machines

MDPI

Article

Novel Gesture-Based Robot Programming Approach with the Ability of Code Reuse

Vladyslav Andrusyshyn [1,2], Kamil Židek [1], Vitalii Ivanov [1,2,*] and Ján Piteľ [1]

[1] Faculty of Manufacturing Technologies with a seat in Presov, Technical University of Kosice, 1, Bayerova St., 080 01 Presov, Slovakia; vladyslav.andrusyshyn@tuke.sk (V.A.); kamil.zidek@tuke.sk (K.Ž.); jan.pitel@tuke.sk (J.P.)

[2] Department of Manufacturing Engineering, Machines and Tools, Faculty of Technical Systems and Energy Efficient Technologies, Sumy State University, 116, Kharkivska St., 40007 Sumy, Ukraine

* Correspondence: ivanov@tmvi.sumdu.edu.ua

Abstract: Nowadays, there is a worldwide demand to create new, simpler, and more intuitive methods for the manual programming of industrial robots. Gestures can allow the operator to interact with the robot more simply and naturally, as gestures are used in everyday life. The authors have developed and tested a gesture-based robot programming approach for part-handling applications. Compared to classic manual programming methods using jogging and lead-through, the gesture control method reduced wasted time by up to 70% and reduced the probability of operator error. In addition, the proposed method compares favorably with similar works in that the proposed approach allows one to write programs in the native programming language of the robot's controller and allows the operator to control the gripper of an industrial robot.

Keywords: teleoperation; assembly; gesture recognition; production line; collaborative robotics; industrial growth; process innovation

Citation: Andrusyshyn, V.; Židek, K.; Ivanov, V.; Piteľ, J. Novel Gesture-Based Robot Programming Approach with the Ability of Code Reuse. *Machines* **2024**, *12*, 217. https://doi.org/10.3390/machines12040217

Academic Editor: Dan Zhang

Received: 26 December 2023
Revised: 21 March 2024
Accepted: 21 March 2024
Published: 25 March 2024

1. Introduction

Currently, factories need to optimize costs, reduce production time, and increase production flexibility due to market instability, growing demands of customers, ecology requirements, competition, etc. Implementing Industry 4.0 concepts into manufacturing can solve today's manufacturing challenges [1].

Although the first mention of the Industry 4.0 concept appeared in 2011, at the moment, the reference implementation or application of the concept has not yet been implemented [2]. This fact means there is still a demand for research in the field of Industry 4.0, mainly to simplify its integration into modern industries.

The use of industrial robots as a means of industrial automation is a popular and actively used solution for increasing production efficiency. Over the past decade, industrial robots have become essential factory automation components due to their performance and versatility [2]. Robotics is also one of the most researched technologies within the framework of the Industry 4.0 concept [3]. Industrial robots can perform a wide range of tasks in production: parts handling, welding, painting, assembly, dispensing, processing, etc. [4]. Industrial robots are highly repeatable and can perform monotonous, heavy, or dangerous tasks for long periods with high accuracy.

Nowadays, advances in the field of smart sensors, advanced materials, information processing algorithms, and increasing computer computing power have made it possible to develop and introduce a new generation of industrial robots—collaborative robots. Classic industrial robots can effectively perform only a limited list of tasks, do not have cognitive functions, and require readjustment when changing production conditions. Consequently, factories cannot abandon manual labor despite its low efficiency compared to industrial robots. A human has flexibility unreachable for a robot, can make decisions independently,

and can perform new production tasks without instructions, using his previous experience with similar tasks. Also, classic industrial robots work separately from humans due to safety issues. The introduction of collaborative robots has dramatically changed production. Humans can now collaborate safely with robots, combining their benefits while increasing productivity and flexibility and creating more operator-friendly workplaces [2].

However, there are still challenges that have not yet been resolved in robotics. Despite improvements in the design of industrial robots, robot programming methods have not changed significantly for more than 40 years [5]. The creation of control programs for industrial robots requires highly skilled engineers and specialized software if it is necessary to create large programs or programs with complex trajectories. It is especially critical for small- and medium-sized enterprises that cannot afford to have a separate department that could serve industrial robots. Moreover, due to single-item or small-batch production, classical programming methods are unsuitable for small- and medium-sized enterprises. There is a huge potential for industrial robots to be applied in small- and medium-sized enterprises, but this has not been realized [6]. At the same time, small- and medium-sized enterprises are essential for the European economy, as they significantly contribute to the economy and innovation [7]. Small- and medium-sized enterprises have noted that the ease of programming and operating an industrial robot is one of the weighty criteria when buying collaborative robots [8].

This fact encourages scientists to work with robot manufacturers to create more intuitive and simple methods for programming industrial robots. Instead of using the teach pendant as an input channel for informing an industrial robot about human commands, researchers suggest using more natural methods, such as recognition of objects, gaze, gestures, voice commands, and emotions [9]. Implementing such systems is possible through solutions related to collaborative robots, in which similar information input channels are used to increase safety. Gaze, object, and emotion recognition are mainly used to predict human intentions and optimize robot performance. Gestures and voice commands are used to transfer commands to the robot, from which the program will eventually be formed. The use of gestures is the most promising method of transmitting commands to the industrial robot since voice control is not effective and reliable enough in a production environment due to ambient noise.

In turn, gesture recognition can be performed using contact and non-contact systems. Contact systems such as wearable markers or sensors are highly accurate, have low latency, are less prone to occlusion, and may contain tactile feedback devices. However, wearable devices hinder human movement and may require individual adjustment for a person or group of people. Non-contact gesture recognition systems are considered more promising since they do not hinder human movements, and accuracy, speed, and resistance to occlusions are constantly increasing due to the introduction of new machine-learning algorithms. Instead of physical feedback, it is possible to use visual or voice feedback. Also, non-contact solutions are usually cheaper than are contact ones.

The authors discovered that works related to new simpler programming methods do not consider the interaction between the robot controller and the developed software and hardware complex for simplified programming and do not describe how the program is saved and can be reproduced. In addition, related works do not consider the setting of the application, e.g., the effect of singularity on the programming process is not described. However, the singularity phenomenon will directly affect the application's performance depending on the robot's initial position.

Based on the above, this article is dedicated to creating a more intuitive method for programming industrial robots using gestures recognized by non-contact methods. In addition, the authors focus particularly on code reuse. Code reuse improves the efficiency and ease of programming industrial robots [10]. The authors of [11] also noted that in the creation of a new flexible approach to control a robotic cell, code reuse simplifies and speeds up the process of programming robot controllers.

The article is organized as follows. The Section 2 section critically reviews articles devoted to designing intuitive methods to interact with a robot using gestures. Particular attention was paid to analyzing software and hardware solutions for tracking human hand movements and recognizing gestures. The Section 3 section presents hardware and software solutions for testing the proposed approach and describes its operating principle. The Section 4 section provides the results of an experimental comparison of the proposed approach with classical approaches for the manual creation of programs for industrial robots. Finally, in the Section 5 section, the proposed approach is compared with other works in gesture-based manual program creation, the study's findings are presented, and plans for future research directions are summarized.

2. Literature Review

Due to the advantages and capabilities of camera-based gesture recognition systems, they are the most popular choice when developing intuitive applications for human–robot interaction. However, to better understand the current state of research, it is necessary to analyze the articles on intuitive human–robot interaction and highlight the tools currently used for gesture recognition.

Some authors [12] presented a collaborative robotic cell where the interaction between a human and a robot occurs with the help of pointing gestures. The presented collaborative cell consists of a sensory part (which includes the Leap Motion Controller camera for tracking the position of the operator's hands), visual feedback, and the Universal Robotics UR5e industrial collaborative robot. Using gestures as input methods was motivated by the desire to improve productivity and increase worker satisfaction during human–robot interaction. Other authors [13] used the Orbec Astra S camera for developing a teleoperation application for the ABB YuMi robot. In the proposed application, frames from the camera are processed by the AlphaPose neural network to determine the key points of the skeleton and hands. Using AlphaPose allows for expanded applications for teleoperation with several people since this solution can segment several people in the picture. The work is also notable because it pays considerable attention to the review of applications for determining the key points of the human skeleton based on neural networks and datasets for their training and testing. Other authors [14] proposed a novel application for teaching a robot by demonstration. The iPad's front camera frames were processed by Google MediaPipe solution for image segmentation (highlighting the area around the hand). Next, the obtained data were processed by the FrankMocap solution, which provides the position of the finger joints in the axis-angle format. Further, the Perspective-n-Point algorithm provides the coordinates of the key points of the hands. The developed application was tested on the UFactory XArm-6 robot with a dexterous hand installed. Other authors [15] used a DAVIS 240 C event camera to track human hand movements. Compared to classical machine vision systems, the event camera has lower latency, no blurring in dynamic conditions, and is less demanding on processing power when processing images. The proposed algorithm of the region of interest due to the removal of noise made it possible to reduce the data size by 89% without reducing the accuracy of determining the hand's position. The developed algorithm's accuracy for determining the hand's position was compared. The wearable magnetic tracking sensor Polhemus Liberty was used as a reference. As a result, the error in determining the hand's position was about 28 mm in three-dimensional space.

Nevertheless, in addition to the robot's understanding of the operator's commands, the person must understand the robot's intentions, which are realized through feedback. In order not to constrain the movement of an operator but at the same time to reliably inform an operator, visual feedback is used in most cases. Augmented and virtual reality are used as feedback during human–robot interaction since headsets mainly have solutions for determining the position of key points of hands and hand gestures or for simplifying the integration of solutions from third-party manufacturers. For example, in [16], the authors proposed a method for programming an industrial robot by demonstrating using

the Microsoft HoloLens 2 augmented reality headset. The solution is based on ROS and has a modular structure. The solution was successfully tested on the Universal Robots UR5 and ABB IRB 2600 robots, during which simple programs were created and reproduced. Other authors [17] proposed a method for the remote control of an industrial robot using Microsoft HoloLens augmented reality glasses. This work's main focus was creating a semi-automatic method for combining the base coordinate system of a real and virtual industrial robot in augmented reality. The solution was successfully tested on the KUKA KR-5 industrial robot. However, it is worth noting that the authors encountered problems tracking the hand on a black background, probably due to absorption properties.

It is also crucial to mention a disadvantage of non-contact systems for determining the position of key points of the human hand based on cameras. Camera-based solutions are subject to the phenomenon of self-occlusion. Self-occlusion or partial or complete overlap of the hands, fingers, or palms decreases the accuracy in determining key points or makes the recognition process impossible.

Different researchers have attempted to solve the problem of occlusions in different ways. The most common way to avoid occlusions is to use multi-camera systems. For example, the authors of [18] used 3 Intel RealSense D435 cameras to track the position of hands and recognize gestures when creating an intuitive approach to human–robot interaction. The Google MediaPipe solution was used for the software component. As a result, the developed gesture control application made it possible to speed up the creation of a program by two times compared to manual control and four times compared to programming using the teach pendant. Other authors [19] developed an inexpensive remote control system based on 4 Intel RealSense D415 cameras for the KUKA LBR iiwa 7 R800 robot with a Wonik Robotics Allegro gripper, which has 23 degrees of freedom. The developed application made it possible to perform highly complex manipulation tasks, such as opening a box, changing the position of a part while grasping it, extracting money from a wallet, etc.

However, there are also quite non-standard solutions. For example, to solve occlusion problems, in [20], the authors used an active machine vision system to capture a human hand from the most optimal angle and distance. The active machine vision system consisted of the Universal Robotics UR5 collaborative robot, on which the Intel RealSense SR300 camera was installed. For a preliminary assessment of the position of the human hand, the PhaseSpace motion tracking system and light LED wrist markers were used. The remote robotic cell consisted of a Willow Garage PR2 robot, a Shadow Inc. gripper with 19 degrees of freedom, a Microsoft Kinect V2 camera, and a webcam that captures the robot's performance. The authors were motivated to create this system because multi-camera systems do not entirely solve the problem of occlusions, require frame synchronization, and require more time to process frames.

It is also worth mentioning that delays occur even when using a single camera since the system needs time to transmit and process information from the camera. Researchers are trying to solve this problem by trying to predict human behavior. Also, this approach makes human–robot interaction more predictable and intuitive. Some authors [21] also used a long short-term memory neural network to inform the robot about human intentions. Information about the position of the key points of the operator's hand and the direction of his gaze was used as input data. The Google MediaPipe solution was used as the software for determining the position of the hands, and the Nexigo N930AF RGB camera was used as the hardware. In addition, object recognition was implemented using the RGB camera and the YOLO v5 library. Information about the direction of the operator's gaze was obtained using the Pupil Core platform. Experiments showed that using all three information streams (hand, gaze, object) showed better results in recognizing intentions than did hand–object and hand–gaze.

In addition, prediction can improve the performance of the robotic cell. For example, in [22], the authors used a long short-term memory neural network to predict the progress of assembly operations—screwing manually or with the help of tools and hammering.

A convolutional neural network processed readings from a Microsoft Kinect V1 camera and inertial measurement units from Shimmer Inc. to recognize assembly operations. The developed application was tested on a robotic cell, where the robot had its tasks in addition to collaborative ones. With planning, the system minimized downtime by switching between tasks. Despite inaccuracies in the process of recognition of actions, the success of delivery of the next part in the collaborative assembly process was more than 80%. In [23], the authors used the Tobii Eye Tracker 4C gaze tracker and the Leap Motion Controller camera to create a collaborative assembly application. The use of the gaze tracker was driven by the desire to expand the list of commands for interacting with the robot while not overloading the operator with a large number of hand gestures. The experiment showed that the developed approach reduced the total task execution time by 57.6% compared to classical programming through the teach pendant.

The hardware and software solutions used for human position tracking are summarized in Table 1.

Table 1. Hardware and software solutions used in the reviewed scientific papers.

Robot	Pose Recognition Method	Sensor for Pose Recognition	Resistance to Occlusions	Body Parts for which Position Is Determined	Source
Universal Robots UR5e	Leap Motion SDK	Leap Motion	–	Palm	[12]
ABB YuMi	AlphaPose and Halpe libraries	Orbbec Astra S	–	Skeleton, palm	[13]
UFactory XArm-6 with Allegro gripper	MediaPipe and FrankMocap libraries	iPad or iPhone camera	–	Palm	[14]
–	Custom region of interest algorithm	DAVIS 240 C event camera	–	Palm	[15]
Universal Robots UR5, ABB IRB 2600	Microsoft HoloLens 2 SDK	Microsoft HoloLens 2 sensors	–	Palm	[16]
KUKA KR-5	Microsoft HoloLens 2 SDK	Microsoft HoloLens 2 sensors	–	Palm	[17]
Universal Robots UR3e	MediaPipe library	Intel RealSense D435 cameras (3 pcs)	+	Palm	[18]
KUKA LBR iiwa7 R800 with Allegro gripper	Dense articulated real-time tracking framework	Intel RealSense D435 cameras (4 pcs)	+	Palm	[19]
Willow Garage PR2 with Shadow Inc. gripper	End-to-end hand pose regression network	Intel RealSense SR300 camera installed on a Universal Robot UR5, PhaseSpace tracking system	+	Palm	[20]
–	Mediapipe, Pupil Core	Nexigo N930AF RGB camera	–	Palm, gaze	[21]
Universal Robots UR3	Convolutional neural network	Shimmer Inc. inertial measurement units, Microsoft Kinect V1 camera	–	Skeleton, palm	[22]
Universal Robot UR5, simulation	Convolutional neural network	Leap Motion and Tobii Eye Tracker 4C	–	Palm, gaze	[23]

3. Materials and Methods

3.1. Hardware Setup

The proposed approach was tested using the SmartTechLab laboratory's equipment (Faculty of Manufacturing Technologies with a seat in Prešov, Technical University of Košice). The proposed approach was tested using the ABB YuMi dual-arm collaborative robot, each arm of which has seven axes for increased flexibility. The robot's controller used in the laboratory had the 689-1 externally guided motion option installed to transmit the robot's position at a frequency of 250 Hz. ABB SmartGrippers were installed on each robot arm.

The performance and accuracy of the application depend on the optimal choice of the camera for hand tracking. The Leap Motion Controller camera was chosen (Figure 1) since this camera was explicitly created to determine the coordinates of key points of the hands accurately and does not require initial training. High frame rates of up to 120 fps allow reliable tracking of hand movements and gesture recognition with high frequency. This camera also has a relatively affordable price. The literature review in [12] shows that the Leap Motion Controller is a popular choice as a hand-tracking device in teleoperation applications. As for the disadvantages, the Leap Motion Controller is very sensitive to the position of the human hands relative to the camera. The developers of Leap Motion Controller suggest only three tracking modes: desktop, screentop, and head-mounted. Also, the Leap Motion Controller solution does not have a built-in solution for synchronizing multiple cameras, which does not allow the avoidance of self-occlusion of fingers and expanding the limited workspace. However, the authors of [24] proposed an approach to synchronizing the readings of two Leap Motion Controller cameras, which made it possible to avoid the self-occlusion of the fingers during robot teleoperation using gestures. Additionally, the authors proposed a process for automating camera calibration, simplifying the workplace setup. However, while testing the cameras, the authors noticed that interference in a multi-camera system only occurs when using the first-generation Leap Motion Controller (Firmware version 1.7.0). When testing a multi-camera system based on the second-generation Leap Motion Controller (Firmware version 3.8.6), no interference is observed, and the cameras reliably read the position of the hands, which theoretically can expand the workspace of the cameras and make the application resistant to occlusions. However, the application was based on a single camera at this stage for simplicity.

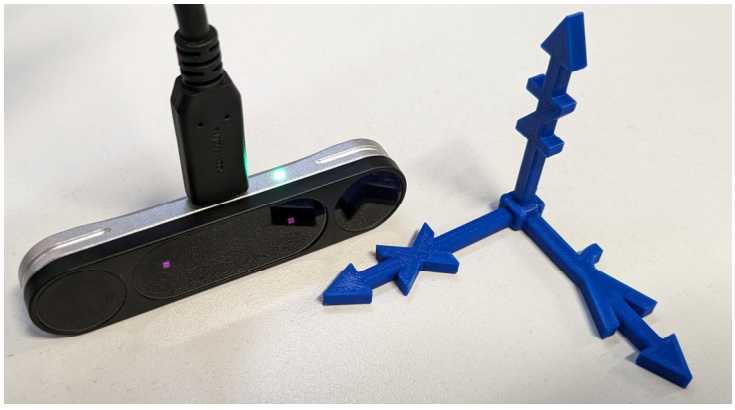

Figure 1. Leap Motion Controller (second generation).

According to the manufacturers, the Leap Motion Controller has viewing angles of 140° × 120°. The recommended working area is an X axis from −117.5 to 117.5 mm, a Y axis from 82.5 to 317.5 mm, and a Z axis from −73.5 to 73.5 mm [25]. One study [26] compared the declared performance with the actual one. Based on real-world testing, the authors recommended using the Leap Motion Controller camera in the following ranges: X and Z axes from −200 to 200 mm and an Y axis from 50 to 600 mm. However, it is worth

noting that the studies were carried out with one hand size (the size of an adult man 185 cm tall), and only static positions were tested.

Regarding accuracy issues, the authors of [27] experimentally analyzed the accuracy and stability of hand tracking with a Leap Motion Controller camera under dynamic conditions. Within the working space chosen by the authors (X and Z axes from -250 to 250 mm, Y axis from 0 to 400 mm), the sensor showed a less than 5 mm hand detection deviation and less than 10 mm outside the working space. The authors also noted an increase in deviation at a hand distance of more than 200 mm in the Y axis from the camera to a value of less than 10 mm. Nevertheless, these accuracy characteristics are acceptable to confirm the concept stated in this article.

For the laboratory conditions of our study, the following working area of the camera was experimentally selected: axis X for the left hand from -180 to -30 mm and for the right hand from 30 to 180 mm, axis Z from -40 to 80 mm, and axis Y from 180 to 260 mm.

In the proposed approach, a Lenovo Legion Pro 5 16IRX8 laptop was responsible for processing data from the camera and for exchanging information with the robot and had the following parameters: Intel Core i7-13700HX processor, 16 GB DDR5 RAM, Nvidia GeForce RTX 4060 8 GB graphics card, and M.2 PCIe SSD 1000 GB.

3.2. Software Setup

3.2.1. Robot Configuration

The StateMachine Add-In was used to simplify interaction with the robot. After the add-in was installed, modular RAPID programs and configuration files were loaded onto the controller based on system characteristics, such as the number of robots and available options. Modular RAPID programs were implemented according to the finite state machine model. This add-in provides an interface for controlling ABB SmartGripper grippers, managing the connection to the robot via the externally guided motion protocol, and executing custom RAPID modules.

The structure of the RAPID task was as follows (Figure 2):

- The TRobEGM program module provides an interface for starting and stopping robot position control via the 689-1 externally guided motion option. With the externally guided motion option, it is possible to transmit the desired position of the tool center point at high frequency, which minimizes delays, thereby improving the user experience of human–robot interaction. In addition, this module provides an interface to set externally guided motion parameters, such as the condition time, connection timeout, maximum speed deviation, the tool and work object coordinate systems, etc.
- The TRobMain program module initializes submodules and manages the state machine.
- The TRobRAPID program module provides an interface for interacting with the system and user RAPID routines.
- The TRobSG program module provides an interface for interacting with ABB SmartGrippers via RobotWeb Services. These include requesting gripper initialization, calibration, gripper finger movement, setting gripper finger speed and gripping force, and controlling suction cups on the gripper if available.
- The TRobUtility program module contains some system functions and variables other modules use.
- The BASE system module stores predefined system data, such as a description of the characteristics of the tool, work object, and loads attached to the robot's mechanical interface.
- The TRobSystem system module allows users to add a system-specific initialization procedure and run custom RAPID procedures.

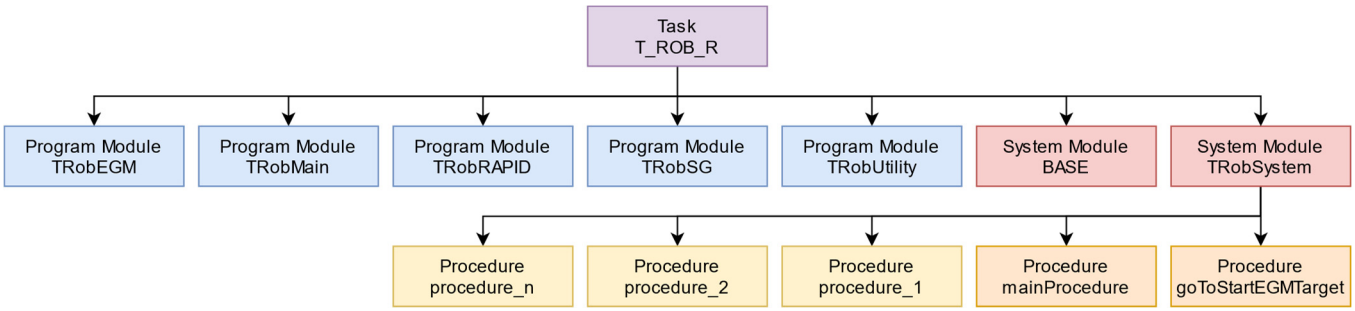

Figure 2. The structure of the RAPID task.

It was necessary to add the following custom RAPID procedures to the TRobSystem system module for the developed application: mainProcedure and goToStartEGMTarget. mainProcedure is the main procedure, which includes a set of supplemental procedures that are called sequentially. Dividing a program into procedures is a good practice and makes the code easier to develop, maintain, and understand. This practice improves code readability and understanding, allows code reuse, and simplifies program debugging, scaling, and maintainability. In mainProcedure, which contains a sequence of procedures, it is possible to add procedures that are in the scope of the current task or the TRobSystem module.

The goToStartEGMTarget procedure is responsible for moving the central tool point to the starting point of the working object area in joint mode. Since a 7-axis robot was used for the experimental validation of the approach, there could be different joint positions for the same tool center point position. Also, a larger number of axes allows singularity areas to be more effectively avoided when programming with simulation software (ABB RobotStudio, FANUC Roboguide, RoboDK, etc.) but also increases the number of singularity areas. Since, for ease of implementation of the application, the robot is controlled by sending the position of the tool center point in the Cartesian coordinate system, depending on the position of the robot joints before the start of the application, there may be cases of moving outside the range of one joint axis of the robot or of the robot entering the singularity area, which can lead to an emergency stop of the application. To avoid this, when transferring the position of the tool center point in a Cartesian coordinate system, it is recommended to experimentally select the optimal configuration of the joint at the starting point of the working object area. The optimal configuration of robot joints can be selected by solving forward and inverse kinematics equations. Some authors have provided equations for the forward and inverse kinematics of the ABB YuMi robot, although the provided technique is universal and suitable for any articulated robot. A general description of forward and inverse kinematics is presented in [28]. Also, others authors [29] provided Denavit–Hartenberg parameters (DH parameters) only for the left arm of the ABB YuMi robot, which are necessary for solving kinematic equations. Also, they did not consider the robot arm's location relative to the robot's base coordinate system and the joints' zero position. DH parameters for the right hand, with consideration to the robot's base coordinate system and the joints' zero position, are provided in Table 2. DH parameters were obtained according to the manufacturer's 3D CAD model of the ABB YuMi robot [30].

Another option is to use the ABB RobotStudio, which provides forward and inverse kinematics solutions. ABB RobotStudio provides a list of joint configurations for a given tool center point (Figure 3). ABB RobotStudio contains information about the kinematics of ABB robots; however, it is possible to solve forward and inverse kinematics equations for third-party industrial robots. Creating a library file containing information about kinematics, such as Denavit–Hartenberg parameters, joint limits, etc., is necessary. In addition, the Robot Operating System (ROS) also provides solvers for forward and inverse kinematics problems.

Table 2. Denavit–Hartenberg parameters for the right hand.

i	a_i, mm	α_i, Degree	d_i, mm	θ_i, Degree
0	46.0946	−61.9756	373.6889	−112.7959
1	30	270	187.9456	$-37.6 + \theta_1$
2	30	270	0	$180 - \theta_2$
3	40.5	90	251.5	θ_7
4	40.5	90	0	$90 - \theta_3$
5	27	90	265	$180 + \theta_4$
6	27	90	0	$180 - \theta_5$
7	0	0	36	$180 + \theta_6$

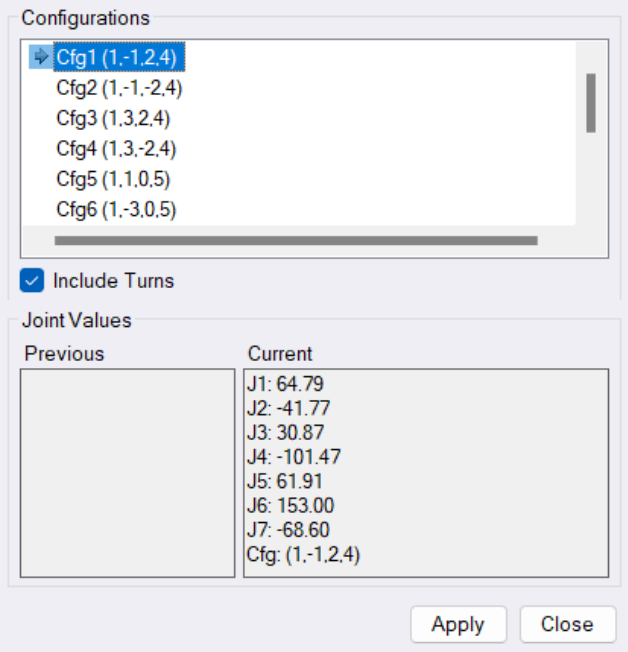

Figure 3. Example of a list of joint configurations obtained in the ABB RobotStudio.

In choosing the optimal configuration of joints at the initial point, avoiding positions of joints close to the limits and singularity points is recommended. At singularity points, the robot loses one degree of freedom, and in positions close to the singularity point, movement becomes impossible or unpredictable. Often, when the end point of the tool passes near the singularity point, the joints begin to move at high speed while the speed of the tool center point slows down. When controlling the ABB industrial robot using the Externally Guided Motion option, passing the tool center point near the singularity area causes the application to stop. The topic of singularity is complex and depends on the robot's geometric configuration, the chosen joint configuration, and required speed of the tool center point.

The TRobSystem system module can also store custom procedures that can be generated by the developed application and which can later be named in the mainModule module, for example, "procedure_1", "procedure_2", "procedure_n", etc.

3.2.2. Desktop Application Implementation

A desktop application was developed to exchange and process data between the industrial robot, camera, and user. The program was written in C++ and tested on the Windows operating system, but since it used cross-platform libraries, it should work on

Linux without the source code being changed. The Qt library was used to build the graphical user interface [31]. In addition, the Network and WebSockets modules of the Qt library were used to write an interface to interact with the robot's HTTP server through a set of protocols and standards called RobotWebServices, which is designed to control and monitor ABB industrial robots. The C++ library abb_libegm was used to interact with the externally guided motion client of the robot controller [32]. A conceptual sketch of the interaction between an external computer and an ABB robot controller is shown in Figure 4.

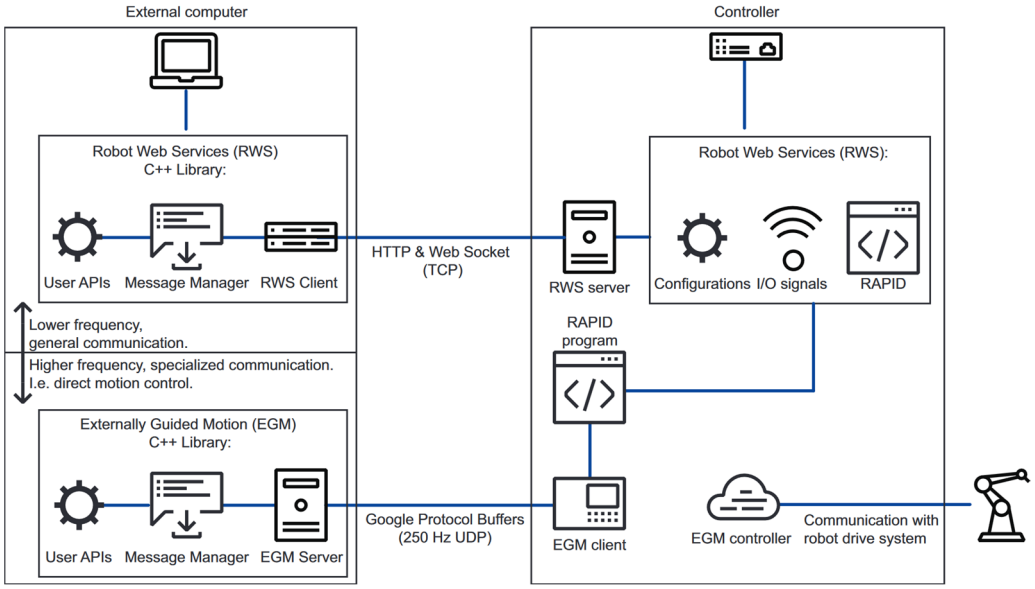

Figure 4. Interaction between an external computer and an ABB robot controller [33].

The LeapC SDK interacts with the Leap Motion Controller camera [34]. The LeapC SDK provides service information about the status of receiving and processing camera events and information about the position of key points of the hands, including the position of the central point of the palm, and in addition, provides implementation of the recognition of simple gestures such as pinch and grab. An example of performing the pinch and grab gestures is shown in Figures 5 and 6, respectively. In the developed application, the right hand is used to move the robot's arm, and the pinch gesture of the right hand is used to control the gripper installed on the robot. The left hand is responsible for recording the state of the robot and the grip when creating a new procedure using the grab gesture. The library also provides information about confidence in recognizing hand positions and gestures, which was used to improve the reliability of the application.

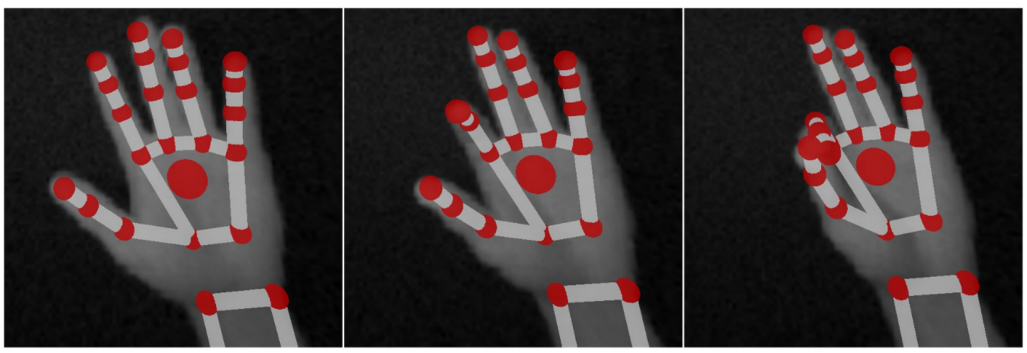

Figure 5. The pinch gesture.

Figure 6. The grab gesture.

By default, the location of hand key points relative to the camera's coordinate system and the location of the robot's tool center point in its global coordinate system are not connected. In addition, the robot and camera have their own working area. The working area of the camera is limited by the values (in mm) x_{lmc_min} and x_{lmc_max}, and the robot's working area is limited by the values (in mm) x_{tcp_min} and x_{tcp_max}. The coordinates of the position of the hands key points x_{lmc}, mm, in the camera coordinate system can be converted into the coordinates of the tool center point x_{tcp}, mm, using Equation (1):

$$x_{tcp} = C_{offs} + C_r \times x_{lmc}. \quad (1)$$

Here, C_{offs}, mm, is the offset of the camera and robot workspaces, and C_r is the ratio of the workspaces. These values can be calculated according to Equations (2) and (3), respectively:

$$C_{offs} = (x_{tcp\_max} - x_{tcp\_min})/(x_{lmc\_max} - x_{lmc\_min}), \quad (2)$$

$$C_r = x_{tcp\_min} - x_{lmc\_min} \times C_{offs}. \quad (3)$$

The left side of the main application window (Figure 7) contains a block of connection parameters, with fields for entering the IP address, port, login, and password of the HTTP robot server, which is responsible for RobotWebServices, as well as the port for connecting to the client externally guided motion of the robot controller. Before the application is launched for the first time, the connection settings to the externally guided motion client on the controller must be checked. In "Configuration", "Communication", and "Transmission Protocol", the StateMachine Add-In automatically creates the connection parameters for remote control of the mechanical unit groups. However, the system administrator must ensure that the IP address specified in the UdpUc protocol connection parameters matches the IP address of the remote computer. On the right side of the application, there is a drop-down list in which the user can select the task with which he will work and the contents of the mainProcedure procedure of the selected task. The procedure list widget allows the user to change the sequence of procedures in the list by simply dragging and dropping. When the user right-clicks on the list of procedures, a context menu will appear, with which the user can add a new procedure to the list, add a previously created procedure to the list, or remove the selected procedure from the list.

When the user selects the "Add a new procedure" action from the context menu, a new window opens (Figure 8). In the left part of the "Add a new procedure" window, the user can see the status of the application services: the RobotWebServices HTTP client, the externally guided motion service, and the Leap Motion camera service. For the normal functioning of recognition of hand positions and gestures, the person's hands must be in the optimal working area of the camera. The application will signal the person when his hands move outside the optimal working area of the camera. For this purpose, "Adding points" and "Right arm in working space" indicators have been added to the services status widget.

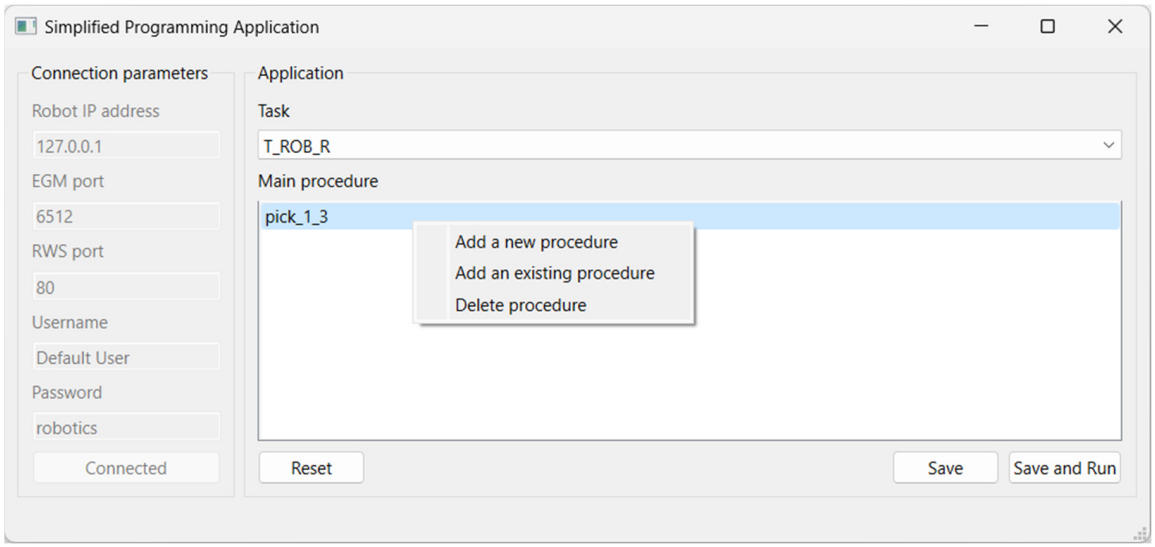

Figure 7. The main desktop application window.

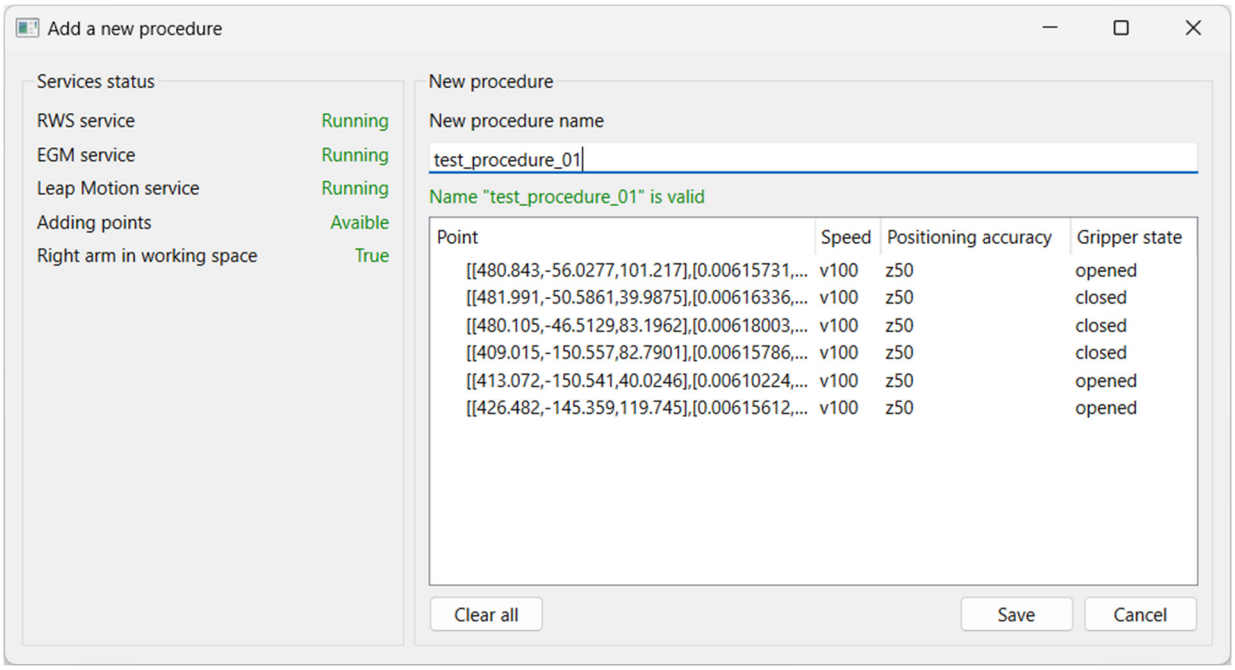

Figure 8. The "Add a new procedure" window.

The "Adding points" indicator informs the user whether the following conditions are met: whether the camera recognizes the position of the left and right hands simultaneously and whether the left hand is in the optimal working area of the camera. If the "Adding points" indicator has the status "Available", the user can record the robot's position and gripper state in the procedure using the grab gesture of the left hand. The "Right arm in working space" item informs the user whether the person's right hand is in the optimal working area of the camera, which is responsible for controlling the position of the robot and the state of the robot's grip (open or closed). Going outside the optimal working area of the camera results in a significant reduction in the accuracy of gesture recognition, which can lead to either unintentional recording of the robot's position and gripper state or unintentional opening or closing of the gripper (which in early testing of the application led to collisions between the gripper and the part where the operator tried to pick the part with the robot).

On the right side of the "Add a new procedure" window, there is a field for entering the name of the new procedure and a list of robot position points with the state of the gripper that will be written in the new procedure. After writing the procedure and entering the correct name of the new procedure, the user can save it in the native programming language of the robot controller (ABB RAPID) by clicking the "Save" button. The user can also delete the created point using the context menu, which is called by right-clicking the mouse, completely clearing the list of points using the "Clear all" button, and closing the window to cancel the creation of a new procedure.

Example of RAPID code generated by the program:

```
    PROC test_procedure_01()
    g_GripOut;
    MoveL [[480.843,-56.0277,101.217],[0.00615731,-0.707066,-0.707094,0.00623345],[1,
-1,2,4],[162.399,9E+09,9E+09,9E+09,9E+09,9E+09]], v100, z50, Servo;
    MoveL [[481.991,-50.5861,39.9875],[0.00616336,-0.707082,-0.707077,0.00619356],[1
-1,2,4],[166.821,9E+09,9E+09,9E+09,9E+09,9E+09]], v100, z50, Servo;
    WaitRob \InPos;
    g_GripIn;
    MoveL [[480.105,-46.5129,83.1962],[0.00618003,-0.70707,-0.707089,0.00619892],[1,
-1,2,4],[164.041,9E+09,9E+09,9E+09,9E+09,9E+09]], v100, z50, Servo;
    MoveL [[409.015,-150.557,82.7901],[0.00615786,-0.707057,-0.707103,0.00613658],[1,
-1,2,4],[154.645,9E+09,9E+09,9E+09,9E+09,9E+09]], v100, z50, Servo;
    MoveL [[413.072,-150.541,40.0246],[0.00610224,-0.707113,-0.707048,0.00612799],[1,
-1,2,4],[159.223,9E+09,9E+09,9E+09,9E+09,9E+09]], v100, z50, Servo;
    WaitRob \InPos;
    g_GripOut;
    MoveL [[426.482,-145.359,119.745],[0.00615612,-0.707063,-0.707097,0.00616934],[1,
-1,2,4],[151.882,9E+09,9E+09,9E+09,9E+09,9E+09]], v100, z50, Servo;
ENDPROC
```

When the user selects the "Add an existing procedure" action from the context menu of the main window, a dialog box opens (Figure 9) in which, using a drop-down list, the user can select a previously created procedure that is not a StateMachine Add-In system procedure, and which is either in the global scope or which is in the TRobSystem task, and add it to mainProcedure procedure.

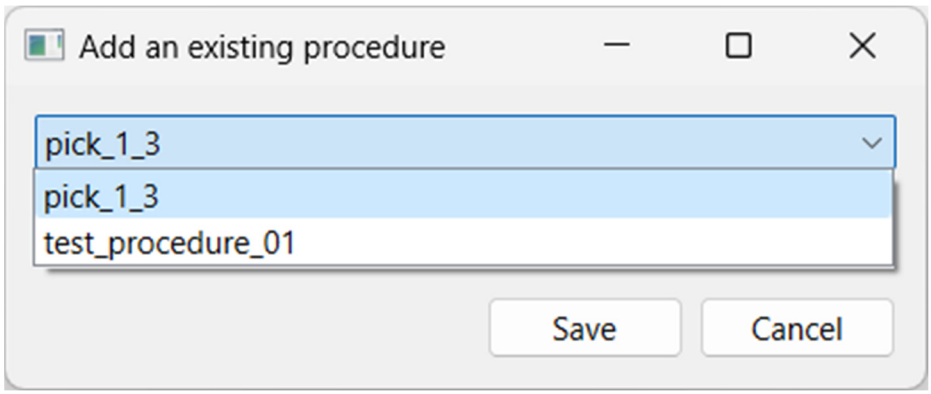

Figure 9. The "Add an existing procedure" window.

After forming the desired sequence of supplemental procedures in the mainProcedure procedure, the user can request that it be saved on the controller and saved and run. In addition, the user can cancel all changes made to the mainProcedure procedure by clicking the "Reset" button.

A simplified diagram of the entire developed application is shown in Figure 10.

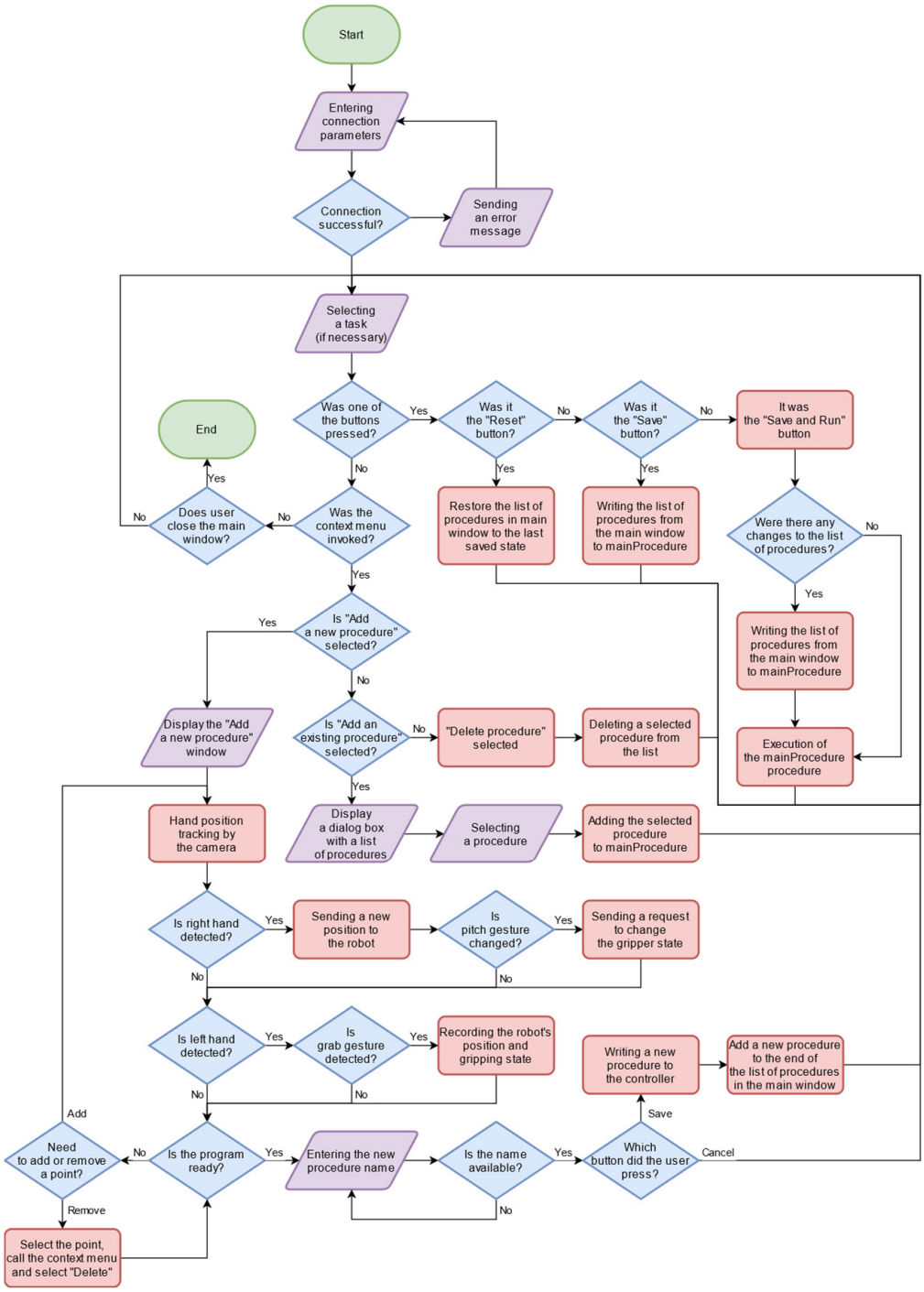

Figure 10. The flowchart of the data flow and control of developed software.

4. Results

Regarding manual programming, classic industrial robots can generally be programmed via jogging. Collaborative robots can also be controlled through lead-through programming. Using these methods simultaneously allows the operator to create programs more flexibly, increases accuracy, and reduces program creation time, especially for complex and precise applications. However, due to the simplicity and naturalness of the interaction, gesture-based manual programming methods can significantly reduce the time needed to create programs, especially for tasks with complex trajectories. Also, due to simplicity and intuitiveness, this approach can reduce the amount of errors in the program, which is especially critical for inexperienced operators.

An experiment was conducted to confirm this theory (Figure 11). As part of the experiment, the operator was required to create a program to move a cube-shaped part to specified positions on the workspace, which were marked from 1 to 6 (Figure 12). The length of the cube-shaped part edges was 20 mm. The experiment consisted of four tests. On the first test, the operator was required create a program to move the part from position 1 to position 6. During the second test, the operator needed to create a program to move the part from position 6 to position 1. On the third test, moving the part from position 4 to position 3 was necessary. The fourth test was to create a program to move the part from position 3 to position 4. On each test of the experiment, the program was created from scratch. The size of the operating area relative to the robot's base coordinate system was as follows: X axis, from 380 to 500 mm; Y axis, from -200 to -40 mm; and Z axis, from 40 to 120 mm. The dimensions of the working area on the base plate were a length of 120 mm and a width of 95 mm. The pitch between the positions for workpiece installation was 75 mm vertically and 50 mm horizontally.

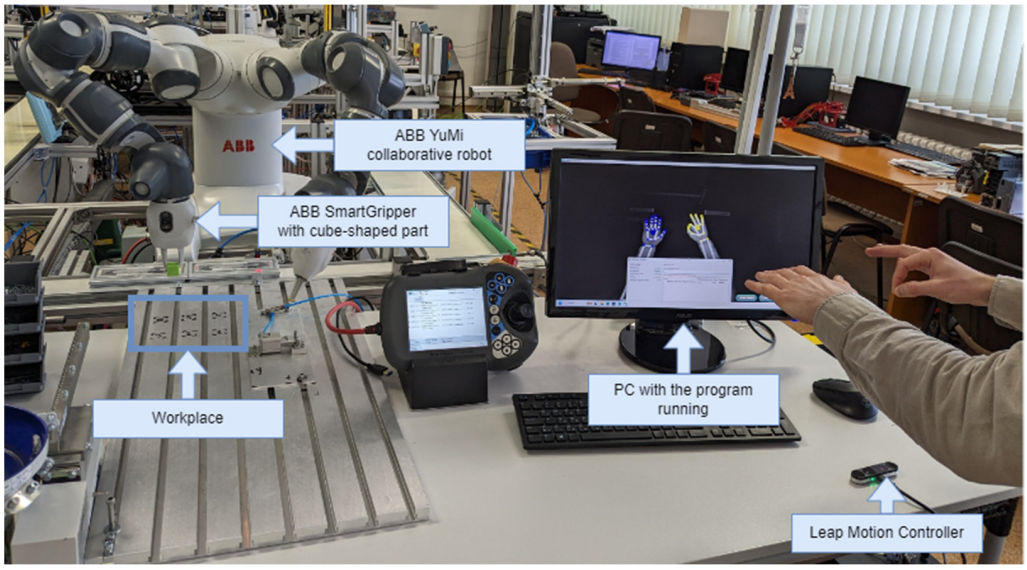

Figure 11. Experimental verification of the study.

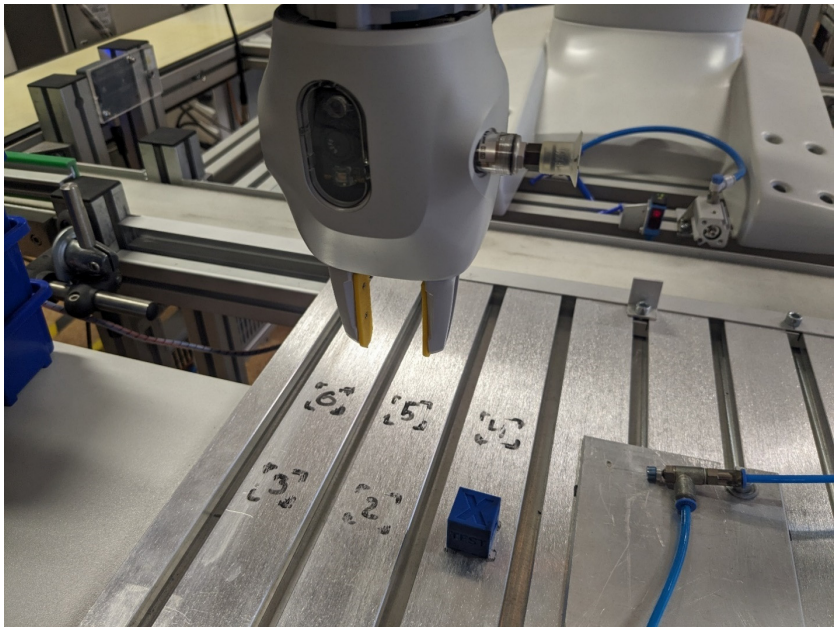

Figure 12. Robot workplace configuration.

As a result, with both the limits of the robot's working area and the camera's working area being known, Equation (4) can be obtained to calculate the current position of the tool center point depending on the position of the hands with Equations (1)–(3). It is also worth considering the different directions of the coordinate systems of the robot and the camera: the *Z* axis of the camera is directed along the *X* axis of the robot, the *X* axis of the camera is directed along the *Y* axis of the robot, and the *Y* axis of the camera is directed along the *Z* axis of the robot.

$$\begin{aligned} x_{tcp} &= 420 + z_{lmc}, \\ y_{tcp} &= -220 + x_{lmc}, \\ z_{tcp} &= -140 + y_{lmc} \end{aligned} \tag{4}$$

The purpose of the experiment was the following. It was necessary to determine the difference in time between the existing and proposed manual programming methods and to analyze the probability of errors occurring during programming. The time was measured from the beginning of program creation to the end of execution of the created program without errors (the first successful debugging run). The results of the experiment are summarized in Table 3. The final table contains the time of creating programs without errors, and the total number of errors during the entire experiment for each method is in a separate row. It is worth considering that the experiment involved an operator who had experience working with ABB industrial robots.

Table 3. Time spent on program creation and a number of errors during the programming process.

Criterion	Jogging + Lead-Through	Gestures
Time spent, test 1, seconds	258	95
Time spent, test 2, seconds	296	106
Time spent, test 3, seconds	333	81
Time spent, test 4, seconds	306	77
Average time spent, seconds	298.25	89.75
Number of errors	2	0

As a result, the proposed approach reduced the program creation time by up to 70%. In addition, when using built-in programming methods at the fourth stage of the experiment, errors occurred twice—the first time, the operator did not reassign one of the points in the program, and the second time, there was a collision between the captured part and the base plate. The time spent creating a program with error correction was 502 and 417 s, respectively. This means that, on average, the presence of an error increases the program creation time by up to 50%. As a result, using the proposed method made it possible to significantly reduce the time required to create a program manually, reducing time losses.

It is also worth noting that since lead-through programming is only available for collaborative robots due to safety issues, the time required to create a program manually for classic industrial robots may increase. However, since the proposed approach, unlike lead-through programming, does not require contact between the operator and the robot, it is available for classical industrial robots.

In addition to testing programming speed and the number of errors encountered during the programming process, the accuracy of object placement was tested. It is necessary to be aware of the proposed approach's accuracy to understand the range of industrial applications for which this programming method is applicable. The principle of this test is that while picking and placing the part, its position along the *X*-axis is measured by a laser sensor. A Micro-Epsilon optoNCDT 1420-10 sensor was used to measure the position of the part. The Micro-Epsilon optoNCDT 1420-10 sensor has a measuring range of 10 mm, the middle of the measuring range is 25 mm, and the sensor can measure the distance to an object with a repeatability of 0.5 μm and a frequency of 2000 Hz. The reference location of the part is set to the middle of the laser sensor's measuring range. The process of the

experiments is shown in Figure 13. Graphs of the distance to the part during the process of picking and placing the part are shown in Figures 14 and 15, respectively.

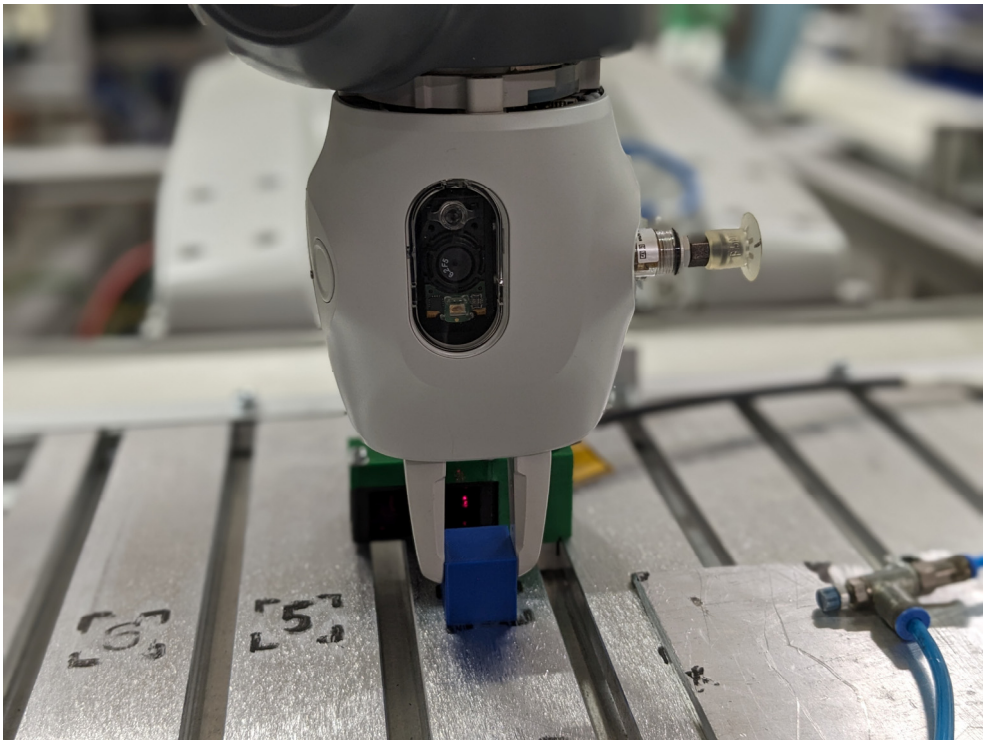

Figure 13. Application accuracy testing.

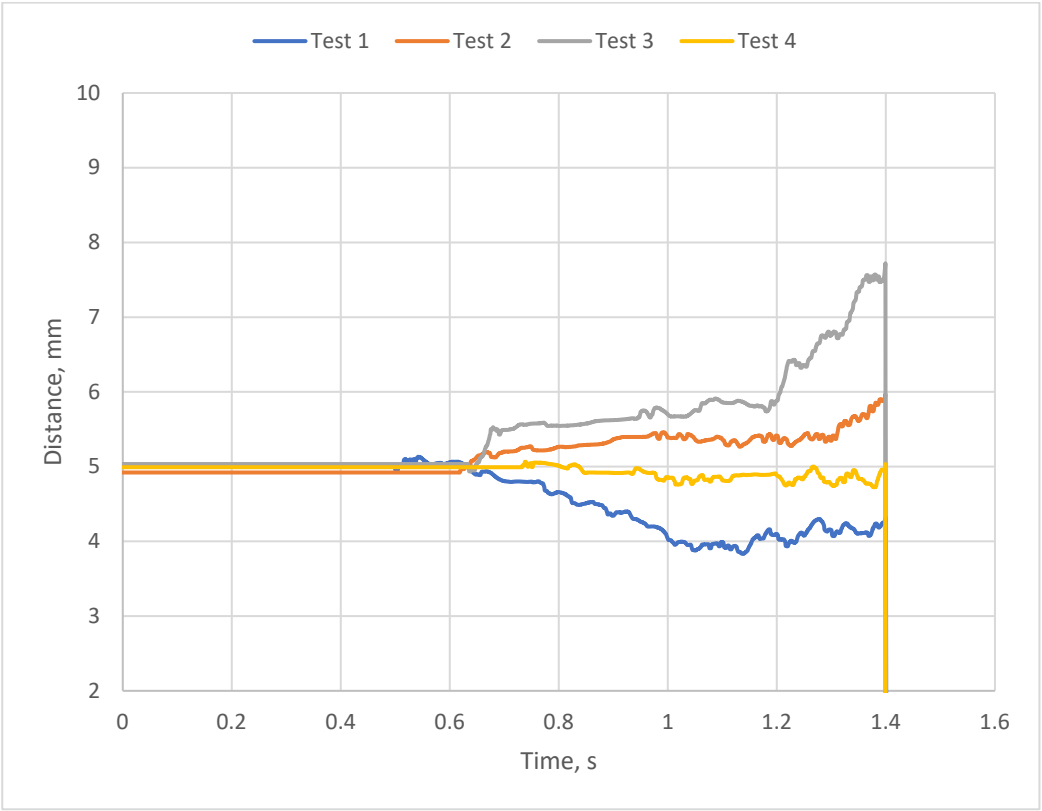

Figure 14. Accuracy of picking operation.

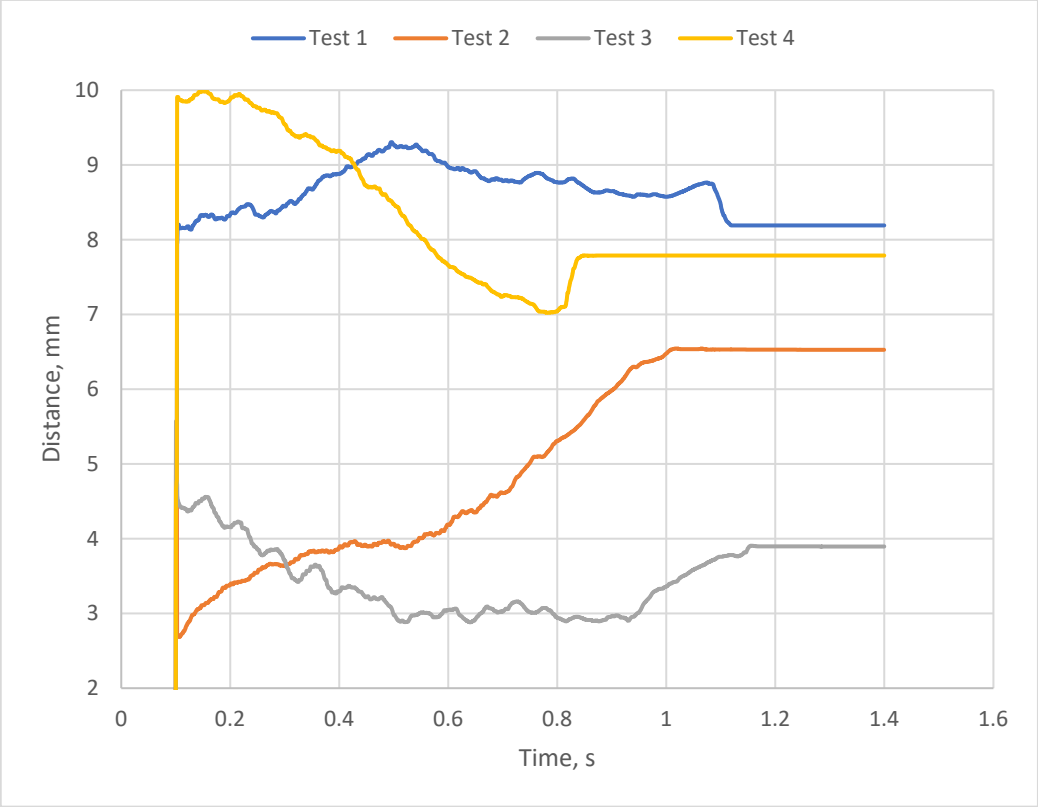

Figure 15. Accuracy of the placing operation.

As a result, the distance deviation during picking operations ranged from −1.1 mm to 2.8 mm; the distance deviation during placing operations ranged from −2.3 mm to 5.0 mm.

5. Discussion

Several studies in this area use gesture-based position control and robot programming methods. For example, in [5], the authors created applications for the teleoperation of the ABB YuMi robot using gestures. The application uses the Leap Motion Controller 1 camera, which can provide information about the position of key points of the hands. In contrast to our proposed solution, the authors in [5] used ROS, which allows for the use of position solvers for this type of robot, thereby flexibly specifying the position of the tool. Our proposed approach has a limited rotational position of the tool—it can only be perpendicular to the base plate. However, the application developed by the authors in [5] is not designed for gesture-based program creation. In addition, the authors in [5] did not implement control of the robot's grippers.

It is also possible to compare the proposed approach with the study described in [18]. In contrast to our proposed method, in [18], the authors exclusively used gesture control, simplifying the organization of the operator's workspace. Also, visual feedback from the robot on the display improved the user experience. However, other researchers [23] have noted that a large number of gestures can increase the load on the user. Also, in [18], the authors did not provide gripper control option, which does not allow for the use of the developed approach for parts handling.

One of the main contributions of this article is as follows. The difference between this article and similar works related to new, more simplified methods lies in the form of interaction with the controller. Particularly, the created programs are directly written to the controller in the native programming language of the controller. The proposed approach allows the creation of modular, flexible, and portable programs. This approach is also of interest to enterprises that can easily integrate the programming method into the production area due to compatibility with existing (previously developed) programs.

6. Conclusions

The article is devoted to creating a methodology for developing an application for programming robots using gestures, and it also contains a detailed description of an example of creating such an application for an ABB YuMi collaborative robot. The article also provides a detailed description of the process of setting up and calibrating the application for gesture-based programming. The proposed concept allows the operator to control the gripper and write new programs directly into the controller using ABB's native robot programming language—RAPID. Moreover, the proposed concept allows for the addition of previously created procedures to the main program list. Thereby, the operator can create flexible programs quickly by reusing the code. The developed application and programs written using this application can work not only with the ABB YuMi collaborative robot but also with all ABB robots with the RobotWare software platform installed from version 6.07.01 to version 6.15.04. Similar interaction protocols are proposed for other popular industrial robots, such as robots from Universal Robots, KUKA, Fanuc, etc., and the proposed approach can be adapted for robots from other manufacturers. Due to these features, the proposed approach can be used in production to create simple programs quickly. Simplified programming methods can significantly reduce the time spent writing a program. During experimental testing, the time spent creating a program was reduced by up to 70%.

The proposed approach is theoretically applicable in production. The X axis part positioning deviations ranged from -2.3 to 5 mm, which should be satisfactory for most pick and place operations (e.g., moving goods in a warehouse, loading and unloading equipment), especially when damping elements are used in the gripper design. However, the accuracy of this type of programming is highly dependent on individual motor skills as well as the visibility of the work area. One of the options for solving this problem can be the adjustment of the sensitivity coefficient of hand movement. Issues of accuracy and implementation of the sensitivity coefficient will be an area of future research.

It is also worth noting that when testing the proposed solution, we encountered gesture recognition problems using the LeapC API's built-in methods. The problem was partially solved by experimentally selecting the working area of the camera. However, when testing the application with inexperienced users, it took time for it to adapt to the gesture recognition system. In addition, there were cases of the false triggering of the gripper release gesture while a part was being moved, even in the recommended working area of the camera. In future research, the authors plan to use other types of cameras in conjunction with neural networks to increase the reliability and intuitiveness of gesture recognition. Problems with gesture recognition could also arise due to lighting conditions: the laboratory contained sunlight and sources of infrared lighting. The applicability of other types of cameras should be tested in the future.

Author Contributions: Conceptualization, V.A. and K.Ž.; methodology, V.A. and K.Ž.; software, V.A.; validation, V.A., V.I. and J.P.; formal analysis, V.A.; investigation, V.A. and V.I.; resources, V.A., K.Ž., V.I. and J.P.; data curation, V.A. and K.Ž.; writing—original draft preparation, V.A.; writing—review and editing, K.Ž. and V.I.; visualization, V.A.; supervision, V.I. and J.P.; project administration, V.I. and J.P.; funding acquisition, V.I. and J.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Slovak Research and Development Agency under contract Nno. APVV-19-0590 and by the projects VEGA 1/0061/23 and KEGA 014TUKE-4/2023 granted by the Ministry of Education, Science, Research and Sport of the Slovak Republic. The research was funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under project no. 09I03-03-V01-00102 and no. 09I03-03-V01-00094. The research was partially supported by The NAWA Ulam Programme under grant number BPN/ULM/2022/1/00045, the Research and Educational Center for Industrial Engineering (Sumy State University), and the International Association for Technological Development and Innovations.

Data Availability Statement: The application's source code is available on request from the authors.

Acknowledgments: The authors wish to acknowledge the European Union's Horizon research and innovation program under the Marie Skłodowska-Curie Grant aAgreement ID 101086487).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hricová, R.; Madzinová, R. Innovations in a Modern Engineering Enterprise in the Context of Industry 4.0 Strategy. *JES* **2023**, *10*, A1–A9. [CrossRef] [PubMed]
2. Evjemo, L.D.; Gjerstad, T.; Grøtli, E.I.; Sziebig, G. Trends in Smart Manufacturing: Role of Humans and Industrial Robots in Smart Factories. *Curr. Robot. Rep.* **2020**, *1*, 35–41. [CrossRef]
3. Karabegović, I.; Karabegović, E.; Mahmić, M.; Husak, E. Implementation of Industry 4.0 and Industrial Robots in the Manufacturing Processes. In *New Technologies, Development and Application II*; Karabegović, I., Ed.; Springer International Publishing: Cham, Switzerland, 2020; pp. 3–14.
4. Bill, M.; Müller, C.; Kraus, W.; Bieller, S. *World Robotics 2022 Report*; International Federation of Robotics: Frankfurt, Germany, 2022.
5. Forgo, Z.; Villanueva Portela, M.A.; Hypki, A.; Kuhlenkoetter, B. Dual Arm Robot Control by Hands Gestures Using ROS. In Proceedings of the ISR 2020—52nd International Symposium on Robotics, Online, 9–10 December 2020; pp. 1–6.
6. Perzylo, A.; Rickert, M.; Kahl, B.; Somani, N.; Lehmann, C.; Kuss, A.; Profanter, S.; Beck, A.B.; Haage, M.; Rath Hansen, M.; et al. SMErobotics: Smart Robots for Flexible Manufacturing. *IEEE Robot. Autom. Mag.* **2019**, *26*, 78–90. [CrossRef]
7. Multi-Annual Roadmap (MAR) for Horizon 2020 Call ICT-2017 (ICT-25, 27 & 28); SPARC, 2016. Available online: https://old.eu-robotics.net/sparc/newsroom/press/multi-annual-roadmap-mar-for-horizon-2020-call-ict-2017-ict-25-27-28-published.html (accessed on 12 February 2023).
8. Schwind, T. *Safe, Fast, and Flexible—Cobots. An Ideal Solution for Small and Mid-Sized Businesses*; IFR: Frankfurt, Germany, 2023.
9. Mukherjee, D.; Gupta, K.; Chang, L.H.; Najjaran, H. A Survey of Robot Learning Strategies for Human-Robot Collaboration in Industrial Settings. *Robot. Comput.-Integr. Manuf.* **2022**, *73*, 102231. [CrossRef]
10. A Survey on End-User Robot Programming | ACM Computing Surveys. Available online: https://dl.acm.org/doi/10.1145/3466819?sid=SCITRUS (accessed on 30 August 2023).
11. Wiese, T.; Abicht, J.; Friedrich, C.; Hellmich, A.; Ihlenfeldt, S. Flexible Skill-Based Control for Robot Cells in Manufacturing. *Front. Robot. AI* **2022**, *9*, 1014476. [CrossRef] [PubMed]
12. Čorňák, M.; Tölgyessy, M.; Hubinský, P. Innovative Collaborative Method for Interaction between a Human Operator and Robotic Manipulator Using Pointing Gestures. *Appl. Sci.* **2022**, *12*, 258. [CrossRef]
13. Michalík, R.; Janota, A.; Gregor, M.; Hruboš, M. Human-Robot Motion Control Application with Artificial Intelligence for a Cooperating YuMi Robot. *Electronics* **2021**, *10*, 1976. [CrossRef]
14. Qin, Y.; Su, H.; Wang, X. From One Hand to Multiple Hands: Imitation Learning for Dexterous Manipulation from Single-Camera Teleoperation. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10873–10881. [CrossRef]
15. Duarte, L.; Safeea, M.; Neto, P. Event-Based Tracking of Human Hands. *Sens. Rev.* **2021**, *41*, 382–389. [CrossRef]
16. Soares, I.; Petry, M.; Moreira, A.P. Programming Robots by Demonstration Using Augmented Reality. *Sensors* **2021**, *21*, 5976. [CrossRef] [PubMed]
17. Puljiz, D.; Stöhr, E.; Riesterer, K.S.; Hein, B.; Kröger, T. General Hand Guidance Framework Using Microsoft HoloLens. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 5185–5190.
18. Vysocký, A.; Poštulka, T.; Chlebek, J.; Kot, T.; Maslowski, J.; Grushko, S. Hand Gesture Interface for Robot Path Definition in Collaborative Applications: Implementation and Comparative Study. *Sensors* **2023**, *23*, 4219. [CrossRef] [PubMed]
19. Handa, A.; Van Wyk, K.; Yang, W.; Liang, J.; Chao, Y.-W.; Wan, Q.; Birchfield, S.; Ratliff, N.; Fox, D. DexPilot: Vision-Based Teleoperation of Dexterous Robotic Hand-Arm System. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 9164–9170.
20. Li, S.; Hendrich, N.; Liang, H.; Ruppel, P.; Zhang, C.; Zhang, J. A Dexterous Hand-Arm Teleoperation System Based on Hand Pose Estimation and Active Vision. *IEEE Trans. Cybern.* **2024**, *54*, 1417–1428. [CrossRef] [PubMed]
21. Adebayo, S.; McLoone, S.; Dessing, J.C. Hand-Eye-Object Tracking for Human Intention Inference. *IFAC-PapersOnLine* **2022**, *55*, 174–179. [CrossRef]
22. Male, J.; Martinez-Hernandez, U. Deep Learning Based Robot Cognitive Architecture for Collaborative Assembly Tasks. *Robot. Comput. Integr. Manuf.* **2023**, *83*, 102572. [CrossRef]
23. Zhao, X.; He, Y.; Chen, X.; Liu, Z. Human–Robot Collaborative Assembly Based on Eye-Hand and a Finite State Machine in a Virtual Environment. *Appl. Sci.* **2021**, *11*, 5754. [CrossRef]
24. Jin, H.; Chen, Q.; Chen, Z.; Hu, Y.; Zhang, J. Multi-LeapMotion Sensor Based Demonstration for Robotic Refine Tabletop Object Manipulation Task. *CAAI Trans. Intell. Technol.* **2016**, *1*, 104–113. [CrossRef]
25. Coordinate Systems—Leap Motion C# SDK v3.2 Beta Documentation. Available online: https://developer-archive.leapmotion.com/documentation/csharp/devguide/Leap_Coordinate_Mapping.html (accessed on 10 October 2023).

26. Tölgyessy, M.; Dekan, M.; Rodina, J.; Duchoň, F. Analysis of the Leap Motion Controller Workspace for HRI Gesture Applications. *Appl. Sci.* **2023**, *13*, 742. [CrossRef]
27. Vysocký, A.; Grushko, S.; Oščádal, P.; Kot, T.; Babjak, J.; Jánoš, R.; Sukop, M.; Bobovský, Z. Analysis of Precision and Stability of Hand Tracking with Leap Motion Sensor. *Sensors* **2020**, *20*, 4088. [CrossRef] [PubMed]
28. Sokolov, O.; Hošovský, A.; Trojanová, M. Design, Modelling, and Control of Continuum Arms with Pneumatic Artificial Muscles: A Review. *Machines* **2023**, *11*, 936. [CrossRef]
29. Shi, J.; Mao, Y.; Li, P.; Liu, G.; Liu, P.; Yang, X.; Wang, D. Hybrid Mutation Fruit Fly Optimization Algorithm for Solving the Inverse Kinematics of a Redundant Robot Manipulator. *Math. Probl. Eng.* **2020**, *2020*, e6315675. [CrossRef]
30. ABB Library—IRB 14000. Available online: https://library.abb.com/r?cid=9AAC184341 (accessed on 24 January 2024).
31. Qt Documentation | Modules. Available online: https://doc.qt.io/qt.html (accessed on 29 October 2023).
32. GitHub - Ros-Industrial/Abb_libegm: A C++ Library for Interfacing with ABB Robot Controllers Supporting Externally Guided Motion (689-1). Available online: https://github.com/ros-industrial/abb_libegm (accessed on 29 October 2023).
33. ABB Robotics Application Manual - Externally Guided Motion - RobotWare 6.14 202. Available online: https://library.e.abb.com/public/4c9bfa6a4e9542bf9386c87f5377a27f/3HAC073319%20AM%20Externally%20Guided%20Motion%20RW6-en.pdf (accessed on 2 September 2023).
34. Leap Motion C API: LeapC Guide. Available online: https://developer.leapmotion.com/documentation/v4/index.html (accessed on 29 October 2023).