR coordinate systems. We focus on determining the six DoF pose deviation with the help of CNN-based approaches to ensure reliable operation even in cluttered scenes and under partial occlusions. The CNNs-based approaches demand for an individual dataset and training process for each robot type or model. Recent publications show the feasibility of successfully using solely synthetic data [34] for training as well as discussing limitations [35]. Hence available for nearly every AR device, we primarily consider approaches based on RGB image data, but also present a depth-image-based approach for rigid mobile robots. As a consequence, spatial calibration is made automatic and takes less than a second. Transformations are calculated and implemented directly into the respective AR application as software service. In terms of using fiducial markers, this process could last up to two hours comprising appropriate marker attachment in the environment, measuring markers in the robot coordinate system as well as calculation and implementation of the transformations. Our latest evaluations [35] show that achievable robustness and accuracy of the proposed methods are comparable with those of fiducial markers.

4.1. RGB images

The proposed RGB-based approach consists of a 2D-3D perspective transformation of inherent keypoints of the robot. Consequently, we rely on a camera with calibrated intrinsic parameters. Unlike [36], we solve the perspective transformation with classic perspective-n-point (PnP) algorithms in order to provide a solution that is applicable for different cameras and parameter settings. CNNs are utilized for the detection of 2D keypoints in the image, whereas 3D point correspondences are known from forward kinematics and joint encoder readings from the respective robot's API.

4.1.1. 2D–3D perspective transformation

Extending the transformation matrix between camera and robot T_a^r (s. Section 2.2) by incorporating the 2D keypoints in the image plane $\vec{c}_i$, 3D keypoint correspondences $\vec{d}_i$ and the intrinsic camera matrix P leads to Eqs. (3) and (4) involving a scaling factor h.

$$h \cdot \vec{c}_i = P \cdot T_a^r \cdot \vec{d}_i \tag{3}$$

$$h \cdot \vec{c}_i = \begin{bmatrix} f_X & 0 & c_X \\ 0 & f_Y & c_Y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{bmatrix} \cdot \vec{d}_i \tag{4}$$

The class of PnP algorithms, e.g. EPnP [37], solves this problem statement for n given point correspondences providing T_a^r which includes a rotatory matrix R and a translatory vector $\vec{t}$.

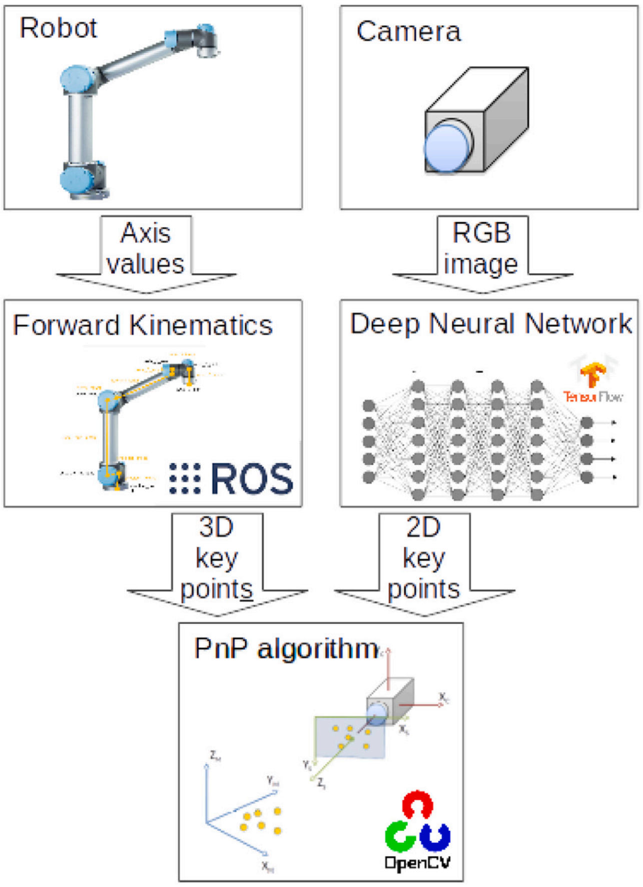

Fig. 3. Processing pipeline for the robot pose estimation.

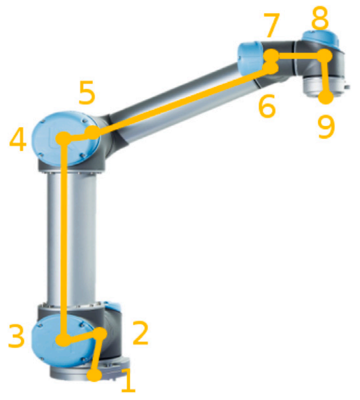

Fig. 4. Keypoint configuration for the UR5 robot.

4.1.2. Articulated robot arms

The overall processing pipeline form articulated robot arms is depicted in Fig. 3. In the following, particular steps of the approach are illustrated taking the example of a Universal Robots (UR) six-axes UR5 manipulator.

Keypoint configuration. For the purpose of keypoint detection, we define a keypoint configuration that is applicable for point correspondences for both, the 2D camera images as well as corresponding 3D positions in respect to the base of the robot. As we aspire to define a generic method, it is useful to consider points directly located along the kinematic chain, i.e. axes and connecting lines. Thus, the 3D points are directly derived from forward kinematics and the 2D points

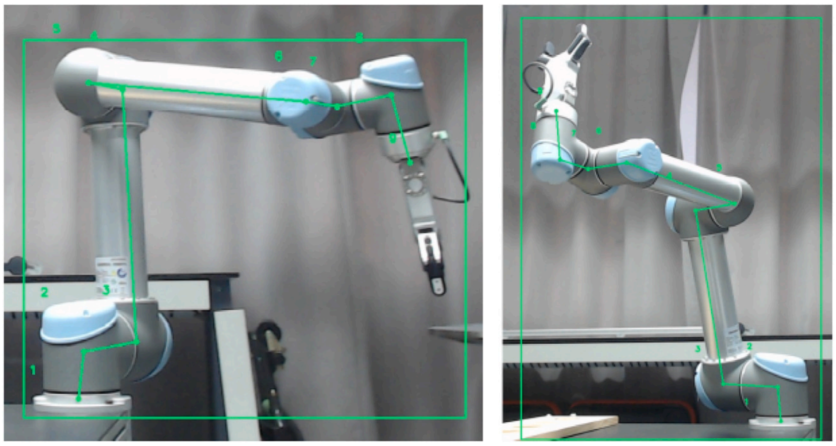

Fig. 5. Exemplary results of the 2D keypoint detection establishing a skeleton model of the robot.

are derived from adapting recent human pose estimation approaches towards articulated manipulators. The keypoint configuration for the UR5 robot includes $k = 9$ keypoints (s. Fig. 4).

2D keypoint detection through CNNs. In order to define the 2D pose of the robot in the image plane, we introduce the following notation encoding the location of all k keypoints along the kinematic chain in an array $(..., (\vec{c}_i)^T, ...)^T, i \in 1, 2, ..., k$, where $\vec{c}$ contains the x and y image coordinates of the ith keypoint in the image plane. The 2D keypoint detection (s. Fig. 5) is considered as a regression task, e.g. taking an Inception-ResNet-v2 backbone for 2D object and keypoint detection in RGB images, s. [38] for further information on our capitalized network topology and evaluation results.

Similar results comprising different CNNs have recently been introduced in [34,39] proving that approach is transferable to other types of robotic manipulators. Providing the keypoint detection deployed on a server or edge infrastructure equipped with an adequate GPU, the service is real-time applicable, e.g. with up to 30 fps using a MobileNet backbone. We published a reference dataset that is free to use in academic context[1] involving a huge amount of labelled real and synthetic images of the UR manipulator [38]. In [40], we prove that the incorporation of synthetic data generated from precise simulation, leads to an overall better precision of the pose estimation because of error-prone manual labels.

3D Keypoint determination through forward kinematics. In order to derive the 3D keypoints in reference to the robot base, we rely on the real-time joint encoder readings $\theta_j, j \in 1, 2, ..., 6$ and utilize point-by-point forward kinematics incorporating joint dimensions. The 3D position array is defined as $(..., (\vec{d}_i)^T, ...)^T, i \in 1, 2, ..., k$, where d_i is the result of the forward kinematics involving θ_j and contains the X, Y and Z coordinates of the keypoints in the robot coordinate system. We illustrate this procedure through the example of the UR5 robot. The UR-controller is connected via the manufacturer-provided Ethernet-based socket interface wrapped into a ROS node. Thus, we obtain the current joint angles of the robot. With the help of respective ROS-industrial packages for the UR5 containing forward kinematics and the Unified Robot Description Format (URDF), the 3D coordinates are computed for the keypoints in reference to the robot base.

4.1.3. Mobile robots

As mobile robots are considered as rigid objects, we apply the pre-described method with $k = 8$ static keypoints representing a 3D bounding box. Therefore, we investigate two recently released neural networks: BetaPose [41] and SSPE [42]. Both are deployed as software

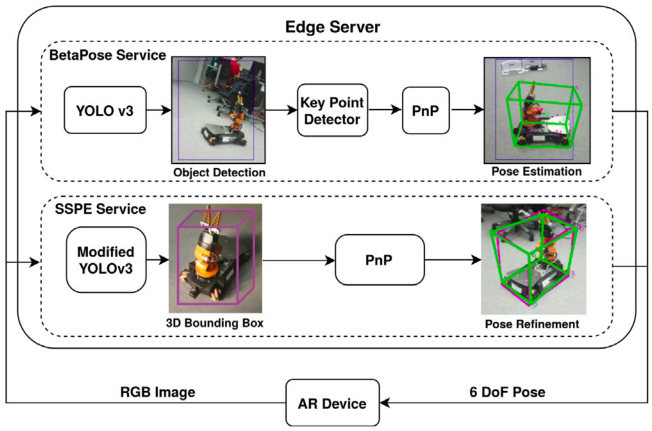

Fig. 6. Workflow for both edge-based mobile robot pose estimation services: BetaPose and SSPE.

services following the proposed architecture (s. Section 3). Basically, both CNNs rely on the Yolov3 object detection [43]. On the one hand, BetaPose initially utilizes the standard object detector in order to localize the robot through a 2D bounding box in the image. Subsequently, a keypoint detector applied to the extracted region yields the eight keypoints. On the other hand, SSPE is a modified Yolo with a direct output of the keypoints. Both approaches cover a PnP algorithm to estimate the 6 DoF pose, whereas the SSPE service is using an additional pose refinement step. The overall workflow of both services is depicted in Fig. 6. Our self-developed annotation tool is available online.[2] The resulting pose estimation for a KUKA Youbot comparing both methods is shown in Fig. 7. Both approaches show robust results with SSPE being robust even on far distances.

4.2. Depth images

Using solely depth images and respective point clouds for pose estimation of the mobile robots has the advantage of avoiding the error prone PnP algorithms. Nevertheless, depth images tend to be more ambiguous for detections in cluttered scenes. On this account, we propose a CNN-based approach. We directly utilize point clouds to train an end-to-end CNN based on the recently released VoteNet [44]. The general workflow is illustrated in Fig. 8. Following the distributed service approach, also the computationally demanding point cloud extraction operation is offloaded from the AR device.

[1] https://tubcloud.tu-berlin.de/s/pFiz229pD25JKBm.

[2] http://annotate.photo.

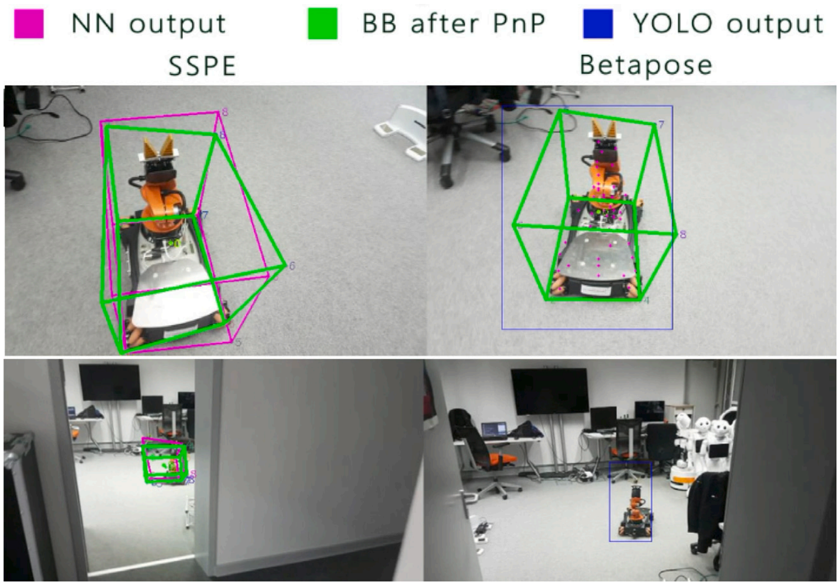

Fig. 7. Results of both CNN approaches for near and far distances: CNN keypoint output of SSPE, resulting pose of both approaches (green) and object detection of BetaPose (blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

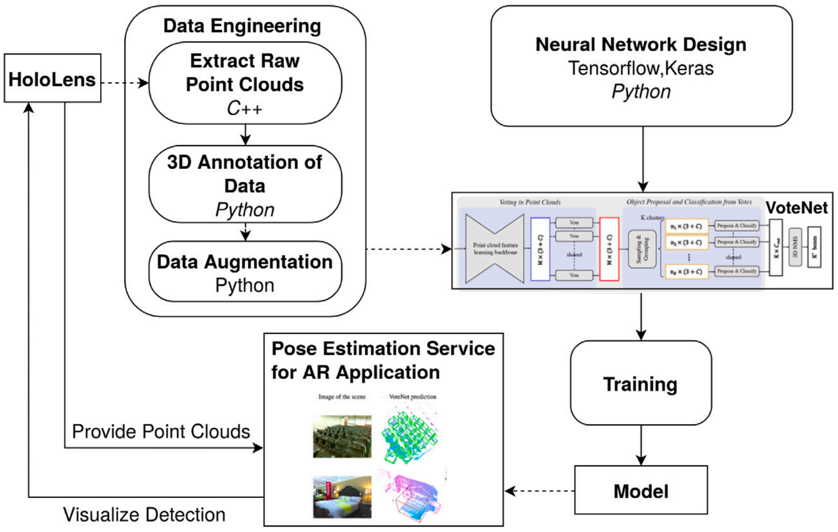

Fig. 8. Workflow of the depth-image-based pose estimation for the HoloLens including data engineering, neural network design and training.

4.2.1. *Point cloud data extraction and processing*

The conversion from depth data to a point cloud is derived using the camera intrinsic parameters. The depth streamer returns a depth buffer D, which contains the distances for each pixel from the 2D image plane. Each pixel point of the 2D image is mapped to the camera unit plane and the results are normalized. To calculate a 3D point out of the 2D coordinates [u,v], additionally the following equations are used.

$$[X, Y, Z] = Z \cdot [u, v, 1] = [Z \cdot u, Z \cdot v, Z] \tag{5}$$

Where Z is the normalization of the distance array D which is computed using Eq. (6).

$$Z = \frac{D}{\sqrt{(u^2 + v^2 + 1)}} \tag{6}$$

$$X = \frac{D \cdot u}{\sqrt{(u^2 + v^2 + 1)}} \tag{7}$$

$$Y = \frac{D \cdot v}{\sqrt{(u^2 + v^2 + 1)}} \tag{8}$$

4.2.2. *3D data annotation*

The annotation of the acquired point cloud data is done with a self-developed 3D labelling tool which takes raw point clouds as input to be visualized with the python library *PPKT*. In order to annotate the robot in the visualizer, the user has to select three points that represent threes corners of the robot base and define a robot-related coordinate system. The fourth corner is computed automatically in order to provide a proper visualization feedback of the base plane.

4.2.3. *Neural network design*

The basic neural network topology is based on VoteNet. We modified the loss functions for our use case as well as training settings to achieve better performance. For detailed explanations, refer to our previous work [45]. Exemplary results are depicted in Fig. 9.

5. Combining offline and online programming

One of the greatest advantages of OLP is the possibility of using additional data, e.g. CAD models of the robot cell, during programming. These additional information enables the optimization of the

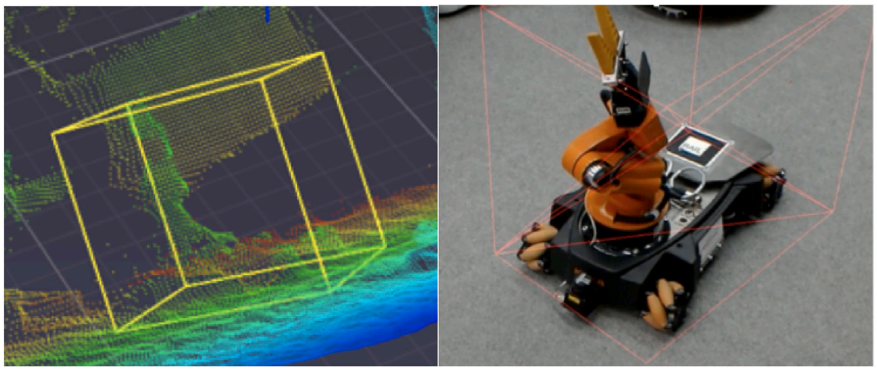

Fig. 9. Detection results for the depth image approach: extracted point cloud visualized within our labelling tool (left) and 3D bounding box visualized in RGB image (right).

robot cell in terms of reachability or even the usage of computational heavy algorithms like path planning for collision free paths. A crucial disadvantage of OLP is the missing link between simulation used during programming and the real robot cell. Even small changes in the robot's environment have to be applied to the model in order to keep the model up-to-date. In contrast, online programming allows direct interaction between robot and operator providing methods and tools according to the physical proximity. Disadvantageous are however the limited computation capabilities of classic, monolithic robot controllers. Mostly, those controllers are limited to primitive interpolation between points, like for example linear or point to point movements. Combining these advantages of direct interaction with the availability for computational heavy tasks will therefore be advantageous.

The outcome of the previous section (s. Section 4) is the spatial transformation from a robot to the operator's viewpoint, e.g. HoloLens or hand-held device. Since it is computed in a distributed environment (s. Section 3), this architecture already allows processing various data acquired with arbitrary sensors. For instance, processing HoloLens's spatial measurements, as described in the previous sections enables building a map of the robot's environment.

In order to apply OLP functions to AR-based on-site programming, SLAM-based mapping services serve as real-time input to online path planning algorithms. In this manner, we are combining additional assistance through computational heavy services with on-site interaction between operator and industrial robots in terms of programming. Due to the nature of real-time 3D data acquisition and mapping, this process is not error-free resulting in reconstruction errors. Even planning based on perfect maps with standard path planning algorithms may lead to surprising results in terms of robot movements as can be seen in Fig. 10. Here, a simple path planning query does not lead to the expected point-to-point interpolation, but includes various via-points.

Since the map will never be perfect and is changing during operator's movements in combination with dynamic changes in the environment, only sampling-based path planning approaches can be chosen [46]. Due to the randomness of sampling-based algorithms, the computed paths will almost always yield to different results when computed again. Thus, this fact is leading to two difficulties:

- dealing with the computed paths and the via-points as well as
- dealing with incomplete maps as an input to path planning.

Both issues are tackled using the presented approach by explicitly involving the human operator into the path planning process. Consequently, this also applies especially to highly-dynamic environments, e.g. in HRC scenarios.

In case the sensors did not detect any obstacle due to a reflective surface or failures during reconstruction, the input to the path planning algorithms will be of poor quality. Thus, yield trajectories that are collision-free in the map, but not in reality. By presenting computed paths to the human operator, collision-free movements can be ensured,

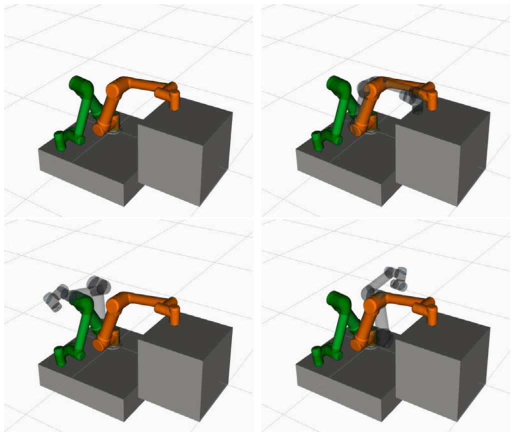

Fig. 10. A planned path in a perfect environment: robots start and goal pose (green and orange) and RRT-planned via-points (light grey). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

since human operators are capable of quickly distinguishing between space that seems to be free to the algorithms but are not in reality. Additionally, paths planned by the human operator can be checked for collisions since a map of the robot's environment is available. Dealing with the randomness of sampling-based path planning approaches is solved in the same way: manipulating the computed via-points and resulting re-planning only between changed points yields to trajectories and movements which are predictable by the operator.

An experimental HRC setup can be seen in Fig. 11. The operator's view through the head-mounted display, i.e. the HoloLens, is shown. According to Section 3 the ROS middleware is utilized to connect AR device, software services on a server and robot.

Since a computed map, i.e. a generated mesh for the HoloLens, of the robot's environment is available, collision checking between a robot path and the environment is made possible. Planning a path from the robot's initial pose to the point specified with the tap gesture leads to a collision with an object placed on the desk. It is presented to the operator by dyeing the robot red. In contrast, a collision-free path, which is generated by inserting a via-point, is shown in a different colour. By directly providing feedback to the operator, we are able to not only avoid collisions and costly repairs, but to also greatly reduce the time spent programming and commanding industrial robots in HRC scenarios.

6. Discussion and conclusion

We presented novel methods for overcoming obstacles to industrial applicability of AR in robotics. As an addition, we can state that we

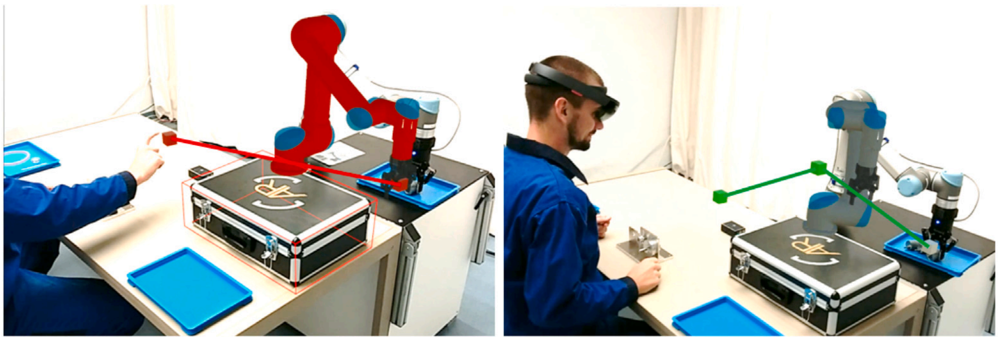

Fig. 11. Programming an industrial robot with the HoloLens AR device in a HRC scenario: a path with collision is programmed and highlighted to the operator (left), a collision free path is generated by inserting an additional via-point (right).

clearly see synergies between the three proposed solutions as a the resilient software architecture is the basis for efficiently offloading high performance computing services, e.g. to the edge, that enable automatic spatial calibration and novel programming features. Especially, the extended AR-based programming benefits from eliminating the manual spatial calibration providing an overall plug and play usage.

In regards to achieve resilience, we introduced a flexible, hardware-agnostic and expandable microservice architecture that enables the usage of computational support from cloud and edge computing and connects standard interfaces and communication middleware solutions. Consequently, the proposed software architecture is highly interoperable, supports distributed mobile computing through new radio networks, e.g. 5G, and automatic deployment as well as cloud and edge computing.

In addition, we presented automatic marker-less CNN-based pose estimation methods in order to reference AR device and robot's coordinate systems. Thus, the commissioning of AR application can be conducted fully automatically and without artificial aids. Specific CNN-based methods for different sensor setups that are applicable for stationary robotic manipulators as well as for mobile robots have been shown. It should be noted that accuracies of the proposed methods basically depend on the applicated sensor systems. Besides, robustness and accuracies of CNN-based methods heavily dependent on the quality of the training data. In this context, proper data engineering is essential taking into account synthetic data as human labels are a typical source for errors above all for pose estimation tasks. In the future, we recommend the provision of public datasets for pose estimation of robots as well as the usage of appropriate quality metrics in order to estimate resulting performance, e.g. accuracies and real-time capability.

Finally, we presented approaches towards the synergetic combination of online and offline programming through AR applications creating further benefits for applicators. In particular, we see a broad potential for added value in terms of merging virtual and real domains of robotics towards novel application scenarios of Digital Twins. For this purpose, AR and its sensor-equipped devices are crucial technologies making the virtual world visible to users and simultaneously sensing the real world. Especially in collaboration scenarios with a strong need for communication and synchronization, novel assistance functions from future work will support human decisions, optimizing robot trajectories and workflows as well as enhancing the overall safety.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] T. Masood, J. Egger, Augmented reality in support of Industry 4.0—Implementation challenges and success factors, J. Robot. Comput. Integr. Manuf. 58 (2019) http://dx.doi.org/10.1016/j.rcim.2019.02.003.

[2] J. Krüger, T.K. Lien, A. Verl, Cooperation of human and machines in assembly lines, CIRP Ann. - Manuf. Technol. 58 (2) (2009) 580–589, http://dx.doi.org/10.1016/j.cirp.2009.09.009.

[3] G. Schroeder, C. Steinmetz, C.E. Pereira, I. Muller, N. Garcia, D. Espindola, R. Rodrigues, Visualising the digital twin using web services and augmented reality, in: IEEE Int. Conf. on Industrial Informatics, 2016, pp. 522–527, http://dx.doi.org/10.1109/INDIN.2016.7819217.

[4] F. Tao, M. Zhang, Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing, IEEE Access 5 (2017) 20418–20427, http://dx.doi.org/10.1109/ACCESS.2017.2756069.

[5] J. Hügle, J. Lambrecht, J. Krüger, An integrated approach for industrial robot control and programming combining haptic and non-haptic gestures, in: IEEE Int. Symp. on Robot and Human Interactive Communication, 2017, pp. 851–857, http://dx.doi.org/10.1109/ROMAN.2017.8172402.

[6] G. Michalos, S. Makris, P. Tsarouchi, T. Guasch, D. Kontovrakis, G. Chryssolouris, Design considerations for safe human–robot collaborative workplaces, Proc. CIRP 37 (2015) 248–253, http://dx.doi.org/10.1016/j.procir.2015.08.014.

[7] A. Hietanen, R. Pieters, M. Lanz, J. Latokartano, J.-K. Kämäräinen, AR-based interaction for human–robot collaborative manufacturing, J. Robot. Comput.-Integr. Manuf. 63 (2020) http://dx.doi.org/10.1016/j.rcim.2019.101891.

[8] N. Kousia, C. Stoubosa, C. Gkournelosa, G. Michalosa, S. Makris, Enabling human robot interaction in flexible robotic assembly lines: an augmented reality based software suite, CIRP Conf. Manuf. Syst. 81 (2019) 1429–1434, http://dx.doi.org/10.1016/j.procir.2019.04.328.

[9] L.F. de Souza Cardosoa, F.C. Martins Queiroz Marianoa, E.R. Zorzal, A survey of industrial augmented reality, J. Comput. Ind. Eng. 139 (2020) 106159, http://dx.doi.org/10.1016/j.cie.2019.106159.

[10] J. Lambrecht, E.J. Steffens, M. Geiz, A. Vick, E. Funk, W. Steigerwald, Cognitive edge for factory: a case study on campus networks enabling smart intralogistics, in: IEEE Int. Conf. on Emerging Technologies and Factory Automation, 2019, http://dx.doi.org/10.1109/ETFA.2019.8869394.

[11] G. Evans, J. Miller, M.I. Pena, A. MacAllister, E. Winer, Evaluating the microsoft HoloLens through an augmented reality assembly application, in: Mechanical Engineering Conf. Presentations, Vol. 179, 2017, http://dx.doi.org/10.1117/12.2262626.

[12] J. Linowes, K. Babilinski, Augmented Reality for Developers: Build Practical Augmented Reality Applications with Unity, ARCore, ARKit, and Vuforia, Packt Publishing Ltd., 2017.

[13] E. Babaians, M. Tamiz, Y. Sarti, A. Mogoei, E. Mehrabi, ROS2Unity3D; high-performance plugin to interface ROS with Unity3d engine, in: Conf. on Artificial Intelligence and Robotics, 2018, pp. 59–64, http://dx.doi.org/10.1109/AIAR.2018.8769798.

[14] L. Peppoloni, F. Brizzi, C.A. Avizzano, E. Ruffaldi, Immersive ROS-integrated framework for robot teleoperation, in: IEEE Symp. on 3D User Interfaces, 2015, pp. 177–178, http://dx.doi.org/10.1109/3DUI.2015.7131758.

[15] P. Di Francesco, I. Malavolta, P. Lago, Research on architecting microservices: Trends, focus, and potential for industrial adoption, in: IEEE Int. Conf. on Software Architecture, 2017, pp. 21–30, http://dx.doi.org/10.1109/ICSA.2017.24.

[16] J.A. Jones, J.E. Swan, G. Singh, E. Kolstad, S.R. Ellis, The effects of virtual reality, augmented reality, and motion parallax on egocentric depth perception, in: Symp. on Applied Perception in Graphics and Visualization, 2008, pp. 9–14, http://dx.doi.org/10.1109/VR.2008.4480794.

[17] R. Bischoff, A. Kazi, Perspectives on augmented reality based human–robot interaction with industrial robots, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Vol. 4, 2004, pp. 3226–3231, http://dx.doi.org/10.1109/IROS.2004.1389914.
[18] H.C. Fang, S.K. Ong, A.Y.C. Nee, Robot path and end-effector orientation planning using augmented reality, Proc. CIRP 3 (2012) 191–196, http://dx.doi.org/10.1016/j.procir.2012.07.034.
[19] S.K. Ong, J.W.S. Chong, A.Y.C. Nee, Methodologies for immersive robot programming in an augmented reality environment, in: Int. Conf. on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia, 2006, pp. 237–244, http://dx.doi.org/10.1145/1174429.1174470.
[20] Y.S. Pai, H.J. Yap, R. Singh, Augmented reality–based programming, planning and simulation of a robotic work cell, Proc. Inst. Mech. Eng. J. Eng. Manuf. 229 (2015) 1029–1045, http://dx.doi.org/10.1177/0954405414534642.
[21] J.C. Gallo, P.F. Cárdenas, Designing an interface for trajectory programming in industrial robots using augmented reality, in: Proc. of the Latin American Congress on Automation and Robotics, 2019, pp. 142–148, http://dx.doi.org/10.1007/978-3-030-40309-6_14.
[22] S.K. Ong, A.W.W. Yew, N.K. Thanigaivel, A.Y.C. Nee, Augmented reality-assisted robot programming system for industrial applications, J. Robot. Comput. Integr. Manuf. 61 (2020) http://dx.doi.org/10.1016/j.rcim.2019.101820.
[23] D. Abawi, J. Bienwald, R. Dorner, Accuracy in optical tracking with fiducial markers: an accuracy function for ARToolKit, in: IEEE and ACM Int. Symp. on Mixed and Augmented Reality, 2004, pp. 260–261, http://dx.doi.org/10.1109/ISMAR.2004.8.
[24] B. Bellekens, V. Spruyt, R. Berkvens, M. Weyn, A survey of rigid 3d pointcloud registration algorithms, in: Int. Conf. on Ambient Computing, Applications, Services and Technologies, 2014, pp. 8–13.
[25] Y. Amagata, Robot Identification System, US patent US9905016B2, FANUC Corp., 2018.
[26] D. Puljiz, B. Hein, Concepts for end-to-end augmented reality based human-robot interaction systems, 2019, arXiv preprint arXiv:1910.04494.
[27] J. Bickendorf, New applications of cutting and welding robots with automatic offline-programming, in: Int. Symp. on Robotics, 2016, pp. 1–6.
[28] S.K. Ong, A.W.W. Yew, N.K. Thanigaivel, A.Y.C. Nee, Augmented reality-assisted robot programming system for industrial applications, J. Robot. Comput.-Integr. Manuf. 61 (2020) http://dx.doi.org/10.1016/j.rcim.2019.101820.
[29] J. Lambrecht, J. Krüger, Spatial programming for industrial robots based on gestures and augmented reality, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2012, pp. 466–472, http://dx.doi.org/10.1109/IROS.2012.6385900.
[30] J. Guhl, S. Nikoleizig, O. Heimann, J. Hügle, J. Krüger, Combining the advantages of on- and offline industrial robot programming, in: IEEE Int. Conf. on Emerging Technologies and Factory Automation, 2019, pp. 1567–1570, http://dx.doi.org/10.1109/ETFA.2019.8869495.
[31] J. Lambrecht, J. Krüger, Spatial programming for industrial robots: efficient, effective and user-optimised through natural communication and augmented reality, in: Advanced Materials Research, Vol. 1018, Trans Tech Publications Ltd, 2014, pp. 39–46, http://dx.doi.org/10.4028/www.scientific.net/AMR.1018.39.
[32] D. Krupke, F. Steinicke, P. Lubos, Y. Jonetzko, M. Görner, J. Zhang, Comparison of multimodal heading and pointing gestures for co-located mixed reality human-robot interaction, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2017, pp. 1–9, http://dx.doi.org/10.1109/IROS.2018.8594043.
[33] M. Bischoff, ROS#, Siemens AG, 2020, accessed on: Feb. 21, 2020, [online]: https://github.com/siemens/ros-sharp.
[34] T.E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, S. Birchfield, Camera-to-robot pose estimation from a single image, in: IEEE Int. Conf. on Robotics and Automation (ICRA), 2020, pp. 9426–9432, http://dx.doi.org/10.1109/ICRA40945.2020.9196596.
[35] J. Lambrecht, P. Grosenick, M. Meusel, Optimizing keypoint-based single-shot camera-to-robot pose estimation through shape segmentation, in: IEEE Int. Conf. on Robotics and Automation, 2021.
[36] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox, PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes, in: CoRR, 2017, vol. abs/1711.00199.
[37] V. Lepetit, F. Moreno-Noguer, P. Fua, EPnP: An accurate O(n) solution to the PnP problem, Int. J. Comput. Vis. 81 (2008) 155–166, http://dx.doi.org/10.1007/s11263-008-0152-6.
[38] J. Lambrecht, Robust few-shot pose estimation of articulated robots using monocular cameras and deep-learning-based keypoint detection, in: Int. Conf. on Robot Intelligence Technology and Applications, 2019, pp. 136–141, http://dx.doi.org/10.1109/RITAPP.2019.8932886.
[39] C. Heindl, S. Zambal, J. Scharinger, Learning to predict robot keypoints using artificially generated images, in: IEEE Int. Conf. on Emerging Technologies and Factory Automation, 2019, pp. 1536–1539, http://dx.doi.org/10.1109/ETFA.2019.8868243.
[40] J. Lambrecht, L. Kästner, Towards the usage of synthetic data for marker-less pose estimation of articulated robots in RGB images, in: Int. Conf. on Advanced Robotics, 2019, pp. 240–247, http://dx.doi.org/10.1109/ICAR46387.2019.8981600.
[41] Z. Zhao, G. Peng, C. Lu, Estimating 6d pose from localizing designated surface keypoints, 2018, arXiv preprint arXiv:1812.01387.
[42] B. Tekin, S. Sudipta, F. Pascal, Real-time seamless single shot 6d object pose prediction, in: IEEE/CVF Conf. on Computer Vision and Pattern Recognition, 2018, http://dx.doi.org/10.1109/CVPR.2018.00038.
[43] J. Redmon, F. Ali, Yolov3: An incremental improvement, 2019, arXiv preprint, arXiv:1804.02767.
[44] C.R. Qi, O. Litany, K. He, L.J. Guibas, Deep hough voting for 3d object detection in point clouds, in: IEEE Int. Conf. on Computer Vision, 2019, pp. 9277–9286, http://dx.doi.org/10.1109/ICCV.2019.00937.
[45] L. Kästner, V.C. Frasineanu, J. Lambrecht, A 3D-deep-learning-based augmented reality calibration method for robotic environments using depth sensor data, in: IEEE Int. Conf. on Robotics and Automation, 2020, pp. 1135–1141, http://dx.doi.org/10.1109/ICRA40945.2020.9197155.
[46] S.M. LaValle, Planning Algorithms, Cambridge University Press, 2006, accessed on: Feb. 21, 2020 [online]: http://planning.cs.uiuc.edu.