Mechanical Engineering – Scientific Journal, Vol. **43,** No. 2, pp. 101–109 (2025)
Number of article: 698
In print: ISSN 1857–5293
Received: October 20, 2025
On line: ISSN 1857–9191
Accepted: November 2, 2025
UDC: 621.865.8:[004.896:621.766]
https://doi.org/10.55302/MESJ25432101k

Original scientific paper

INTELLIGENT CONTROL OF KUKA ROBOTIC SYSTEMS BASED ON AI-DRIVEN HUMAN MOTION TRACKING

Ivan Kukoski, Viktor Gavriloski, Simona Domazetovska Markovska

Faculty of Mechanical Engineering, "Ss. Cyril and Methodius" University in Skopje, P.O. Box 464, MK-1001 Skopje, Republic of North Macedonia
ivankukoski5@gmail.com

A b s t r a c t: Industrial robot control, often restricted by proprietary systems like KUKA Robot Language, presents a challenge for advanced scientific research and intuitive programming. This paper introduces a novel, low-cost framework for Intelligent Control of KUKA Robotic Systems using AI-Driven Human Motion Tracking to facilitate kinesthetic teaching. The system integrates the MediaPipe Hands Deep Learning model for real-time 3D hand landmark tracking with a custom PyOpenShowVar/KUKAVARPROXY control middleware, enabling soft real-time command transmission to a KUKA KR 16-2. The framework achieved 97.3% accuracy for discrete gesture commands and used a Direct Landmark Differencing approach to provide intuitive, simultaneous control over 3D joint-space movement. While exhibiting 200 ms soft real-time overhead, the performance is highly suitable for path teaching and significantly lowers the technical barrier for human-robot collaboration and flexible manufacturing.

Key words: KUKA manipulator; computer vision; AI-driven control; deep learning; real-time control

КОНТРОЛА НА РОБОТСКИТЕ СИСТЕМИ KUKA ЗАСНОВАНА НА СЛЕДЕЊЕ НА ЧОВЕЧКОТО ДВИЖЕЊЕ СО ПРИМЕНА НА ВЕШТАЧКАТА ИНТЕЛИГЕНЦИЈА

А п с т р а к т: Контролата на индустриските роботи, која често е ограничена од сопствен програмски јазик како што е KUKA Robot Language, претставува сериозен предизвик за напредни научни истражувања и за развој на интуитивни методи на програмирање. Овој труд претставува иновативна и економична рамка за интелигентна контрола на роботските системи KUKA, која користи вештачка интелигенција и следење на човечко движење со цел да се овозможи кинестетичко учење. Развиениот систем го комбинира длабоконевронскиот модел MediaPipe Hands за следење на тридимензионални координати на движењата на раката во реално време, со сопствено развиен посреднички слој за комуникација PyOpenShowVar/KUKAVARPROXY, кој овозможува реалновременска размена на команди со роботот KUKA KR 16-2. Предложената рамка постигнува 97,3% точност при препознавање поединечни гест-команди, со што овозможува интуитивна и истовремена контрола на движењата во тридимензионален простор. Покрај регистрираното доцнење од околу 200 ms во реално време, постигнатите перформанси се целосно соодветни за учење на патеки и значително ја намалуваат техничката бариера за колаборацијата човек – робот и за флексибилно производство.

Клучни зборови: KUKA-манипулатор; компјутерска визија; контрола со вештачка интелигенција; контрола во реално време

1. INTRODUCTION

Industrial manipulators, such as those manufactured by KUKA, are the cornerstone of modern automation, characterized by their precision, robustness, and speed. However, their primary control architecture is designed for industrial efficiency and safety, often relying on proprietary, text-based languages like the KUKA Robot Language (KRL). While suitable for repetitive, pre-programmed manufacturing tasks, this closed-system design presents significant barriers to scientific research [1, 2]. Researchers often seek maximum control, low-level access to variables, and the ability to integrate

advanced mathematics or third-party libraries-functionalities that KRL inherently limits. Consequently, fully exploiting the mechanical capabilities of high-performance industrial platforms in scientific and novel control contexts is often impossible.

To overcome the limitations imposed by industrial controllers, the robotics research community utilizes various middleware and communication interfaces. This is crucial for implementing advanced control methodologies, such as complex Human-Robot Interaction (HRI) schemes or Learning from Demonstration (LfD). Specifically, our approach leverages an open communication interface that not only grants soft real-time access to robot state and control variables but also includes a Python class capable of generating KRL source files (.src) offline. This feature allows users to program complex robot paths in a flexible, high-level environment like Python, completely eliminating the need for specialized KRL knowledge and significantly reducing programming time.

Building on this flexible control foundation, this paper presents a novel framework for Intelligent Control of KUKA Robotic Systems Based on AI-Driven Human Motion Tracking. The core of the system utilizes Deep Learning techniques for robust Hand Gesture Recognition, capturing the operator's movements and translating them into intuitive trajectory commands. This AI-driven approach drastically simplifies the teaching process by translating intuitive human motion directly into executable robot code, saving considerable time and lowering the technical expertise barrier for operators. This research aims to achieve a viable and efficient method for kinesthetic teaching that allows non-expert users to program complex tasks. By seamlessly coupling cutting-edgeAI-based perception with an agile control interface, we demonstrate the potential of open control methods to enable flexible manufacturing and truly intuitive human-robot collaboration.

Prior research in robotics has addressed the challenges of flexible control through various approaches. Regarding industrial system accessibility, several works have proposed software interfaces and middleware to unlock low-level control of platforms like KUKA, attempting to bridge the gap between proprietary KRL and external programming environments [3, 4]. Crucially, traditional online communication with KUKA robots is fundamentally limited to additional packages like KUKA. RobotSensorInterface and KUKA. Ethernet KRL-XML, which restrict the I/O capacity and the complexity of the external control loop [5]. Concurrently, the HRI-field has advanced through Learning from Demonstration (LfD) [6], and more recently, Deep Learning (DL) techniques have demonstrated exceptional performance in real-time vision tasks for hand gesture and human pose estimation [, 8, 9]. While these three areas – open control interfaces, DL – based perception, and LfD – have been explored individually, a unified framework that integrate a high-fidelity, DL – driven gesture system with a flexible, soft real-time KUKA – control architecture remains a critical need.

The key contributions of this work are: (1) The successful integration of a DL-based hand gesture recognition system with an external control interface to realize a stable and responsive soft real-time control loop for the KUKA controller. (2) The development of a precise control and mapping strategy that translates gesture and pose data into safe and stable KUKA joint space trajectories. (3) Comprehensive experimental validation demonstrates the accuracy, low latency, and ease-of-use of the proposed intelligent control framework for complex collaborative tasks, enabled by our flexible, middleware-based architecture. The remainder of this paper is organized as follows: Section 2 details the system architecture, hardware components, software environments, and the connectivity setup. Section 3 presents the experimental setup. Section 4 discusses the results and performance evaluation. Finally, Section 5 offers the conclusion and future work.

2. SYSTEM ARCHITECTURE

System architecture overview

The proposed intelligent control system operates on a distributed architecture designed to facilitate intuitive human interaction with industrial robots. As illustrated in Figure 1, the framework comprises three primary interconnected layers: the human operator providing intuitive inputs, an external laptop/PC responsible for integrated perception and control logic, and the KUKA robot controller (KRC) with its attached KUKA KR 16-2 manipulator for physical task execution. The system uses DL techniques for robust hand gesture recognition and a custom middleware for flexible robot control and offline program generation.

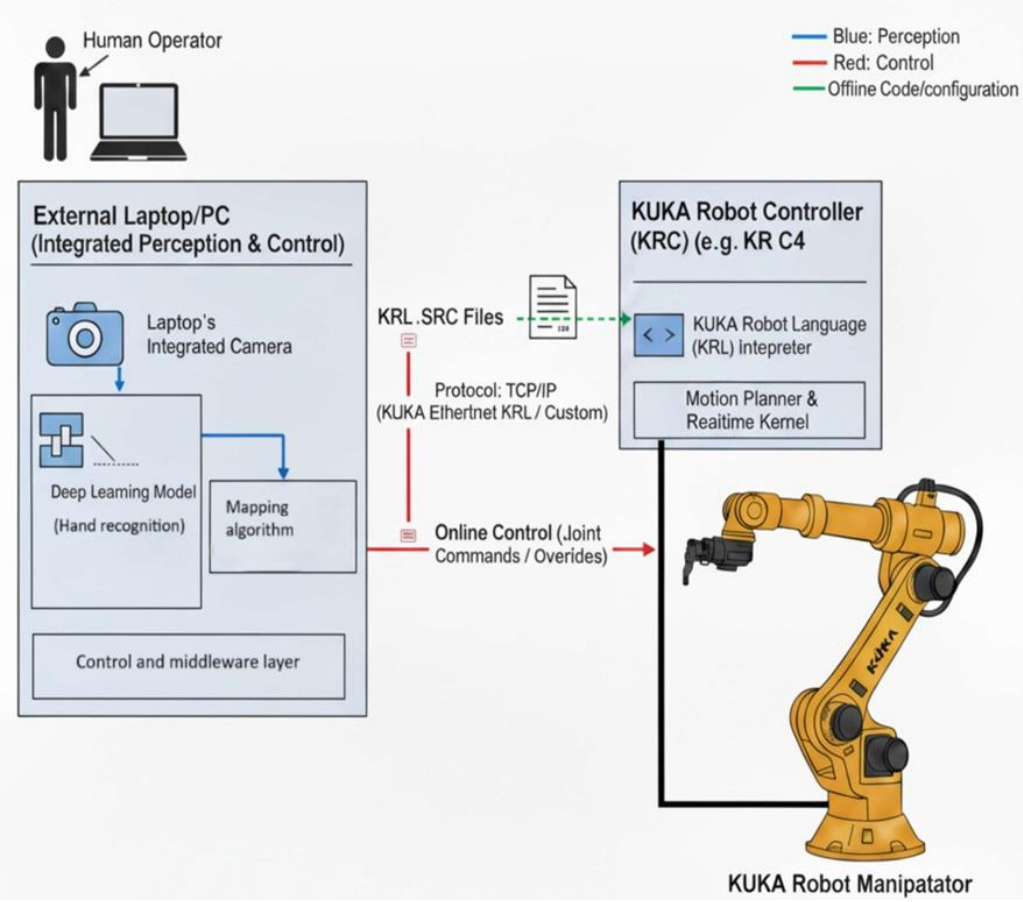

Fig 1. System architecture of AI-driven KUKA framework

Hardware components

The control framework utilizes three core hardware elements: the industrial manipulator, the robot controller, and the perception system.

KUKA KR 16-2 manipulator and controller

The core execution platform for this system is the KUKA KR 16-2 industrial robot, a 6-axis articulated manipulator known for its payload capacity (16 kg) and extensive reach (1610 mm). This model is representative of common industrial applications, providing a robust platform for testing the developed control middleware. The robot is managed by a KUKA robot controller (KRC), typically the KR C4 model, running the proprietary KUKA operating system with a soft real-time kernel. The KRC handles all kinematics, safety functions, and motion planning. External commands are directed to the controller via a dedicated network interface, bypassing the standard teach pendant programming flow to allow for external influence over joint movements and program execution.

Integrated perception system

The human-robot interaction is driven by an integrated visual perception system. This consists of the integrated camera on the external laptop/PC, which provides a live RGB video stream of the human operator's workspace. The camera's feed is processed directly by the external laptop/PC, which hosts the DL-models. This integrated setup offers a portable and cost-effective solution for motion capture, eliminating the need for dedicated, high-cost external depth sensors or specialized Vicon systems. The primary function of this hardware is to reliably deliver video data at a sufficient frame rate to the software layer, ensuring low-latency gesture recognition for the soft real-time control loop.

Software environment and middleware

The control and perception systems are established across three distinct software layers: the deep learning stack for perception, the custom Python middleware for control logic, and the KRL environment on the robot controller. Figure 2 shows the detailed data flow.

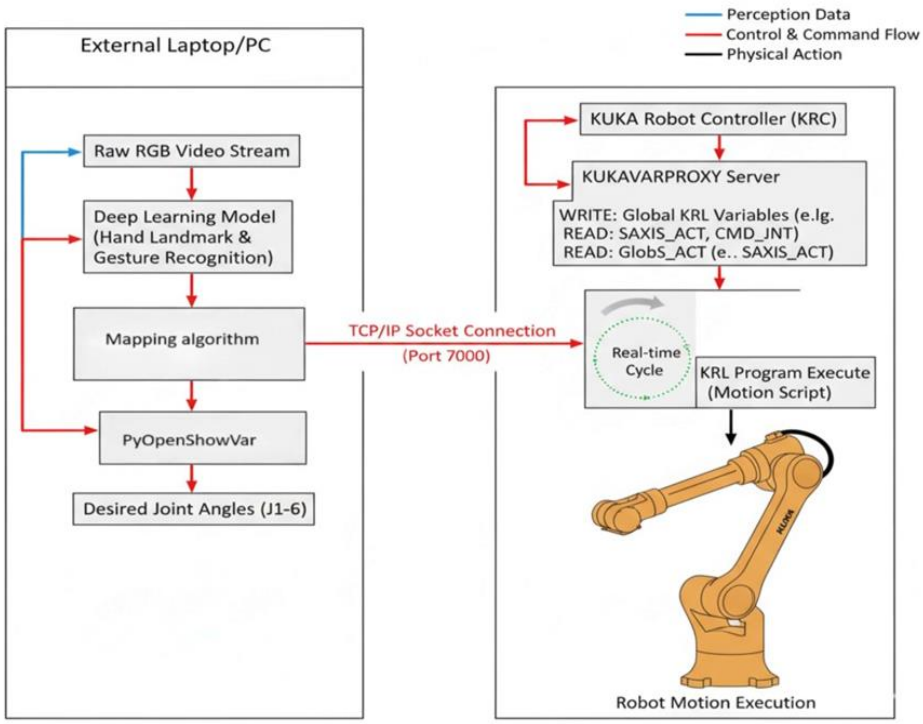

Fig 2. Detailed data flow

Control middleware: KUKAVARPROXY and PyOpenShowVar

The core of the system's external control capability relies on a client-server architecture designed to bypass the limitations of standard KRL programming. On the KUKA robot controller (KRC) side, the open-source KUKAVARPROXY acts as a server. This dedicated application runs on the KRC's Windows environment and establishes a TCP/IP socket connection, typically listening on Port 7000. Its fundamental role is to provide a channel for remote programs to read and write global KRL variables (such as $OV_PRO or dedicated position variables) by implementing the OpenShowVar protocol over the network.

On the external laptop/PC side, the communication client is the PyOpenShowVar library. This Python package handles the low-level network interface, translating the high-level control decisions from the gesture recognition system into the specific messaging format required by the OpenShowVar protocol.

The use of this client-server coupling is essential for achieving soft real-time control and is specifically leveraged for:

– Real-time Joint Overrides: The Python script uses PyOpenShowVar to repeatedly write calculated joint angle increments into pre-declared global KRL variables. A parallel KRL program running on the KRC continuously reads these variables to modify its current motion cycle, effectively enabling the external PC to directly influence the robot's trajectory based on human input.

– State Feedback: The library is used to read back system variables, such as the actual joint position ($AXIS_ACT), closing the control loop and allowing the Python middleware to maintain an accurate model of the robot's state.

– Programmatic Control: PyOpenShowVar allows for remote manipulation of program flow flags and system variables, enabling the user to start, stop, or reset the main KRL execution program based on a recognized hand gesture.

This implementation allows the Python middleware to exert granular control over the robot's motion through variable manipulation, which is the foundational element enabling the AI-driven kinesthetic teaching framework.

3. EXPERIMENTAL SETUP

Description of experimental environment

The experimental validation of the AI-driven control framework was conducted using a KUKA KR 16-2 industrial manipulator controlled by a KRC4

controller. The robot was secured to a standard industrial floor mount within a defined safety cage area.

The perception system utilized the integrated camera on the external laptop/PC, which was positioned approximately 1 meter from the human operator. This arrangement ensured the camera maintained a full-frame view of the operators dominant hand, covering the entire operational range used for gesture control. The robot's primary workspace was configured to be within a safe collaborative zone, and the KRC4 controller and the external laptop/PC were connected via a dedicated Ethernet cable using a static IP configuration for the stable TCP/IP connection.

The experimental validation of the AI-driven control framework was conducted using a KUKA KR 16-2 industrial manipulator controlled by a KRC4 Controller. The robot was secured to a standard industrial floor mount within a defined safety cage area.

The perception system utilized the integrated camera on the external laptop/PC, which was positioned approximately 1 meter from the human operator. This arrangement ensured the camera maintained a full-frame view of the operators dominant hand, covering the entire operational range used for gesture control. The robot's primary workspace was configured to be within a safe collaborative zone, and the KRC4 controller and the external laptop/PC were connected via a dedicated Ethernet cable using a static IP configuration for the stable TCP/IP connection.

Case study I: Discrete position selection

The first case study focuses on validating the discrete gesture recognition capability for fast, programmatic robot control. The system was tasked with recognizing the number of fingers the operator held up (one, two, or three). Based on the recognized count, the robot was commanded to move to a corresponding, pre-determined target position. For example, holding up one finger triggered movement to Position One, two fingers to Position Two, and three fingers to Position Three. This test evaluates the overall latency and reliability of the perception-to-program execution sequence.

Case study II: Continuous control with tracking

The second, more complex case study validates the continuous control and mapping capability. This involved recognizing two primary operational states:

1. Open hand (Continuous control): When the hand was open, the system activated the proportional control mapping, allowing the robot's joints to follow the hand's movements. An open hand moving left, for instance, resulted in the appropriate joint movements toward the left.

2. Depth control: The system also utilized the distance of the hand from the camera (depth, z-axis) to control the robot. Moving the open hand toward or backward from the camera translated into corresponding movements along the robot's principal axis, enabling three-dimensional control over the manipulator's End-Effector or specific joints.

AI model for hand motion tracking

The core of the perception system is a robust, low-latency Deep Learning model responsible for tracking the operator's hand motion in real-time, detecting gestures, and extracting both fine-grained landmarks and categorical gesture IDs.

Dataset and preprocessing

The system utilizes the MediaPipe Hands framework, which employs a highly optimized, pre-trained model for real-time hand detection and tracking. The input video stream from the integrated camera is preprocessed by MediaPipe to the first identify the hand's Region of Interest (ROI) and then predict 21 3D hand landmarks (normalized Px, Py and Pz coordinates). The *z* coordinate provides relative depth information crucial for Case study II. The classification of the discrete gestures (0, 1, 2, or 3 fingers) for Case study I was handled by a lightweight classifier built on top of the raw landmark data, which relied on geometric features such as the distance and relative position between fingertip and joint landmarks.

Model selection and training process

Model selection: The system leverages the pre-trained, pipeline-based model from MediaPipe Hands. This architecture was chosen for its optimal balance between accuracy and extremely high inference speed, which is a non-negotiable requirement for soft real-time robot control. The pipeline consists of a lightweight Palm Detector followed by a dedicated Hand Landmark Model. This design allows the system to achieve an average processing

rate of approximately 30 Frames Per Second (FPS) on the external PC, minimizing overall system latency.

Training process: The underlying hand detection and landmark models are pre-trained. The necessary "training" effort focused on system calibration – empirically tuning the proportional scaling factor k and the coordinate mapping boundaries – and on the lightweight gesture classifier, which was trained using a small, self-collected dataset to validate the geometric thresholds used for the finger-counting task (Case study I).

Evaluation metrics

The validation of the AI perception system was quantified using two primary performance metrics:

1. Gesture Classification Accuracy (ACC gestures). This metric specifically validates the reliability of Case study I: Discrete position selection. It was measured as the percentage of correctly identified discrete gestures across a diverse test set.

2. Detection latency. Reported as the achieved Frames Per Second (FPS). Consistent, high throughsput (30 FPS) is crucial because minimizing the perception delay ensures that the overall control loop remains responsive enough for smooth, intuitive guidance as validated in Case study II.

Real-time data acquisition from hand tracking

The Python middleware acts as the central hub, continuously acquiring and translating data from the AI Perception system. This process is executed at approximately 30 FPS. The control strategy implements a Direct Landmark Differencing approach, leveraging the structured output of Media-Pipe to ensure low-latency control. The wrist position vector, $P_{\text{human}}(t)$, is extracted directly from the Media-Pipe output, consisting of the normalized x, y, and z coordinates of the wrist landmark. The kinematic mapping algorithm calculates the required change in robot joint angles $\Delta\Theta_{\text{robot}}$ using the proportional control law based on the change in the raw, filtered landmark position:

$$\Delta\Theta_{\text{robot}} = k * (P_{\text{human}}(t) - P_{\text{human}}(t-1)).$$

This Direct Landmark Differencing method is superior as it uses the inherent normalized 3D information from Media-Pipe to control the robot along the x, y, and z axes simultaneously. The continuous control loop is active when the "Open Palm" gesture is detected $k = 0.15$.

Data transmission to KUKA robot controller

The calculated commands (either the program flag or the $\Delta\Theta_{\text{robot}}$) are immediately prepared for network transmission to the KRC via the established TCP/IP link. The transmission process utilizes the specialized client-server architecture:

– Client encoding: The PyOpenShowVar library formats the data into the OpenShowVar protocol string.

– TCP/IP transmission: The message is sent over the dedicated Ethernet link to the KUKA-VARPROXY server on the KRC.

– Server execution: Upon receipt, KUKA-VARPROXY instantly writes the data to the predefined global KRL variables, enabling the following control states:

- Discrete position select (Case study I): An integer variable is updated, causing the main KRL program to execute a pre-programmed PTP motion to position 1, 2, or 3.
- Continuous control (Cases study II): The calculated $\Delta\Theta_{\text{robot}}$ values are written to variables that are continuously read by a loop in the KRL program to incrementally adjust the robot's ongoing motion every cycle, achieving soft real-time guidance.
  - Stop / Pause (Closed fist): Sets
    $$\Delta\Theta_{\text{robot}} = 0,$$
    halting all joint velocity.
- Teach / Key-frame save (Four fingers): An instruction flag is sent to the KRL program to record the robot's current joint angles ($AXIS_ACT) to a file for offline program generation.

4. RESULTS AND DISCUSSION

Results from Case study I: Discrete position selection

Case study I validated the system's ability to reliably translate discrete hand gestures into fixed program commands, with the primary metric being Gesture Classification Accuracy. The system's reliability for programmatic command execution was measured across 50 trials for each of the three finger-count gestures (1, 2, and 3 fingers up). The geometric classifier implemented using Media-Pipe landmarks proved highly effective and robust. Table 1 shows the results from testing data for the hand recognition model.

T a b l e 1

Results from testing data for the hand recognition model

Gesture (target)	Number of trials	Correctly classified	Accuracy %
One fingers (Pos 1)	50	49	98
Two fingers (Pos 2)	50	49	98
Three fingers (Pos 3)	50	48	96
Overall	150	146	97.3

The high average accuracy of 97.3% confirms the robustness of the geometric thresholding approach for converting visual input into reliable, discrete operational states.

The total time from the gesture being recognized to the robot initiating its PTP movement was measured to assess the pipeline's overall efficiency. The 200 ms difference represents the combined overhead introduced by the perception system (Media-Pipe frame processing) and the Python middleware's control logic execution. This establishes that the system is suitable for non-critical programmatic moves but highlights the latency component contributed by the soft real-timeAI framework.

Results from Case study II: Continuous kinesthetic control

The system successfully achieved a continuous control loop rate tied to the perception system's throughput of approximately 30 FPS, as shown on Table 2.

T a b l e 2

Perception value and implication for control

Metric	Measured value	Implication for control
Perception	30 FPS	Loop cycle time of ~33 ms (ideal minimum latency).
Proportional scaling factor (k)	0.15	Empirically tuned for safe and intuitive joint velocity.

Qualitatively, the robot's motion was perceived as smooth and responsive for slow-to-moderate hand movements, validating the choice of $k = 0.15$. High-frequency hand movements resulted in noticeable lag and jerkiness, confirming the limitations inherent in the soft real-time communication (TCP/IP) and the KRC's KRL execution cycle.

The proportional control successfully mapped the x and y screen coordinates to the appropriate robot joint movements. Crucially, the z-coordinate (relative depth) provided by MediaPipe was utilized to control axial movement. The result was moving the hand closer to the camera (decreasing z) consistently caused the robot's end-effector to move along its direction of extension and retraction, enabling intuitive forward and backward control. This demonstrates the successful exploitation of Media-Pipe's 3D landmark data for control beyond planar movement.

Discussion of system performance

The system's overall control rate is constrained by the 30 FPS perception throughput and network overhead, placing it firmly in the domain of soft real-time control. The achieved fidelity allows operators to effectively "teach" spatial points and paths by walking the robot through the desired trajectory. For kinesthetic teaching, where intuitiveness and safety are prioritized over microsecond precision, this performance is adequate.

Acritical consideration for the Direct Landmark Differencing approach is the risk of driving the manipulator into kinematic singularities. To ensure robustness and safety, coding solutions were implemented within the KRL program on the KRC:

1. Maximum delta value limit: The program enforced a maximum magnitude on the incremental joint commands ($\Delta\Theta_{\text{robot}}$) received from the PC. This prevents excessive acceleration near singularity zones, capping the maximum velocity command for any single axis.

2. Singularity region detection: The KRL code includes logic to monitor the robot's current joint configuration. If the robot enters a predefined proximity to known singularities (e.g., wrist or shoulder singularities), the KRL program automatically reduces the proportional gain (k) to zero or switches to a position-hold mode. The external PC is then notified with a status message (e.g., "Cannot reach this target, possible singularity").

The implementation of the Direct Landmark Differencing approach proved highly intuitive. Operators did not need to perform complex mental mapping; a change in the hand's position in space directly resulted in a proportional change in the robot's joint velocity. This method:

1. Eliminated system overhead: By avoiding complex inverse kinematics and external OS calls, the control was streamlined.

2. Enabled 3D control: The direct use of MediaPipe's coordinate offers a low-cost, effective method for controlling the third dimension (depth), a significant enhancement over planar 2D vision systems.

The combined results confirm that the low-cost, vision-based framework is a viable and highly accessible alternative for developing natural, kinesthetic teaching interfaces for industrial manipulators.

5. CONCLUSIONS AND FUTURE WORK

This work developed a low-cost, intuitive, and soft real-time kinesthetic teaching interface for industrial manipulators using AI-based vision. The system successfully integrated the MediaPipe Hands model for gesture-based perception and used PyOpenShowVar/KUKAVARPROXY middleware for TCP/IP communication with the KUKA KR 16-2 robot.

Experimental results confirmed the system's effectiveness:

High reliability: The gesture classification system achieved 97.3% accuracy, demonstrating the robustness of the geometric gesture recognition approach.

Intuitive 3D control: The Direct Landmark Differencing method effectively utilized 3D depth data from MediaPipe, enabling natural control of depth (Pz axis).

Precision: During continuous motion, the system maintained end-effector accuracy within one decimal point of the target, meeting typical industrial tolerances.

Soft real-time performance: Despite a 200 ms overhead from AI processing and TCP/IP communication, the system proved suitable for non-time-critical kinesthetic teaching and path recording tasks.

Overall, the proposed framework provides a viable and accessible alternative to expensive commercial solutions, showing that consumer-grade vision systems and open-source middleware can enable natural, expressive robot programming interfaces.

Building on the success and current limitations of the system, several future improvements are proposed:

– Reduce latency with hard real-time control: Migrating from soft real-time TCP/IP to KUKA. RSI (Robot Sensor Interface) with UDP communication could reduce latency and significantly improve motion smoothness.

– Enable Cartesian-space control: Implementing a direct inverse kinematics (IK) solver would allow intuitive control of the robot's end-effector in x, y, z space, overcoming the non-linear behavior of joint-space control.

– Incorporate predictive AI models: To mitigate perception delays, methods like Kalman filtering or RNNs could predict hand positions a few frames ahead, allowing proactive motion and improved responsiveness.

– Develop a full teaching platform: A graphical user interface (GUI) could be added to allow editing, saving, and replaying taught paths, along with features like speed profiles and tool state control – transforming the system into an end-to-end prototyping tool.

– Enhance safety: Expanding the middleware's safety logic to include collision prediction based on 3D workspace mapping (e.g., using RGB-D data) would improve operational safety.

– Expand gesture vocabulary: Adding dynamic gestures for more complex commands (e.g., grip/release, speed control, or mode switching) would allow operators to execute more advanced tasks without using the physical SmartPAD.

These directions aim to refine the system into a robust, flexible, and user-friendly tool for advanced human-robot collaboration and intuitive robot programming.

REFERENCES

[1] Sanfilippo, F., Hatledal, L. I., Zhang, H., Fago, M., Pettersen, K. Y. (2015): Controlling KUKA industrial robots: Flexible communication interface JOpenShowVar. *IEEE Robotics & Automation Magazine*, **22** (4): 96–109.

[2] Elshatarat, H. L., Baniyounis, M., Biesenbach, R. (2016, March): Design and implementation of a RoBO-2L MATLAB toolbox for a motion control of a robotic manipulator. In: *13th International Multi-Conference on Systems, Signals & Devices (SSD),* pp. 89–95, IEEE.

[3] Mehner, M., Matkovic, N., Muehlbeier, E., Mayer, D., Fleischer, J., Verl, A. (2023, September): Evaluation of external control of KUKA industrial robots for laboratory and prototype environments. In: *ISR Europe 2023; 56th International Symposium on Robotics,* pp. 278–284. VDE.

[4] Bilancia, P., Schmidt, J., Raffaeli, R., Peruzzini, M., Pellicciari, M. (2023): An overview of industrial robots control and programming approaches. *Applied Sciences*, **13** (4), 2582.

[5] Abdeetedal, M., Kermani, M. R. (2019): An open-source integration platform for multiple peripheral modules with KUKA robots. *CIRP Journal of Manufacturing Science and Technology*, **27**, 46–55.

[61 Shang-Liang, C., Li-Wu, H. (2021): Using deep learning technology to realize the automatic control program of robot arm based on hand gesture recognition. *International Journal of Engineering and Technology Innovation*, **11**(4), 241.

[7} De Smedt, Q. (2017): *Dynamic hand gesture recognition – From traditional handcrafted to recent deep learning approaches* (Doctoral dissertation, Université de Lille 1, Sciences et Technologies; CRIStAL UMR 9189).

[8] Akgun, B., Cakmak, M., Yoo, J. W., Thomaz, A. L. (2012, March): Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In: *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction,* pp. 391–398.

[9] Foteinos, K., Cani, J., Linardakis, M., Radoglou-Grammatikis, P., Argyriou, V., Sarigiannidis, P., Varlamis, I., Papadopoulos, G. T. (2025): Visual hand gesture recognition with deep learning: a comprehensive review of methods, datasets, challenges and future research directions. DOI:10.48550/arXiv.2507.04465.