Gesture-based Robot Programming Enabling Non-Sequential Control Flow

Fabian Mikula, Sascha Sucker, and Dominik Henrich

Chair for Robotics and Embedded Systems, University of Bayreuth,
Universitätsstraße 30, D-95447 Bayreuth, Germany
`Fabian.Mikula@uni-bayreuth.de`,
WWW home page: `http://www.robotics.uni-bayreuth.de/`

Abstract. Medium-sized companies have the potential to automate numerous tasks. However, this often faces challenges due to small batch sizes with costs outweighing the benefits. One solution is enabling non-experts to program robots with an intuitive interface. Prior research hardly focused on gestures for such an interface. In this paper, we contribute a concept for a unimodal control structure-based robot programming system. To this end, we present a concept and prototype that records the user's gestures, classifies them and converts them into a robot program with control structures. Further, we present methods for error detection. We assessed the usability of our prototype with a user study resulting in a median SUS score of 81.25. We found that users can program both simple and complex tasks in less than 2 minutes on average.

Keywords: gesture programming, intelligent robots, intuitive robot programming, human-robot interaction

1 Introduction

Automating tasks presents a significant value for companies [21]. Nevertheless, robot programming demands expertise, incurring high costs [20, 21]. A common solution lies in having a robot capable of intuitive and flexible reprogramming by non-experts enabling seamless adaptation to evolving task requirements [20, 21]. Research is currently underway to increase programming accessibility using natural language [23, 22], kinesthetic guidance [25], augmented reality [18, 19, 11], or gestures [15, 14]. *Gestures* are body movements conveying ideas or meanings [28]. Further, they are applicable in loud environments and can express complex ideas (e.g., sign language). Thus, gestures offer a promising interface for intuitive human-robot interaction [27, 26].

This paper targets intuitive robot programming of complex tasks using only gestures (see Fig. 1). In robot programming, gestures often supplement a main input modality (e.g., voice commands in [14]). Here, we specifically focus on gestures as unimodal programming input modality. *Programming* refers to complex control flow and parametrization of robot skills differing from simple motion primitives – thus, allowing the expression of complex tasks. Such control flow

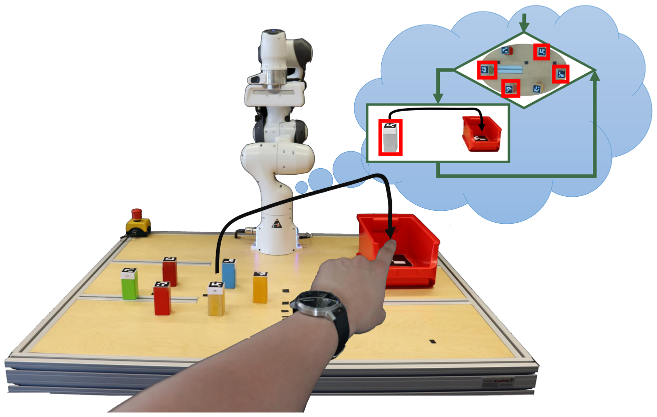


Fig. 1: We envision robot gesture programming to instruct complex tasks. Here, we instruct a for-each loop to place the selected objects into the box.

encompasses various control structures, including sequences, branches, and loops (for, while, and for-each).

As our contribution, we present intuitive robot programming of complex tasks using only gestures (Sec. 3). This includes a dynamic world representation, gesture recognition (Sec. 3.1), and program construction including syntax (Sec. 3.2) and semantic interpretation (Sec. 3.3). Based on a user study, we examine the usability of robot programming with gestures only, given our prototype (Sec. 4).

2 Related Work

Research in robot programming using gestures focuses significantly on teleoperations. In this method, users often direct the robot to move in increments in a specific direction through gestures [1–5]. Another form of teleoperation involves extracting the skeletal model of the user from a video stream with depth data and transferring the model's joint positions to a humanoid robot [6] or robot arm [7] in real-time. Alternative approaches specify the position of a robot, using arm movements or predefined hand gestures that trigger a linked robot program [8–11, 16]. However, these programming methods cannot handle more complex tasks as no branches are supported. Furthermore, programming techniques that connect robot programs with gestures would necessitate expert knowledge to program the robot's movements.

To program more complex tasks, we must extract control structures from gestures. For instance, there are methods for programming pick and place operations using hand gestures [12, 14, 13, 15, 17]. Individual objects [12, 13, 15, 17] or groups [14] can be selected using pointing gestures and placed at the target location. Some of these approaches are multimodal [14][17], allowing for input in multiple forms. For example, voice commands and gestures can be used for pick

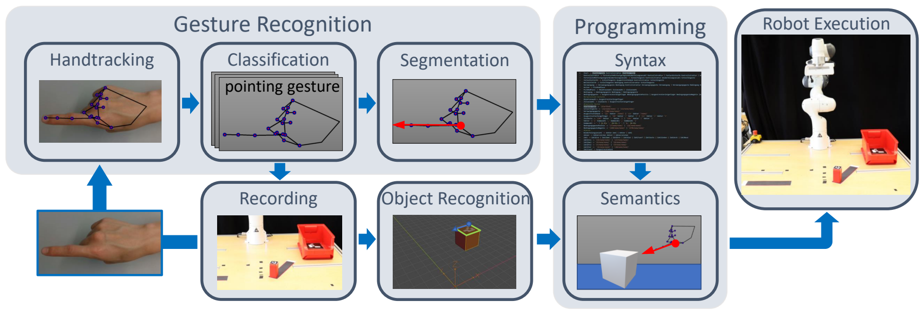


Fig. 2: This is an overview of the programming process with gestures. Gesture and Object Recognition are done parallelly. The programming step interprets both results to create the program for robot execution.

and place operations [14, 17]. However, these implementations only include control structures such as sequences for the successive execution of operations (all works) or the for-each loop [14] for moving entire selected object groups. Other control structures need to be included in these approaches, thus not allowing complex programs, which is the central focus of this work.

Additionally, prior research explored gesture programming in augmented reality, which specifies the target position of an object more precisely. For instance, augmented reality allows the user to grasp and move virtual objects, enabling the parameterization of both the object orientation and target position [18, 19]. Another approach uses object relations to define target positions through gesticulation [15]. However, we aim to limit ourselves to the most necessary hardware.

Regarding programming with gestures, prior approaches focus on control structures such as sequences (all works), for-each loops [14], and for loops [16]. However, they lack the integration of branches and while-loops, which are essential for the control flow of complex programs. Therefore, this work fills the gap by focusing on programming control structures using gestures.

3 Robot Programming with Gestures

This work aims to realize the programming of the control structures using gestures. The first step in the programming process with gestures is to record a video stream with depth information (Recording, Fig. 2). For gesture recognition, a hand skeleton model, consisting of markers with a fixed position on the hand, is extracted from the video stream. A neural network classifies these markers into gesture tokens. After the classification, the tokens are filtered to avoid noise (unintended gestures) contributing to the program's correctness. The filtered gesture tokens are analyzed syntactically to further check correctness utilizing a context-free grammar. After this, a semantic parsing is conducted to generate a control structure based robot program consisting of semantic gesture information (Programming, Fig. 2). Before execution, the semantic gesture information is analyzed and checked for semantic correctness in the relevant context. For instance, in a pick and place operation, it is verified whether the selected objects exist or the target position is reachable. Finally, the robot executes the program.

3.1 Object and Gesture Recognition

During *object recognition* we detect and localize objects in the robot's workspace creating a dynamic world model [31, 30]. For this, we require a video stream with depth information from a calibrated sensor encompassing the workspace. Concurrently, in *gesture recognition*, we extract a sequence of gesture tokens from image data of the same video stream. Gesture recognition can be achieved using a neural network, typically through CNN-based approaches [32, 33]. However, the programming system can be complex to train due to the varying characteristics of many individuals, such as different skin colors. Moreover, incorporating the orientation of the gesture into the training data presents a challenge. Therefore, we have implemented gesture recognition based on hand pose estimation. Hand pose estimation can be approached through vision-based or sensor-based methods [36]. Vision-based approaches often use neural networks to determine a *hand skeleton model* M from an input image I containing individual hand skeleton *markers* $m_i \in M \subseteq I$ [36]. Sensor-based methods often use a data glove to measure the bending angle and the level of adduction of each finger captured by embedded sensors [36]. We opted for the vision-based neural network approach as publicly available software (e.g., media pipe [35]) can generate 21 markers M. Furthermore, there is no need for additional hardware on the body. To recognize the gestures, we use a neuronal network, which receives the hand skeleton markers M as input and delivers the gesture class as output. There is no pre-trained neural network that classifies gestures for programming control structures. Thus, a custom neural network is required (e.g., on the basis of [33]). The neural network's training data can be generated by recording a gesture over several frames and storing the extracted labeled data. However, the training data would be required to cover all orientations and positions of a gesture for accurate classification. To remedy this, we normalize the training data for translation and rotation invariance. We achieve translation invariance of the hand skeleton model by specifying all markers m_i relative to the base marker m_0. We obtain rotational invariance by aligning the hand skeleton model with the y-axis. Accordingly, we can use this normalization to record fewer training samples.

Natural behavior, such as thinking, may occur during the programming process. It is essential to acknowledge this as it can impact the correctness of the program. To prevent a false robot program, we have to filter out gestures irrelevant to the programming task. For example, suppose a user extends his index finger as he thinks about which object to point next. In that case, the inadvertent extension of the index finger is also accidentally recognized as a gesture. It is feasible to solve this problem in several ways. One possibility would be to analyze the posture data and the direction of the programmer's gaze and deduce whether the user is currently programming or engaging in natural behavior such as thinking. However, this requires further research and needs to be considered in another paper. A more straightforward solution would be to enforce a specific dwell time for the gestures before transmitting them to the robot controller. This would result in minor uncertainties and hand movements being ignored. We implement this type of filtering by looking at the probability of a gesture occurring in a time window. If this probability exceeds a certain threshold, the gesture is then added to the program's gesture token stream.

3.2 Program Construction with Control Structures

This work aims to realize the programming of the control structures using a *context-free grammar* $G = (N, T, P, S)$. The production rules P map the non-terminal symbols N into sequences of terminal symbols T, where a terminal symbol $t \in T$ corresponds to a single gesture. From the start symbol S, all words w of the language $L(G)$ are derived from the grammar G. Using context-free grammar implies that the set of all terminal symbols is interchangeable without losing the functionality of the language. Furthermore, the assignment of the words $w \in L(G)$ can be checked for syntactic correctness, allowing us to decide whether a gesture sequence w makes sense and is part of the defined language, which is essential for the generation of a correct robot program. In addition, semantic parsing enriches the grammar to ensure executable code.

The program is constructed by first sending the filtered gestures to the robot programming instance, which checks the transmitted gesture sequence for affiliation to the language $L(G)$ by syntactic analysis. The framework of each control structure can only be constructed as part of the program using syntactic analysis if the program of the gesture is syntactically correct. Therefore, each control structure has a grammar G', a subset of the grammar G from which we derive the control structures. When we recognize a word $w \in L(G')$, we apply semantic parsing to generate a control structure using the semantic information from the gestures. The terminal symbols of the grammar result from a preliminary study with three participants. First, the participants are gesticulating an instruction for a task and then executing a gestured instruction. However, the choice of gestures requires a further, more elaborate user study with more participants. All loops start and end with an initial gesture. Following this, we provide loop-specific instructions, such as a condition (thumbs up, down, or an interval with index finger and thumbs) or an object selection with the flat of the hand over an object group. Now, we can define the loop body, which can contain further instructions. We conclude loops with an end gesture that corresponds to the start gesture. The only exception is the for loop with a fixed number of repetitions, particularly if the number 5 is gesticulated. Therefore, we define the number of repetitions of the for loop at the end. This is important to avoid confusion when selecting objects with a flat hand in a for-each loop. The number n of repetitions can be defined in different ways. One possibility is to define n by gestures from the most significant digit x_p to the least significant digit x_0. This method is independent of the base b of n and can be calculated with $n = \sum_{i=0}^{p} x_i \cdot b^i$. For humans, the decimal system ($b = 10$) is suitable because of the ten fingers. Another possibility is an additive method, in which we calculate n as the sum of the numbers x_i shown in the numerical gestures. The last possibility is a hybrid approach. For the calculation of $n = c \cdot 10 + d$, we first add the number c of tens and then the digits d. This was commonly used in our preliminary study leading us to choose this approach.

Branching is initiated with a branching gesture, followed by a set of conditions represented by gestures such as thumb down for negative, thumb up for positive, and an interval. To define the interval, we use the distance between the thumb and index finger. Now, we can define the body of the branch and terminate the branch with a branch gesture. In the case of multiple conditions in a single branch, the logical 'and' is applied to them. Several branches can be instructed in succession to implement the logical 'or'.

3.3 Semantic Interpretation

Following the program setup step, the control structures and operations provide an understanding of the context of individual gestures. However, the deictic gestures, such as pointing gestures and object selection with a flat hand in the for-each loop, require interpretation and semantic verification to ensure an executable program. For instance, verifying if the objects have been selected is necessary during the pick and place operation. Additionally, we indicate an accessible point on the table when selecting a target by a pointing gesture. For interpreting the pointing gestures, we calculate a line

$$g = \hat{m_5} + \lambda \cdot (\hat{m_7} - \hat{m_5}); \text{ with } \lambda \in \mathbf{R}, \hat{m_i} \in \hat{M} \tag{1}$$

from the hand skeleton model, where $\hat{m_i} \in \hat{M}$ represents the world coordinates of the marker i from the hand skeleton model M. The program checks whether the straight line g intersects with one of all oriented bounding boxes of the world model. If it is an object selection and the intersection test is positive, the corresponding object is noted in the pick and place operation. If the cut test of an object selection is negative, an error is output, and the program is not executed. When selecting a target, a positive intersection test results in stacking, and a negative intersection test results in calculating the intersection point between the worktop and the straight line g. We model the worktop as a plane D, resulting in the following intersection test:

$$E = u \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + v \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = g; \text{ with } u, v \in \mathbf{R}. \tag{2}$$

For the group selection, we project the palm's center point

$$p = \frac{1}{n} \cdot \sum_{i=1}^{n} \hat{m_i}; \text{ with } n = |\hat{M}|, \hat{m_i} \in \hat{M} \tag{3}$$

onto the plane E of the robot's work surface. Next, we perform a K-Means clustering for all objects in the world model. Then, we select the object group closest to the projected point p of the palm when p lays within a radius from the nearest cluster's center. An error message will be displayed if no cluster is found near the palm. Otherwise, the program can be executed.

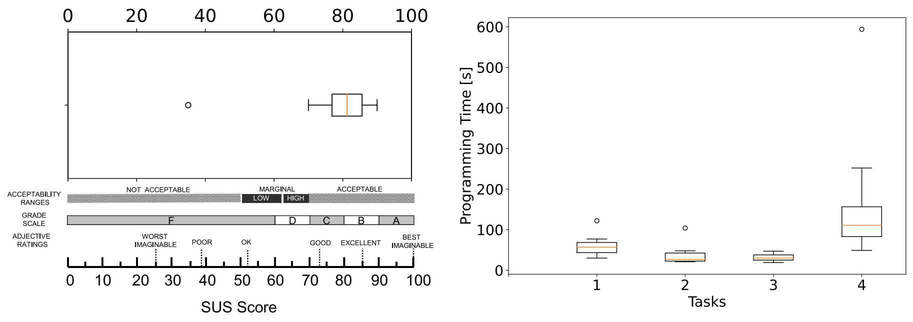
4 Evaluation

The following user study aims to answer the scientific question of the extent to which robots can be programmed to use gestures. To this end, the study consists of four tasks, each requiring different types of control structures, which are used to analyze the program's usability with the SUS score and the programming time. The first task involves programming a sequence of pick and place operations, while the second task requires a for loop to program a pick and place operation. Task three involves a for-each loop to move a set of objects using a pick and place operation. The fourth task requires programming a for-each loop that evaluates a condition within the loop body. We also set up a monitor for feedback about the hand skeleton model, the recognized gesture, and the filtered gesture. A suspected confounding factor is unreliable hand tracking. As the hand skeleton model is not displayed correctly on the monitor, it may affect the overall usability.

The user study commences with a written briefing for each participant and addresses any questions. Next, we give the participants a brief period to acquaint themselves with the robot programming system. Subsequently, we present individual tasks to the participants in paper form, which they can solve one after the other. Upon completing all tasks, the participants complete a SUS and a demographic questionnaire. The questionnaire covers information such as age, gender, highest degree attained, programming experience, experience with robots, and experience with gesture recognition systems.

The study involved twelve participants, all from convenience sampling, primarily students and employees of the University of Bayreuth. The age range of the participants was between 22 and 57 years, with a median age of 25 years, including eleven male and one female participants. Half of the participants had prior experience with handling robots, and 28.8% of all participants had previously worked with gesture recognition systems, all of whom had prior experience with robots. The analysis of the effects of previous knowledge on handling the programming system was not possible due to the small number of participants. However, this could be investigated in further research with a larger user group

The data shows that the median SUS score awarded is approximately 81.25, with an average SUS score of 77.7 (Fig. 3a). These values are higher than the average SUS values in the hardware (=71.8) and GUI (=76.2) categories and can be classified as good [29]. The outlier with a SUS Score of 35 is due to hand tracking. In particular, the hand skeleton model is not always detected or is lost during movement making programming the robot more complex and requiring additional effort. This was the case with the female subject, possibly due underrepresented training data in Media Pipe. However, in general, the user study evaluation by the SUS score demonstrates the convenience of programming complex tasks with gestures.



(a) Usability of the programming system (b) Programming time including failures

Fig. 3: The program's usability is 'good' [29], with a median SUS over 81.25 across all tasks. We suspect a familiarisation effect at the beginning reducing the programming time of Task 2 and 3. With increasing complexity in Task 4, however, the programming time increases due to multiple failed attempts.

When evaluating programming time, it is measured until a syntactically correct program is generated, including the times of failed programming attempts. As tasks get more complex, programming time increases, and more failed attempts occur before a correct program is achieved. Task four took twice as long as tasks two and three combined, with an average completion time of 111 seconds over two attempts and an outlier with 594 seconds due to hand tracking. Task one had a longer completion time of 57 seconds due to unfamiliarity with the programming system (Fig. 3b). The evaluation of the programming time in the user study showed that simple and complex tasks can be programmed quite quickly using gestures.

Most of the feedback from all participants relates to hand tracking. The hand skeleton model is not found during movements, which causes irritations. Additionally, participants suggested adding meta gestures that would influence the programming process to improve user-friendliness. For example, meta-gestures would allow users to undo gestures sent incorrectly rather than having to cancel programming and repeat programming the whole task. The study participants suggested including acoustic feedback to characterize the transmission of the gesture to the programming system, in addition to displaying text on the monitor. Another suggestion was to recognize natural behavior and segment it out.

5 Conclusions and Future Work

This work contributes a gesture-based robot programming system with good usability. In this case, gesture-based programming means the robot is programmed unimodally, and control structure-based with gestures. The programming system has proven to be good, as the median SUS score from the user study is above 71.4. For user-friendly programming with gestures, it is crucial to distinguish relevant gestures from natural behavior, which is achieved through gesture recognition and a sliding window filtering method. A context-free grammar is presented to check the syntactic correctness of a gesture input sequence and build up the context of the instructions using control structures. To make the program executable, the gestures in the control structures must be semantically interpreted. The programming system's usability was evaluated as part of a user study. The average SUS score was 77.7, which is considered good. The evaluation of programming time to generate a syntactically correct program showed a increase for more complex tasks. This was due to additional failed attempts caused by incorrect hand tracking during programming.

Therefore, our prototype needs further improvements. For example, quicker and more robust hand tracking could improve the system's usability. Additionally, reducing the filtering time of the gestures enhances user-friendliness and the programming time. Moreover, a customized feedback could improve the user's understanding and, thus, the overall usability. Also specifying the object position more precise would increase the number of industrial assembly tasks. In conclusion, this concept allows users to program a robot with good usability using only gestures. With this, even complex tasks can be programmed quickly.

References

1. Tsarouchi et al. (2016). High Level Robot Programming Using Body and Hand Gestures. Procedia CIRP, 55, 1–5.
2. Neto, P., Pires, J. N., Moreira, A. P. (2009). Accelerometer-based control of an industrial robotic arm. RO-MAN, 1192–1197.
3. Coronado, E. et al. (2017). Gesture-based robot control: Design challenges and evaluation with humans. 2017 ICRA, 2761–2767.
4. Shin, S., Tafreshi, R., Langari, R. (2018). EMG and IMU based real-time HCI using dynamic hand gestures for a multiple-DoF robot arm. Journal of Intelligent & Fuzzy Systems, 35(1), 861–876.
5. Mashood, A. et al. (2015). A gesture based kinect for quadrotor control. 2015 ICTRC, 298–301.
6. Yavşan, E., Uçar, A. (2016). Gesture imitation and recognition using Kinect sensor and extreme learning machines. Measurement, 94, 852–861.
7. Fang, B. et al. (2017). A novel data glove using inertial and magnetic sensors for motion capture and robotic arm-hand teleoperation. Ind. Rob.: An Intern. J., 155–165.
8. Sylari, A., Ferrer, B. R., Lastra, J. L. M. (2019). Hand Gesture-Based On-Line Programming of Industrial Robot Manipulators. INDIN, 1, 827–834.
9. Rudd, G., Daly, L., Cuckov, F. (2020). Intuitive gesture-based control system with collision avoidance for robotic manipulators. Ind. Rob.: The Intern. J., 243–251.
10. Devine, S., Rafferty, K.,Ferguson, S. (2016). Real time robotic arm control using hand gestures with multiple end effectors. UKACC (CONTROL), 1–5.
11. Peppoloni, L. et al. (2015). Immersive ROS-integrated framework for robot teleoperation. 3DUI, 177–178.
12. Quintero, C. P. et al. (2015). Visual pointing gestures for bi-directional human robot interaction in a pick-and-place task. RO-MAN, 349–354.
13. Becker, M. et al. (1999). GripSee: A Gesture-Controlled Robot for Object Perception and Manipulation. Autonomous Robots, 6(2), 203–221.

14. Han, J. et al. (2020). Structuring Human-Robot Interactions via Interaction Conventions. RO-MAN, 341–348.
15. Gäbert, C. et al. (2022). Gesture Based Symbiotic Robot Programming for Agile Production. CIVEMSA, 1–6.
16. Nuzzi, C. et al. (2021). MEGURU: A gesture-based robot program builder for Meta-Collaborative workstations. Rob. and Computer-Integrated Manufacturing.
17. Van Delden, S. et al. (2012). Pick-and-place application development using voice and visual commands. Industrial Robot: An International Journal, 39(6), 592–600.
18. Lambrecht, J., Krüger, J. (2012). Spatial programming for industrial robots based on gestures and Augmented Reality. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 466–472.
19. Lambrecht, J., Krüger, J. (2014). Spatial Programming for Industrial Robots: Efficient, Effective and User-Optimised through Natural Communication and Augmented Reality. Advanced Materials Research, 1018, 39–46.
20. Dietz, T. et al. (2012). Programming System for Efficient Use of Industrial Robots for Deburring in SME Environments. ROBOTIK; 7th German Conf. on Rob., 1–6.
21. Goel, R., Gupta, P. (2020). Robotics and Industry 4.0. In A. Nayyar A. Kumar (Eds.), A Roadmap to Industry 4.0: Smart Production, Sharp Business and Sustainable Development (pp. 157–169). Springer International Publishing.
22. Weigelt, S. (2022). Eine agentenbasierte Architektur für Programmierung mit gesprochener Sprache.
23. Sucker, S. and Henrich, D. (2023). A layered Pipeline for Natural Language Robot Programming with Control Structures. Annals of Scientific Society for Assembly, Handling and Industrial Robotics (to appear)
24. Sauer, L., Henrich, D. (2022). Synchronisation in Extended Robot State Automata. IRC, 360–363.
25. Riedl, M. (2022). Intuitive sensorbasierte, editierbare, kinästhetische Programmierung von Pick-and-Place-Aufgaben für Mehrrobotersysteme [Dissertation].
26. Xia, Z.et al. (2019). Vision-Based Hand Gesture Recognition for Human-Robot Collaboration: A Survey. ICCAR, 198–205.
27. Martin, A. (2014). Exploring 2D and 3D gesture interaction with an assistive system for a workplace for impaired workers [Diploma Thesis].
28. Gesture. Oxford Reference. Retrieved 19 Mar. 2024, from https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095850220.
29. Bangor, A., Kortum, P., Miller, J. T. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. Journal of Usability Studies Archive.
30. Holz, D. et al. (2015). Real-time object detection, localization and verification for fast robotic depalletizing. IROS, 1459–1466.
31. Farag, M. et al. (2019). Real-Time Robotic Grasping and Localization Using Deep Learning-Based Object Detection Technique. I2CACIS
32. Shi, Y. et al. (2021). Review of dynamic gesture recognition. Virtual Reality & Intelligent Hardware, 3(3), 183–206.
33. Jain, R., Karsh, R. K., Barbhuiya, A. A. (2022). Literature review of vision-based dynamic gesture recognition using deep learning techniques. Concurrency and Computation: Practice and Experience, 34(22), e7159.
34. Mueller, F. et al. (2018). GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB. IEEE/CVF Conf. on Comp. Vis. and Pattern Recognition, 49–59.
35. Zhang, F. et al. (2020). MediaPipe Hands: On-device Real-time Hand Tracking (arXiv:2006.10214). arXiv.
36. Chen, W. et al. (2020). A Survey on Hand Pose Estimation with Wearable Sensors and Computer-Vision-Based Methods. Sensors, 20(4), Article 4.