Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 107 (2022) 810–814

www.elsevier.com/locate/procedia

In55th CIRP Conference on Manufacturing Systems

Automated Commissioning of Offline-Generated Robot Programs

Lukas Tanz[a,*], Rüdiger Daub[a]

[a] *Institute for Machine Tools and Industrial Management – Technical University of Munich*
Boltzmannstr. 15, 85748 Garching

* Corresponding author. Tel.: +49-89-289-55465; fax: +49-89-289-1555. *E-mail address:* Lukas.Tanz@tum.de

Abstract

The flexibility of industrial robots accounts for their importance in reconfigurable production systems. Advanced methods of offline programming based on 3D simulation promise seamless adaptation to changing circumstances. However, since digital planning models generally do not match reality, the commissioning of computer-generated robot programs involves extensive manual adjustment to the on-site conditions. This paper presents a concept for the automated commissioning of offline-generated robot programs to reduce downtimes of industrial robots and to increase the process consistency of offline robot programming. The presented approach aims for a hybrid motion control of industrial robots that automatically compensates for deviations between simulation and reality. For this purpose, robot poses are described not only by the joint positions but also by the anticipated sensor information. During process execution, robot poses are adjusted as required on the basis of the sensor data collected online.

Keywords: offline robot programming; visual servoing; reality gap; domain randomization

1. Introduction

Production industries today face a wide range of challenges. Especially shorter product life cycles, smaller batch sizes, and an increasing number of product variants need to be managed [1]. As the final manufacturing step, assembly is the point at which variants are formed and is therefore significantly affected by these trends. Reconfigurable assembly systems (RAS) offer the flexibility to maintain high productivity under such challenging conditions [2]. Due to their inherent flexibility, industrial robots are essential in the design of RAS. Therefore, not only the number of robots used but also the variety of tasks performed per robot system are growing continuously, which leads to a tremendous increase in the effort involved in robot programming [3, 4].

In practice, industrial robots are programmed either online or offline. Play-back and Teach-in are two of the most common online programming techniques meaning that they are performed directly on the robot in the production environment. Such programming procedures are relatively simple, since they do not require knowledge of a specific programming language. However, the robot is blocked during programming, and the implementable complexity is highly restricted. Furthermore, even small changes in the process may necessitate complete reprogramming.

In contrast, offline programming generally involves simulating the robot application, which means that programming is mainly done at the computer, thus avoiding the aforementioned disadvantages associated with online methods. It is therefore the predominant method employed in industrial practice [5]. Nevertheless, since the underlying digital planning models do not exactly match the real world, the commissioning of computer-generated robot programs requires extensive manual adjustments to the conditions prevailing on-site [6]. These include the calibration of the components involved in the process (robot, workpiece, tools, and environment) as well as the correction of individual robot poses. Considering the rising programming effort, this process step, in which simulation-

10.1016/j.procir.2022.05.067

based robot code is transferred to the actual application, is increasingly problematic.

In this paper, we present a new concept for the automated commissioning of offline-generated robot programs. It focuses on utilizing visual features from camera images as additional references to describe robot poses. This information is processed using artificial intelligence (AI) trained on synthetic data and used to compensate for deviations between simulation and reality. Our approach thus reduces both manual effort and system downtimes.

The rest of the paper is structured as follows: Section 2 reviews the state of the art in robot programming and vision-based robot control. Section 3 highlights the need for action. In Section 4, we present our concept and discuss its crucial aspects. Finally, Section 5 summarizes our paper and provides an outlook of our future work.

Nomenclature

AI	Artificial Intelligence
AOLP	Automated Offline Programming
CNN	Convolutional Neural Network
IBVS	Image-based Visual Servoing
PBVS	Pose-based Visual Servoing
RAS	Reconfigurable Assembly System
TCP	Tool Center Point
VS	Visual Servoing

2. State of the art

2.1. Automated offline robot programming

To deal with the increasing programming effort of industrial robots, researchers are focusing on the automated generation of robot code. Automated offline programming (AOLP) uses CAD models and sensor data to generate robot programs for predefined processes such as welding [7] or dispensing [8] with as little human intervention as possible. Two different approaches can be distinguished. Either the real conditions are captured by a vision sensor and the simulation is accordingly updated [8, 9], or a calibration routine is used to compensate for deviations between simulation and reality [7].

In [8], process knowledge is stored in a database linked to the CAD model. Thus, after the actual part pose is determined by processing a depth image, the functionalities of a commercial offline programming tool are used to derive robot poses and trajectories. The solution presented in [9] is similar, but the programmer defines the working path in relation to the workpiece. The robot trajectory is then created automatically. However, these approaches require a robot application to be running at programming time. Accordingly, the programming and commissioning processes are highly dependent on the system setup and cannot be prepared in advance.

In contrast, calibration-based solutions enable complete programming in advance premised on a CAD model of the robot cell. At commissioning time, the robot program is adjusted to suit the on-site conditions by processing sensory feedback. [7] considers touch sensing and laser profilometry to determine the actual workpiece pose for a subsequent welding process. An error function is calculated from this information to correct the work path or rather the trajectory of the robot. The adjustment can also be done during process execution by a feedback control [10, 11]. However, both cases require a continuous feature such as a weld seam, by which the robot motion is aligned.

In conclusion, from a robotics point of view, AOLP is mainly restricted to relatively simple processes centered on a single workpiece and with no direct coupling between the robot and its environment. Assembly and handling applications do not comply with these limitations. However, they account for more than half of all installed robot applications, which emphasizes the need for a new solution [4].

2.2. Task-oriented programming

Task-oriented programming is an active field of research that aims to simplify robot programming. Instead of describing the way in which a robot is supposed to do something, the focus is on the task itself and the goal to be achieved. By encapsulating robot functions as reusable and parameterizable skills, [12] presents a method that enables task-level robot programming through simple concatenation of these skills. This reduces the degree of expertise required for robot programming.

Automated task programming goes one step further by merely requiring a goal state to be provided by the end-user. The robot system itself then decides which skills to execute so as to reach the desired state. [13] and [14] present the three main building blocks of such a system. The job of the task planner is to translate the task description into a sequence of appropriate skills. This requires a knowledge base, referred to as the world model, which describes and continuously updates the structure of the robot's environment. The skill manager or task controller provides the runtime for skill execution.

Task-oriented programming is of great value in dealing with the increasing programming effort encountered in industrial robotics. While commercial task-level programming solutions already exist, the autonomous scheduling and execution of skills is still in its infancy. One major drawback are the sensorial capabilities required to manage the uncertainty of reality.

2.3. Visual Servoing

As a result of the high information density, declining prices of sensory hardware, and advances in data processing afforded by AI, machine vision is becoming an increasingly important aspect of industrial robotics. However, robot control based on visual information has been the subject of research for more than three decades. When implemented in a closed-loop, it is commonly referred to by the term 'visual servoing' (VS).

Two main classes of VS exist. The first is position-based VS (PBVS), in which the image is processed to extract the pose of a target object, enabling the error of the robot pose to be calculated [15]. 3D cameras are often used for this purpose. The other class is image-based VS (IBVS), in which the control error is determined and compensated for directly within the feature space of the 2D image [15]. Both approaches have their advantages and disadvantages, leading to the development of hybrid techniques combining PBVS and IBVS [16].

A VS solution designed for smartphone back shell assembly is presented in [17]. The robot deployed carries two monocular cameras in a so-called eye-in-hand configuration. The overlap in the fields of view of the two cameras enables the 3D pose of the smartphone to be reconstructed. As the robot moves towards the smartphone, the camera images become disjoint, and control is switched from PBVS to IBVS, using the corners of the smartphone as image references. [18] and [19] are examples of pure IBVS implementations. While [18] also uses an eye-in-hand configuration to control a robot to plug a USB connector into a hub, the authors of [19] deployed two stationary cameras to solve a peg-in-hole task.

However, all these implementations have a common drawback. The image processing can only be used for the considered use case since specific features such as the corners of a particular smartphone, the outer contours of a USB hub, or the shape of a bolt are extracted and processed. This means that the image processing is not scalable and needs to be revised even in the event of only minor changes to the robot application. To date, this necessitates a high degree of engineering effort.

3. Motivation

Since the number of industrial robot applications continues to grow, simplification and automation of robot programming have become major fields of research. Despite the essential meaning of sensor processing for those approaches its implementation is usually neglected. However, little is achieved when the engineering effort for robot programming is shifted to the configuration of sensor processing.

Based on AI our concept promises the automated configuration of application-specific image processing for automated commissioning of robot programs. The main goal is to integrate the generation of synthetic image data and the identification of visual references in the workflow of simulation-based robot programming. Subsequently, advanced AI methods can be applied to efficiently compensate for deviations between reality and the virtual world by VS.

4. Automated commissioning of offline-generated robot programs

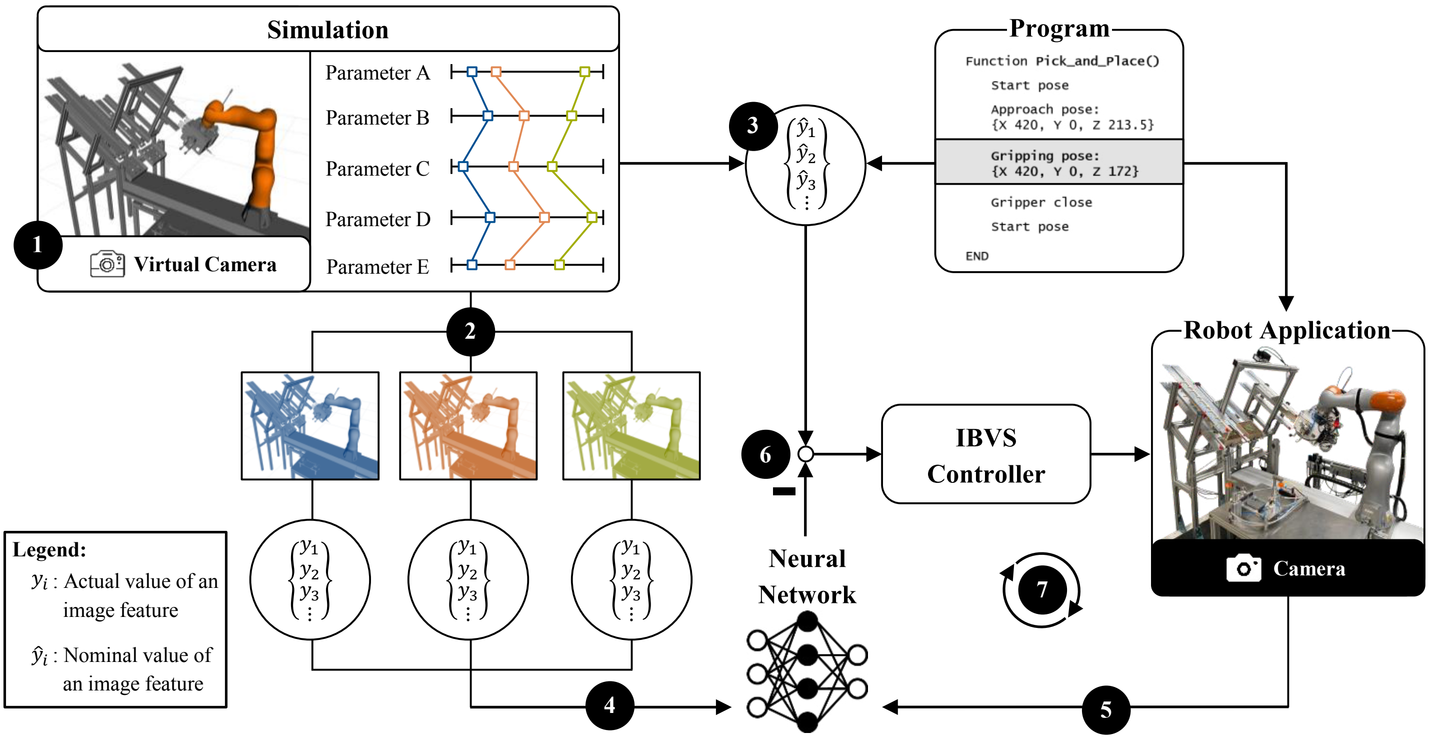
4.1. Concept Overview

The proposed concept for the automated preparation and execution of commissioning offline-generated programs comprises seven steps (see Fig. 1).

Step 1: First, we expand the simulation used for offline programming by adding a virtual camera to derive a 2D projection of the 3D simulation. The single images created represent the robot's perception through a hand-in-eye camera.

Step 2: Here, the simulation parameters, such as lighting conditions or object positions, are varied. This generates different instances of a scene, and an infinitely large data set of synthetic images can be captured by the virtual camera. We use guided domain randomization, as described in Section 4.2, to ensure that the AI trained on this data (see Step 4) also applies to real-world camera images.

Step 3: In this step, particular image features are automatically determined as landmarks. The original robot code is then augmented by the nominal values of these image features as additional pose references. In terms of AI training, the grouped landmarks of an image represent the corresponding



```
Function Pick_and_Place()
    Start pose
    Approach pose:
    {X 420, Y 0, Z 213.5}
    Gripping pose:
    {X 420, Y 0, Z 172}
    Gripper close
    Start pose
END
```

Fig. 1. Concept for the automated commissioning of offline-generated robot programs

image label. The automatic selection of appropriate image features is a crucial aspect of our concept. We discuss this further in Section 4.3.

Steps 4 and 5: In preparation for the rollout of a new robot program, a neural network is trained to detect landmarks based on synthetic data. At commissioning time, the neural network interprets real-world camera images to determine the actual landmark values. We use Convolutional Neural Networks (CNN) since they have proven their great potential for landmark detection in other fields of application [20,21].

Steps 6 and 7: The best fit to their nominal values is calculated from the actual image feature manifestations. Compensation for any deviation is then performed by IBVS. While the motion commands from the original robot code are executed, the IBVS acts as a supervisor that monitors the camera data collected online and adjusts single robot poses as needed.

4.2. Synthetic data generation

Generating a sufficient amount of real-world data for CNN training would be more costly than manual commissioning. So instead, synthetic images derived from the simulation are required. However, since a simulation is just a simplified version of reality, knowledge gained from simulated experience cannot be transferred directly to reality. This is commonly referred to as the reality gap.

In recent years, researchers have developed a number of strategies to bridge the reality gap. Among the most promising ones, particularly with vision-based robot applications, is domain randomization presented by [22]. In a simulation used for CNN training, rather than faithfully modeling reality, parameters that are hard to control and likely to interfere with image processing, such as lighting conditions or surface textures, are subjected to intense variation. This forces the network to learn a representation of the images that is as independent of those parameters as possible. The successful application of the CNN to real-world data has demonstrated the potential of this approach [22]. However, the achieved result quality decreases if the simulation is too heavily randomized. This is also shown in [23], in which performance loss is observed when too much randomization exists in simulation-based training of a deep-reinforcement learning agent for the robotic handling of deformable objects.

We adapt the domain randomization approach in our concept. To avoid performance loss caused by excessive undirected randomization, we distinguish between heuristic and diffuse parameters (see Fig. 2). While both parameter types are likely to disturb image processing, only the scattering of heuristic parameters can be estimated at programming time. Diffuse parameters are handled in the manner already familiar from domain randomization, i.e., scatter is selected to be unrealistically large. In contrast, heuristic parameters are evaluated in more detail. The aim is to obtain a reasonable assessment of the underlying value range and probability distribution.

Representative use cases are selected and analyzed from the vast number of robot-based assembly and handling applications. In particular, the effects of uncertainty in the individual components on the overall system are assessed by a

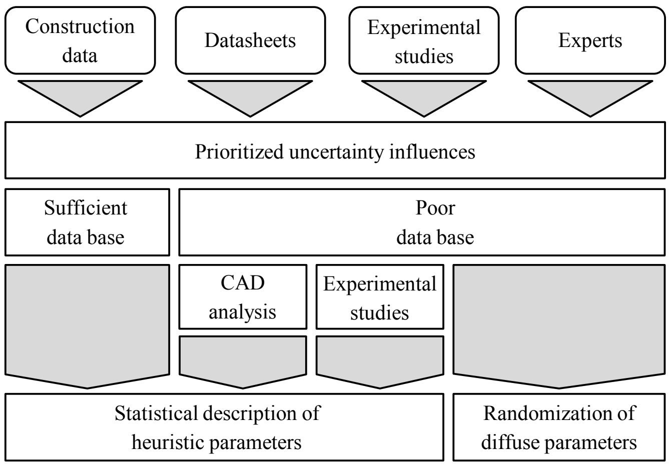


Figure 2. Classification of uncertainty effects

cause-effect analysis based on the various data generated when engineering a new robot application. As indicated in Fig. 2, the primary sources are construction data, datasheets, experimental research studies, and implicit expert knowledge. Evaluation of this information results in a prioritized list of uncertainty effects based not only on their extent but also on their predictability and the existing data basis. Whereas diffuse parameters in the simulation represent uncertainty effects that are hard to model, well-described influences affect the simulation via heuristic parameters with a realistic value distribution. However, insufficient data does not automatically lead to diffuse parameters. According to their priority, underlying uncertainty effects are further investigated. Here, we perform a CAD analysis of simple, task-specific structures such as assembly fixtures to quantify position inaccuracies. For application-specific components that are more complex (e.g., a particular robot), experimental studies are considered.

4.3. Image feature selection

In addition to the data, labels are required to train the CNN. In scientific studies, a target object's pose is commonly chosen for this purpose. However, since parts are usually processed in a constraint position, the assumption that a sufficient amount of a part's features will be within the camera's field of view does not apply to industrial applications. Neither is the use of artificial landmarks such as QR codes or apriltags appropriate, as it would require a significant manual effort to attach them to the robot's environment. We therefore concentrate on natural, application-specific image features as landmarks.

According to [16, 24] and taking the nominal distances between feature points into account, at least three image points are required to determine the six degrees of freedom of an industrial robot. However, the fact that the exact characteristics of the label or rather the values of the image features in the real environment are unpredictable at programming time makes the selection difficult. Due to reflections or positioning inaccuracies, it is possible that certain image features are not even visible in the actual camera image. Furthermore, a holistic consideration of the deviations between simulation and reality is essential. Otherwise, there is a risk of overfitting local outliers. Consequently, more than just the theoretically required set of image features are needed to obtain an overdetermined system of equations that can be solved by the least squared error method.

In order to define an appropriate label, all features of a geometrical class, such as corners and midpoints, are extracted from the simulated scene. The transformation of the virtual camera then derives their image manifestations. Theoretically, all these features can be grouped as one label. Pre-processing is considered as a way of reducing the data load and avoiding the problems associated with this. Potentially poor image features are filtered, and the label size is reduced, by applying assessment criteria and thresholds. From our perspective, two different types of evaluation (or a combination of both) would seem reasonable. The first option is to consider the subsequent visual servoing feedback loop. The criteria are concerned with the following aspects:

- *Uniqueness*: To avoid ambiguity errors, the image features must be locally unique.
- *Detectability*: The image features must be detectable under varying environmental conditions.
- *Controllability*: A distinct robot motion for deviation compensation must be derivable.

Assuming that the process is mainly driven by the robot, the second reasonable option is to consider the Euclidean distance between features and the TCP. In this case, the number of landmarks decreases as the TCP distance increases.

5. Conclusion and Outlook

The concept presented here promises to efficiently compensate for deviations between simulation and reality by utilizing image features as additional references for describing robot poses. This accelerates the commissioning of offline-generated robot code and reduces the required manual effort. Therefore, we described solution strategies for the crucial aspects of data generation and labeling. By advancing domain randomization to guided domain randomization, we retain useful data patterns in the synthetic image data. Thus, we expect increasing accuracy and precision in the task of landmark detection. However, the identification of robust natural landmarks without the possibility of analyzing real-world data or at least the environmental conditions prevailing on-site remains a major challenge.

The practical applicability of our concept requires a high degree of automation of the single process steps to avoid additional effort. While this is already given for the CNN training and the randomization of diffuse parameters we will focus on methods and guidelines to determine heuristic parameters. Furthermore, we will experimentally evaluate different approaches for landmark selection. Therefore, the concept is intended to be tested with typical assembly and handling operations. By deliberate manipulations of the environmental conditions and statistical analysis, the reliability and application limits will be also investigated. In a later step, we plan to consider different neural network structures. For instance, linking region-specific networks shows great potential for facial landmark detection [25].

References

[1] Lasi, H., Fettke, P., Kemper, H.-G., Feld, T. Hoffmann, M., 2014. Industry 4.0, Bus Inf Syst Eng 6, pp. 239-242.

[2] Koren, Y., Gu, X., Guo, W., 2018. Reconfigurable manufacturing systems: Principles, design, and future trends, Front. Mech. Eng. 13, pp. 121-136.

[3] International Federation of Robotics, 2018. Artificial Intelligence in Robotics. Retrieved from https://ifr.org/papers.

[4] Müller, C., Kutzbach, N., 2019. World Robotics 2019 - Industrial Robots. VDMA Services GmbH, Frankfurt.

[5] Hägele, M., Nilsson, K., Pires, J. N., Bischoff, R., 2016. Industrial Robotics, in "Springer handbook of robotics" B. Siciliano, O. Khatib, Editors. Springer, Berlin Heidelberg, pp. 1385-1422.

[6] Villani, V., Pini, F., Leali, F., Secchi, C., Fantuzzi, C., 2018. Survey on Human-Robot Interaction for Robot Programming in Industrial Applications, IFAC-PapersOnLine 51, pp. 66-71.

[7] Larkin, N., Short, A., Pan, Z., van Duin, S., 2018. Automated Programming for Robotic Welding, in "Transactions on Intelligent Welding Manufacturing" S. Chen, Y. Zhang, Z. Feng, Editors. Springer, Singapore, pp. 48-59.

[8] Maiolino, P., Woolley, R., Branson, D., Benardos, P., Popov, A., Ratchev, S., 2017. Flexible robot sealant dispensing cell using RGB-D sensor and off-line programming, Comput. Integr. Manuf. 48, pp. 188-195.

[9] Bedaka, A.K., Vidal, J., Lin, C.-Y., 2019. Automatic robot path integration using three-dimensional vision and offline programming, Int. M. Adv. Manuf. Technol. 102, pp. 1935-1950.

[10] Graaf, M., 2007. Sensor-guided robotic laser welding, Twente, University of Twente.

[11] Huang, Y., Xiao, Y., Wang, P., Li, M., 2013. A seam-tracking laser welding platform with 3D and 2D visual information fusion vision sensor system, Int. J. Adv. Manuf. Technol. 67, pp. 415-426.

[12] Crosby, M., Rovida, F., Pedersen, M. R., Petrick, R. P. A., Krüger, V., 2016. Planning for robots with skills, PlanRob, ICAPS, London, pp. 49-57.

[13] Rovida, F., Crosby, M., Holz, D., Polydoros, A.S., Großmann, B., P. A., R., Krüger, P., Krüger, V., 2017. SkiROS - A Skill-Based Robot Control Platform on Top of ROS, in "Studies in Computational Intelligence" A. Koubaa, Editor. Springer, Cham, pp. 121-160.

[14] Heuss, L., Reinhart, G., 2020. Integration of Autonomous Task Planning into Reconfigurable Skill-Based Industrial Robots, ETFA, IEEE, Vienna, pp. 1293-1296.

[15] Hutchinson, S., Hager, G.D., Corke, P.I., 1996. A tutorial on visual servo control, IEEE Trans. Robot. 12, pp. 651-670.

[16] Chaumette, F., Hutchinson, S., Corke P., 2016. Visual Servoing, in "Springer handbook of robotics" B. Siciliano, O. Khatib, Editors. Springer, Berlin Heidelberg, pp. 841-866.

[17] Chang, W.-C., 2018. Robotic assembly of smartphone back shells with eye-in-hand visual servoing, Robot. Comput. Integr. Manuf. 50, pp. 102-113.

[18] Song, H.-C., Kim, M.-C., Song, J.-B., 2015. USB assembly strategy based on visual servoing and impedance control, URAI, IEEE, Goyang city, pp. 114-117.

[19] Ma, Y., Liu, X., Zhang, J., de Xu, Zhang, D., Wu, W., 2020. Robotic grasping and alignment for small size components assembly based on visual servoing, Int. J. Adv. Manuf. Technol. 106, pp. 4827-4843.

[20] Kunz, F., Stellzig-Eisenhauer, A., Zeman, F., Boldt, J., 2020. Evaluation of a fully automated cephalometric analysis using a customized convolutional neural network, J Orofac Orthop. 81, pp. 52-68.

[21] Feng, Z.-H., Kittler, J., Awais, M., Huber, P., Wu, X.-J. 2018. Wing Loss for Robust Facial Landmark Localisation with Convolutional Neural Networks, CVPR, IEEE, Salt Lake City, pp. 2235-2245.

[22] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P., 2017. Domain randomization for transferring deep neural networks from simulation to the real world, IROS, IEEE, Vancouver, pp. 23-30.

[23] Matas, J., James, S., Davison, A. J., 2018. Sim-to-Real Reinforcement Learning for Deformable Object Manipulation, PMLR 87, pp. 734-743.

[24] Feddema, J.T., Lee, C., Mitchell, O.R., 1991. Weighted selection of image features for resolved rate visual feedback control, IEEE Trans. Robot. 7, pp. 31-47.

[25] Chandran, P., Bradley, D., Gross, M., Beeler, T., 2020. Attention-Driven Cropping for Very High Resolution Facial Landmark Detection, CVPR, IEEE, pp. 5861-5870.