Real-Time Hand Gesture Control of a Robotic Arm with Programmable Motion Memory

Daniel Giraldi Michels[2][a], Davi Giraldi Michels[2][b], Lucas Alexandre Zick[1][c], Dieisson Martinelli[1][d], André Schneider de Oliveira[1][e] and Vivian Cremer Kalempa[2][f]

[1]*Graduate Program in Electrical and Computer Engineering, Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, Brazil*

[2]*Department of Information Systems, Universidade do Estado de Santa Catarina (UDESC), São Bento do Sul, Brazil*

Keywords: Human Robot Interface, Computer Vision, Hand Gesture Recognition, Robotic Arm, MediaPipe.

Abstract: The programming of industrial robots is traditionally a complex task that requires specialized knowledge, limiting the flexibility and adoption of automation in various sectors. This paper presents the development and validation of a programming by demonstration system to simplify this process, allowing an operator to intuitively teach a task to a robotic arm. The methodology employs the MediaPipe library for real-time hand gesture tracking, using a conventional camera to translate human movements into a robot-executable trajectory. The system is designed to learn a manipulation task, such as 'pick and place', and store it for autonomous reproduction. The experimental validation, conducted through 50 consecutive cycles of the task, demonstrated the high robustness and effectiveness of the approach, achieving a 98% success rate. Additionally, the results confirmed the excellent precision and repeatability of the method, evidenced by a standard deviation of only 0.0126 seconds in the cycle time. A video demonstrating the system's functionality is available for illustrative purposes, separate from the quantitative validation data. It is concluded that the proposed approach is a viable and effective solution for bridging the gap between human intention and robotic execution, contributing to the democratization of automation by offering a more intuitive, accessible, and flexible programming method.

1 INTRODUCTION

The advancement of robotics has transformed various sectors of society, from manufacturing to personal assistance (Zick et al., 2024; John, 2011). The growing need for autonomous or semi-autonomous systems, capable of executing tasks with high precision, repeatability, and safety, has driven significant progress in both hardware and software (Moustris et al., 2011). This development seeks to optimize productivity and reduce risks in operations that are complex or hazardous for humans.

In the industrial context, robotic arms play a central role in process automation. Their ability to execute complex movements with strength and accuracy, often superior to human capabilities, makes them indispensable in tasks such as assembly, welding, and painting (Javaid et al., 2021). The versatility of this equipment allows for its integration into different stages of production, contributing to the quality of the final product and the competitiveness of companies. However, the efficiency and adaptability of these systems are directly related to the quality of the control interfaces (Zick et al., 2024).

Teleoperation emerges as a crucial approach for the remote control of robotic arms. It allows human operators to manipulate the robot from a distance, in hostile, hazardous, or hard-to-reach environments (Du et al., 2024). This modality is essential in situations that require human intervention for complex decision-making or for the execution of non-programmable tasks. Teleoperation ensures operator safety and expands the scope of robotic operation to scenarios where full autonomy is still a technical challenge (Martinelli et al., 2020).

Currently, the teleoperation methods employed

[a] https://orcid.org/0009-0005-5037-8815
[b] https://orcid.org/0009-0004-9627-0431
[c] https://orcid.org/0009-0001-8645-9781
[d] https://orcid.org/0000-0001-7589-1942
[e] https://orcid.org/0000-0002-8295-366X
[f] https://orcid.org/0000-0001-9733-7352

Michels, D. G., Michels, D. G., Zick, L. A., Martinelli, D., Schneider de Oliveira, A. and Kalempa, V. C.
Real-Time Hand Gesture Control of a Robotic Arm with Programmable Motion Memory.
DOI: 10.5220/0013717800003982

In *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2025) - Volume 2*, pages 292-300
ISBN: 978-989-758-770-2; ISSN: 2184-2809

range from programmatic interfaces, which demand technical knowledge, to the use of physical devices such as joysticks, keyboards, data gloves, and motion capture systems (Zick et al., 2024; Martinelli et al., 2020). Although these approaches have brought advancements, they often present limitations. These include the need for specific and costly hardware, a lack of intuitiveness in real-time control, and the difficulty in replicating fluid and natural movements. Such barriers often restrict large-scale adoption and the optimization of interaction between humans and machines (Moniruzzaman et al., 2022).

To overcome these shortcomings, this paper proposes an innovative control method for articulated robotic arms. The proposed method utilizes real-time recognition of the user's hand position and movements using a single camera. This eliminates the need for complex and proprietary input devices, promoting a more natural, intuitive, and cost-effective interaction. This approach translates human gestures into precise commands for the robot. A distinctive feature of this algorithm is its ability to store and reproduce movement sequences. This functionality is crucial for repetitive tasks, as it significantly reduces the need for continuous reprogramming, optimizing operational efficiency and broadening the system's applicability in scenarios that require high repeatability and agility in task switching.

To validate the proposed method, an articulated robotic arm was employed in a controlled experimental environment, which simulated realistic operating conditions. The validation process focused on the execution and repetition of a 'pick and place' task. Initially, a human operator demonstrated the movement of picking up an object from an origin point and depositing it at a destination, with their gestures being captured and stored by the system via MediaPipe (Google, 2025) and subsequently reproduced repeatedly. This experimental setup allowed for the analysis of the precision, fluidity, and consistency of the robotic control, as well as the verification of the effectiveness of the movement repetition functionality, in which the arm autonomously reproduced the previously recorded action.

The remainder of this paper is structured as follows: Section 2 presents the proposed strategy, detailing the computer vision system and the control algorithm. Section 3 describes the validation and experimental tests, including the hardware and the experimental environment used. Finally, Section 4 concludes the work and suggests directions for future research.

1.1 Related Work

Teleoperation and the intuitive control of robotic arms are topics of great interest in the scientific community, with various approaches proposed to enhance human-robot interaction. This section presents a review of relevant works that address teleoperation systems, motion recognition interfaces, and the use of different technologies for controlling robots.

A teleoperation system for multiple robots, which utilizes an intuitive hand recognition interface, is presented by (Zick et al., 2024). This study focuses on the ability to simultaneously control more than one robot, using the recognition of the operator's hand movements to generate commands. The proposed approach seeks to simplify the inherent complexity of managing multiple machines, offering a more natural solution for interaction.

In the area of human-robot interfaces for remote control using IoT communication, (Martinelli et al., 2020) describe a system that employs deep learning techniques for motion recognition. The work explores how deep neural networks can be utilized to interpret gestures and translate them into commands for the robot, all with the convenience of communication using the Internet of Things (IoT). The focus is on providing efficient and responsive remote control, even over long distances.

Finally, (Franzluebbers and Johnson, 2019) investigate the teleoperation of robotic arms through virtual reality. The research explores how virtual environments can offer an immersive experience that enhances the operator's perception of the robot's environment, thereby facilitating precise control. The use of virtual reality aims to overcome some of the limitations of traditional teleoperation methods, providing a more intuitive experience rich in visual and spatial feedback for the user.

These works illustrate the diversity of technological solutions aimed at facilitating robotic control, ranging from physical sensors and neural networks to immersive environments. In line with these initiatives, this work proposes an approach that combines real-time hand gesture recognition, via a conventional camera and MediaPipe, with movement recording and playback functionality. The system aims to offer a low-cost and high-usability alternative, suitable for applications that require repeatability, simplicity of implementation, and minimal additional infrastructure.

2 PROPOSED STRATEGY

The proposed system aims to enable the intuitive and real-time control of an articulated robotic arm through hand gestures captured by a single RGB camera, using the MediaPipe library. The main innovation lies in the combination of gesture recognition without additional sensors and the ability to store and reproduce previously demonstrated trajectories, enabling applications in repetitive tasks with low cost and high accessibility.

Figure 1 illustrates the overall architecture of the system. It is composed of three main modules: (i) gesture capture and interpretation, (ii) command mapping and transmission to the robot, and (iii) trajectory recording and playback. The interaction occurs in real time, allowing the operator to directly control the robotic arm's movements with their hand, without physical contact or external devices.

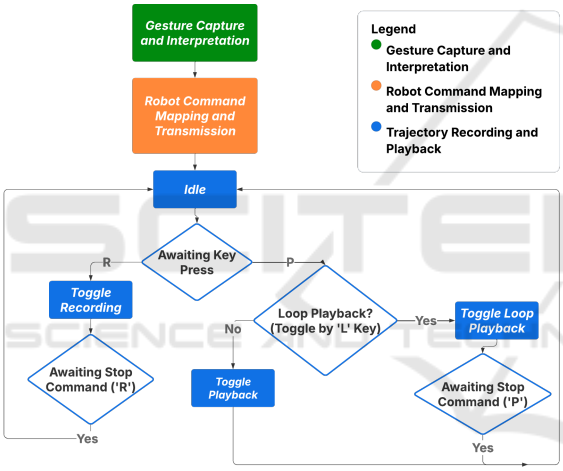

Figure 1: System flowchart.

Figure 2: System usage.

First, hand motion capture is performed using a standard RGB camera, positioned to record the operator's gestures in a frontal plane. For gesture recognition, the MediaPipe Hands framework is used. The positions extracted from the operator's hand are mapped to the robot's workspace through a linear calibration function, which adjusts the normalized values from the camera to the manipulator's real-world coordinates. During manual operation, the sequential positions of the robotic arm are recorded in a temporal data structure. Once stored, the sequence can be automatically replayed by the robot, faithfully replicating the movements performed in the original demonstration. This mechanism allows for Programming by Demonstration without the need for coding or complex interfaces.

With the system architecture defined, it becomes necessary to establish a suitable environment for its implementation and validation. The following describes the aspects related to environment setup, image acquisition and preprocessing, hand detection and tracking, gesture mapping and control logic, and finally, the movement recording and playback module.

2.1 Environment Setup

The development and validation of the gesture control and Programming by Demonstration system were conducted using the following hardware and software configuration, detailed to ensure the clarity and potential reproducibility of the results.

The central hardware component is the Interbotix PincherX-100 robotic arm, a manipulator with 4 degrees of freedom plus a gripper.

Figure 3: PincherX-100 Robot Arm (Trossen Robotics, 2025).

For this gesture control system, three main joints (base, shoulder, elbow) are actively controlled by hand gestures, while the wrist pitch joint is maintained in a neutral position (0.0 radians). The low-level communication and control of the PincherX-100 are managed through the Robot Operating System (ROS) (Open Source Robotics Foundation, 2025). The high-level Python programming interface with the robot is facilitated by the `interbotix_xs_modules` library, which abstracts the underlying ROS topics and services, allowing for simplified joint command and gripper control from the application script.

The image capture of the operator's hand, which is essential for the vision system, is performed by an RGB camera.

The control software, including the vision processing, the gesture mapping logic, and the movement

recording/playback module, was entirely developed in Python. This software ran on a Dell OptiPlex 5070 desktop computer, equipped with an Intel Core i7 9th Gen processor and 16 GB of RAM, running on the Ubuntu 20.04.6 LTS operating system.

The main Python libraries used in the development of the system include:

- **OpenCV (version 4.11.0) (Bradski, 2025):** Used for webcam frame acquisition, image preprocessing (such as converting the color space from `BGR` to `RGB`), and for displaying the visual feedback interface to the user.
- **MediaPipe (version 0.10.11) (Google, 2025):** Employed for the robust detection and real-time tracking of the 21 three-dimensional landmarks of the operator's hand.
- **NumPy (version 1.24.4):** Used extensively for efficient numerical operations, especially in manipulating landmark coordinates and distance calculations.

This configuration enabled the implementation and testing of the proposed system with low cost, high responsiveness, and good stability in movement execution, serving as the basis for the experiments reported in the next section.

2.2 Image Acquisition and Preprocessing

The webcam interface and video frame management are handled by the OpenCV library. In the context of this project, the library was used to capture video from the RGB camera, converting the frames to the appropriate format for processing. Furthermore, it was used to render graphical overlays of the detected hand points onto the image shown to the operator, as can be seen in Figure 4.

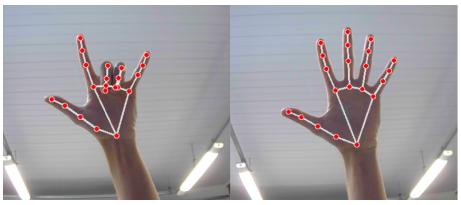

Figure 4: Hand points recognition with MediaPipe.

Once the video frame is captured and preprocessed, it is subsequently sent to the vision processing module. The primary function of this module is the precise detection and real-time tracking of the operator's hand and its respective reference points (Amprimo et al., 2024). For this purpose, the system utilizes the MediaPipe Hands solution.

MediaPipe Hands is designed to identify the presence of hands within the camera's field of view and, for each detected hand, estimate the three-dimensional coordinates of 21 landmarks (Gomase et al., 2022). These landmarks represent key anatomical points of the hand, including the finger joints (metacarpophalangeal, proximal interphalangeal, and distal interphalangeal), the fingertips, and the wrist (Wagh et al., 2023), as shown in Figure 5.

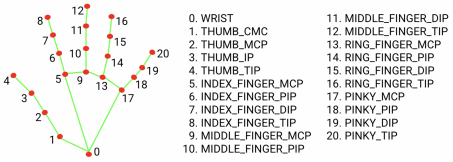

Figure 5: Hand Landmarks. Extracted from (Google, 2025).

In the context of this research, the MediaPipe Hands library is initialized to track a single hand, aiming for a clear and direct gesture control interface for the PincherX-100 robotic arm.

The set of these 21 three-dimensional landmarks provides a detailed representation of the operator's hand pose, orientation, and configuration (Bensaadallah et al., 2023). This information is fundamental and serves as direct input for the gesture mapping module, where the landmark data is interpreted to generate the specific commands that will control the movements of the robotic arm's joints and gripper.

From the set of hand landmarks provided by MediaPipe, as depicted in Figure 5, primary control variables that represent the operator's movement intention are calculated. The key landmarks for this stage include the wrist (landmark 0), the base of the middle finger (landmark 9), the tip of the thumb (landmark 4), and the tip of the index finger (landmark 8).

To enable control of the robotic arm's reach, simulating a forward and backward movement, a depth variable, denoted as `prof`, is calculated. This variable corresponds to the three-dimensional Euclidean distance between the wrist landmark ($P_0 = (x_0, y_0, z_0)$) and the landmark at the base of the middle finger ($P_9 = (x_9, y_9, z_9)$). The formula for `prof` is given by:

$$prof = \sqrt{(x_9 - x_0)^2 + (y_9 - y_0)^2 + (z_9 - z_0)^2} \quad (1)$$

This distance `prof` serves as an indicator of the operator's hand depth in relation to the camera. The calculation of `prof` uses the coordinates of the wrist (P_0) and middle finger base (P_9) landmarks, including the 'z' component provided by MediaPipe, which estimates the relative depth of the landmarks. When the hand moves away from the camera, due to the perspective effect, the hand's projection in the image decreases in size. Consequently, the calculated distance

`prof` tends to *decrease*. Conversely, moving the hand closer to the camera tends to result in a *larger* value for `prof`. Thus, `prof` functions as an indicator of the hand's distance from the camera, where smaller values signify greater distance and larger values signify closer proximity. This variable is subsequently mapped to control the robot's shoulder joint, modulating the reach of the end-effector.

The raw values of the control variables extracted from the hand (`centerX`, `centerY`, `prof`), which are in units of pixels or relative distances, need to be converted to the appropriate angles (in radians) for the corresponding joints of the PincherX-100 robot.

Table 1: Joint angles in radians for the PincherX-100 robot.

Joints	Min	Max
Waist	$-\pi$	$+\pi$
Shoulder	-1.9	$+1.8$
Elbow	-2.1	$+1.6$

This conversion is performed through a linear mapping function, named `map_hand_to_robot` in the code, implemented according to the following equation:

$$V_{\text{robot}} = \frac{(V_{\text{hand}} - V_{\text{hand\_min}})}{(V_{\text{hand\_max}} - V_{\text{hand\_min}})} \times (V_{\text{robot\_max}} - V_{\text{robot\_min}}) + V_{\text{robot\_min}} \quad (2)$$

The horizontal movement of the operator's hand, captured by `centerX`, controls the robot's base (`waist`) joint. As the hand moves from the left to the right side of the camera's field of view (from 0 to 640 pixels in width), the robot's base rotates from $+\pi$ to $-\pi$ radians.

The shoulder joint (`shoulder`) is modulated by the depth variable `prof`. Based on empirical observations, the range of `prof` that represents a comfortable control zone (from 60, for the hand furthest away, to 170, for the hand closest) is mapped to the shoulder angles, ranging from -1.9 to $+1.8$ radians.

The vertical movement of the hand, represented by `centerY`, commands the elbow joint (`elbow`). As the operator's hand moves from the top to the bottom of the image (from 0 to 480 pixels in height), the elbow angle is adjusted from $+1.6$ to -2.1 radians.

It is important to note that, to simplify this gesture control scheme, the wrist pitch (`wrist_angle`), which would be the fourth joint, is kept in a neutral and fixed position of 0.0 radians. The robot's fifth joint, responsible for wrist rotation, is not controlled by this system.

2.3 Command Activation and Sending

To ensure that the robot responds only to intentional movements and to allow the operator to freely position their hand in the camera's field of view before initiating control, the system requires a specific "activation pose." Robot control and movement recording are enabled only when the tip of at least one of the three central fingers — pinky, ring, or middle — is positioned above the point where the finger meets the palm of the hand. This condition, detected by analyzing the relative position of these fingers' *landmarks*, signals the user's explicit intention to control the manipulator.

Once control is activated, the robot's gripper command is determined binarily (open or close) by the distance between the tip of the thumb and the tip of the index finger (`finger_distance`). If this distance is less than a threshold of 35 pixels, the command to close the gripper is sent; otherwise, it is triggered to open.

After calculating the desired angles for the robot's joints and defining the gripper's state, these commands are immediately transmitted to the PincherX-100. The joint positions are sent as a vector, and the arm's control function is used with a movement time of 0.3 seconds, which ensures a smooth transition between movements. It is crucial that the system does not wait for the completion of each individual movement before processing new camera information and subsequent gestures. This feature enables continuous and highly responsive control, which is crucial for both manual operation and trajectory recording. The gripper commands, in turn, are sent for immediate execution, without delay.

2.4 Movement Recording and Playback Module

To empower users without specialized knowledge in robotic programming to define sequential tasks for the robotic arm, a movement recording and playback module was developed. This module allows the operator to demonstrate a trajectory and gripper actions using the gesture control system, record that demonstration, and subsequently command the robot to replay it autonomously.

The module operates based on a finite-state machine, with three main states that dictate the system's behavior:

- **Idle State (`STATE_IDLE`):** In this state, the operator can freely control the robot through hand gestures. No movements are recorded. This is the default state when the system starts and after the

completion or interruption of recording or playback.

- **Recording State (`STATE_RECORDING`):** When the operator triggers the start of recording (by pressing the 'R' key), the system transitions to this state. All joint position commands and gripper states generated from the hand gestures are captured and stored sequentially in memory. Pressing 'R' again ends the recording and returns the system to the idle state.
- **Playback State (`STATE_PLAYING`):** After a movement sequence has been recorded, the operator can start the playback by pressing the 'P' key. The system then executes the stored movements. Pressing 'P' during playback stops the process and returns the system to the idle state. Additionally, the 'L' key allows the user to toggle the loop playback functionality (`loop_playback`), enabling the recorded sequence to be repeated continuously until interrupted.

Feedback on the system's current state (Idle, Recording, Playing), the number of recorded movements, and the loop mode status is provided to the user through console messages and text overlaid on the camera's visual interface.

The demonstrated movements are stored in memory as an ordered list, named `recorded_movements`. Each element of this list represents a discrete "step" of the trajectory and is a Python dictionary containing two main keys:

- `'joints'`: A list with the four angular values (in radians) commanded to the robot's joints (`[waist, shoulder, elbow, 0.0]`) at that moment in the recording.
- `'gripper_closed'`: A boolean value that indicates the commanded state of the gripper (`True` for closed, `False` for open) at that same instant.

This structure allows for a simple and effective representation of the sequence of actions to be replayed. With each new recording session, the `recorded_movements` list is reset.

During the `STATE_RECORDING` state, whenever the gesture control activation condition is met and a new set of joint positions and gripper state is calculated, this data set is encapsulated in the dictionary format described above and appended to the end of the `recorded_movements` list. The recording occurs at the same frequency as the commands are sent to the robot during manual control, capturing the dynamics of the operator's demonstration.

The playback of recorded movements follows a structured logic to ensure the fidelity and safety of the execution:

1. **Movement to the Trajectory's Initial Position:** Upon entering the `STATE_PLAYING` state, and before executing the main sequence, the system commands the robot to move to the joint configuration of the first movement stored in the `recorded_movements` list. This step is crucial to ensure that playback begins from a known and reachable starting point, mitigating potential issues with speed or joint limits that could occur if the robot attempted to jump from an arbitrary position to the start of the recorded trajectory. This initial movement is executed with a relatively long `moving_time` (e.g., 2.0 to 10.0 seconds, adjusted empirically) and with `blocking=True` in the `bot.arm.set_joint_positions()` call, ensuring that the robot reaches this position before proceeding. The initial gripper state is also applied.
2. **Sequential Movement Execution:** After the initial positioning, the system iterates over the `recorded_movements` list. For each step:
   - The stored joint angles are sent to the robot using `bot.arm.set_joint_positions()`. To ensure the complete and orderly execution of each step in the trajectory, the `blocking=True` parameter is used, and a consistent `moving_time` (e.g., 0.5 to 0.7 seconds) is set for each movement.
   - The gripper state (open or closed) stored for that step is commanded to the robot.
3. **Loop Playback:** If the `loop_playback` variable is set to `True`, at the end of the execution of all movements in the `recorded_movements` list, the playback index (`current_play_index`) is reset to the beginning of the list, and the process of moving to the initial position and executing the sequence is repeated.

This recording and playback module aims to significantly simplify the programming of repetitive robotic tasks, making the system accessible to operators without a background in programming.

3 EXPERIMENTAL VALIDATION

This section details the procedures and results of the validation tests conducted to evaluate the effectiveness, precision, and repeatability of the proposed system[1].

[1]The demonstration video can be accessed at https://www.youtube.com/watch?v=Zf2bXpjEzRs

To validate the system, a ”pick and place” test was developed. This test simulated a common task in industrial robotics and demonstrated the system's ability to perform complex and repetitive movements with precision. The experimental environment used a 4 cm cube as the test object. The task required the robotic arm to pick up the cube from an initial platform, elevated 13 cm from the ground, and place it at a destination point on a ground-level surface approximately 30 cm away.

The experiment consisted of:

1. **Movement Definition:** The 'pick and place' movement was defined as the act of the robotic arm picking up an object at an origin point and releasing it at a specific destination point. This movement was first performed and stored by the system.
2. **Movement Repetition:** Following storage, the system was set to repeat this movement 50 consecutive times without manual intervention.
3. **Data Logging:** During the 50 repetitions, the following metrics were monitored and recorded:
   - **Number of Successful Attempts:** A count of the times the robotic arm successfully picked up the object and released it correctly at the destination point. An attempt was considered successful when the object was deposited within a predefined area of ± 1 cm from the target point and without being dropped.
   - **Number of Failures:** A count of the times the robotic arm failed to complete the task. Failures were categorized for later analysis (e.g., object not picked, object dropped during transport, object released outside the target area).
   - **Time per Cycle:** The total time taken to complete each 'pick and place' cycle (from the start of the picking motion to the completion of the releasing motion) was recorded.

3.1 Results

The quantitative results obtained from the 50 repetitions of the 'pick and place' test are presented in Table 2. The evolution of the execution time over the cycles is detailed in Figure 6.

Table 2: ”Pick and Place” Test Results (50 Repetitions).

Metric	Value
Total Trials	50
Successful Trials	49
Failures	1
Success Rate	98%
Mean Cycle Time (s)	13.35
Time Std. Deviation (s)	0.0126

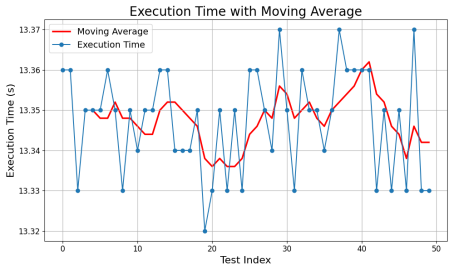

Figure 6: Execution time per cycle for 50 trials, with the moving average in red.

3.2 Discussion of Results

The 98% success rate, as shown in Table 2, demonstrates the high effectiveness and robustness of the proposed system in performing repetitive movements from a recorded gesture. The single recorded failure occurred due to a minor slippage of the object in the gripper, suggesting that future improvements could focus on the actuator's gripping mechanism rather than on the control logic or the vision system.

The system's operational efficiency is evidenced by the mean cycle time of 13.35 seconds, with a low standard deviation of 0.0126 seconds. This consistency is visually corroborated by Figure 6, which plots the execution time for each of the 50 cycles. The red line, representing the moving average, remains remarkably stable, indicating that there was no performance degradation or increase in variability over time. The minor fluctuations observed are to be expected in a physical-mechanical system.

Taken together, these results confirm that the system is capable of interpreting hand movements, storing them, and replaying them with high precision and repeatability, making it a viable and reliable solution for robotic tasks that require intuitive interaction and the automation of predefined movements

4 CONCLUSION

This work demonstrated the development and validation of a robotic programming by demonstration system, based on human hand gesture capture with the MediaPipe library. The proposed methodology achieved its central objective: to validate an intuitive approach for controlling a robotic arm that significantly reduces the complexity associated with traditional programming. The obtained results confirm that the system is capable of learning and reproducing manipulation tasks with high fidelity.

The quantitative analysis of the validation tests, which consisted of 50 autonomous cycles of a 'pick

and place' task, serves as the primary evidence of the system's robustness. The 98% success rate (49 out of 50 attempts) corroborates the effectiveness of the processing chain, from motion capture to execution by the actuator. Additionally, the low standard deviation of 0.0126 seconds in cycle time attests to the high precision and repeatability of the method, indispensable characteristics for any industrial or research application that demands consistency.

The implications of these results point to a notable potential for application in real-world scenarios. In the industrial sector, especially for small and medium-sized enterprises, the technology can enable low-cost automation for assembly, packaging, or quality control tasks, where flexibility and rapid reprogramming are more critical than extreme speed. In the research field, the system presents itself as a rapid prototyping platform for studies in Human-Robot Interaction (HRI), allowing for the testing of new communication and control modalities. Its educational value is also considerable, serving as a practical tool for teaching concepts in kinematics, automation, and computer vision in an applied and engaging manner.

In summary, the main contribution of this work lies in the empirical validation of a solution that integrates advanced computer vision to create an effective and accessible human-robot interface. The research demonstrates that it is feasible to abstract the complexity of robotic programming, offering a control method that is both powerful in its precision and simple in its use, representing a practical advancement in the pursuit of more flexible and human-centered automation systems.

Future directions for this work focus on evolving the system beyond simple repetition, aiming for greater intelligence and flexibility. The next step will be the implementation of conditional operations, allowing the robot to perform different actions based on specific gestures. In parallel, the expansion of the movement repertoire will be pursued to include more complex tasks, such as contour following. A crucial objective will also be the abstraction of the software layer to ensure the solution's portability, enabling its adaptation to control different models of robotic arms with minimal reconfiguration.

ACKNOWLEDGEMENTS

The project is supported by the National Council for Scientific and Technological Development (CNPq) under grant number 407984/2022-4; the Fund for Scientific and Technological Development (FNDCT); the Ministry of Science, Technology and Innovations (MCTI) of Brazil; Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES); the Araucaria Foundation; the General Superintendence of Science, Technology and Higher Education (SETI); and NAPI Robotics.

REFERENCES

Amprimo, G., Masi, G., Pettiti, G., Olmo, G., Priano, L., and Ferraris, C. (2024). Hand tracking for clinical applications: validation of the google mediapipe hand (gmh) and the depth-enhanced gmh-d frameworks. *Biomedical Signal Processing and Control*, 96:106508.

Bensaadallah, M., Ghoggali, N., Saidi, L., and Ghoggali, W. (2023). Deep learning-based real-time hand landmark recognition with mediapipe for r12 robot control. In *2023 International Conference on Electrical Engineering and Advanced Technology (ICEEAT)*, volume 1, pages 1–6. IEEE.

Bradski, G. (2025). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Du, J., Vann, W., Zhou, T., Ye, Y., and Zhu, Q. (2024). Sensory manipulation as a countermeasure to robot teleoperation delays: system and evidence. *Scientific Reports*, 14(1):4333.

Franzluebbers, A. and Johnson, K. (2019). Remote robotic arm teleoperation through virtual reality. In *Symposium on Spatial User Interaction*, pages 1–2.

Gomase, K., Dhanawade, A., Gurav, P., Lokare, S., and Dange, J. (2022). Hand gesture identification using mediapipe. *International Research Journal of Engineering and Technology (IRJET)*, 9(03).

Google (2025). Mediapipe: A framework for building perception pipelines.

Javaid, M., Haleem, A., Singh, R. P., and Suman, R. (2021). Substantial capabilities of robotics in enhancing industry 4.0 implementation. *Cognitive Robotics*, 1:58–75.

John, T. S. (2011). Advancements in robotics and its future uses. *International Journal of Scientific & Engineering Research*, 2(8):1–6.

Martinelli, D., Cerbaro, J., Fabro, J. A., de Oliveira, A. S., and Teixeira, M. A. S. (2020). Human-robot interface for remote control via iot communication using deep learning techniques for motion recognition. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, pages 1–6. IEEE.

Moniruzzaman, M., Rassau, A., Chai, D., and Islam, S. M. S. (2022). Teleoperation methods and enhancement techniques for mobile robots: A comprehensive survey. *Robotics and Autonomous Systems*, 150:103973.

Moustris, G. P., Hiridis, S. C., Deliparaschos, K. M., and Konstantinidis, K. M. (2011). Evolution of autonomous and semi-autonomous robotic surgical sys-

tems: a review of the literature. *The international journal of medical robotics and computer assisted surgery*, 7(4):375–392.

Open Source Robotics Foundation (2025). Robotic operating system (ros).

Trossen Robotics (2025). PincherX 100 Robot Arm - Interbotix XS-Series Arms.

Wagh, V. P., Scott, M. W., and Kraeutner, S. N. (2023). Quantifying similarities between mediapipe and a known standard for tracking 2d hand trajectories. *bioRxiv*, pages 2023–11.

Zick, L. A., Martinelli, D., Schneider de Oliveira, A., and Cremer Kalempa, V. (2024). Teleoperation system for multiple robots with intuitive hand recognition interface. *Scientific Reports*, 14(1):1–11.