# Ordimus Bot: Pick and Place Robot

Arjun Haridas
*Department of Computer Science and Engineering*
*Sardar Vallabhbhai National Institute of Technology*
Surat, India
u23cs089@coed.svnit.ac.in

Mehulkrishna
*Department of Computer Science and Engineering*
*Sardar Vallabhbhai National Institute of Technology*
Surat, India
u23cs087@coed.svnit.ac.in

Dany Binu Luke
*Department of Computer Science and Engineering*
*Sardar Vallabhbhai National Institute of Technology*
Surat, India
u23cs008@coed.svnit.ac.in

Sanjana Kozhipurath
*Department of Computer Science and Engineering*
*Sardar Vallabhbhai National Institute of Technology*
Surat, India
u23cs002@coed.svnit.ac.in

***Abstract*—This paper presents Ordimus Bot, a gesture-controlled 6-DOF robotic arm system designed for pick-and-place operations using intuitive hand gesture recognition. The system integrates MediaPipe Hands for real-time gesture detection with Arduino-based servo control to enable seamless human-robot interaction. A graphical user interface built with PySimpleGUI provides dual control modes including direct slider manipulation and gesture-based control. Hand landmarks captured via webcam are mapped to servo angles ranging from 0 to 180 degrees, allowing users to control the robotic arm through natural hand movements or manual slider adjustments. A teaching and playback mode allows the robot to record and repeat complex trajectories through pose sequencing, making the platform suitable for repetitive industrial tasks and education. Six distinct gestures control different arm functions including pinch for gripper, three-finger for pronation-supination, peace for flexion-extension, point for elbow, open hand for shoulder, and fist for base rotation. Experimental evaluation demonstrates 90% gesture recognition accuracy and 88% pick-and-place task success rate with end-to-end latency of 160 to 180 milliseconds. The system currently operates at Technology Readiness Level 4, validated in laboratory environments with controlled lighting and object constraints.**

***Index Terms*—Robotic arm, gesture recognition, MediaPipe, computer vision, human-robot interaction, pick-and-place, teaching and playback, Arduino, servo control, GUI**

## I. Introduction

Robotic arms are widely used in industries for automation tasks such as pick-and-place operations, assembly, and material handling. Traditional robotic systems rely on pre-programmed sequences or complex control interfaces requiring specialized training. However, there exists a significant gap for intuitive gesture-based control systems that allow users to manually guide robotic arms for pick-and-place tasks in real-time.

The motivation behind this work is bringing human intelligence and robotic precision together through gesture recognition, which enhances adaptability and bridges the gap between manual and fully automated systems. By enabling natural hand gesture control alongside graphical slider-based control, operators can quickly program and adjust robotic tasks without extensive programming knowledge, making automation more accessible.

Recent advances in computer vision and machine learning have enabled accurate real-time hand tracking on standard hardware. MediaPipe Hands, developed by Google, provides a robust framework for hand landmark detection with minimal computational requirements. Combined with affordable Arduino microcontrollers and servo-based actuation systems, cost-effective gesture-controlled robotic platforms become feasible for educational and small-scale industrial applications.

This paper presents the design, implementation and experimental evaluation of Ordimus Bot, a 6-DOF robotic arm controlled through hand gestures and a graphical user interface. The system architecture implements a sense-think-act control loop where webcam input enables gesture sensing, Python-based processing with PySimpleGUI handles mapping logic and user interaction, and Arduino firmware controls servo actuators. A key capability is the teaching-and-playback mode enabling users to record motion sequences as pose keyframes for automated repetition. This feature allows operators to manually demonstrate a task once through gesture control or GUI sliders while the system captures joint angles. The recorded trajectory can then be executed repeatedly with high consistency, eliminating the need for manual programming of complex motion paths. This approach combines the flexibility of human demonstration with the precision and repeatability of automated execution, making the system particularly valuable for small-batch production and educational environments where task requirements frequently change.

The system currently stands at Technology Readiness Level 4 where the integrated prototype has been successfully tested and validated in a laboratory environment for gesture-based and GUI-controlled real-time robotic arm control. The primary contributions include development of an intuitive gesture vocabulary, implementation of a responsive real-time control pipeline with graphical interface, pose sequencing capabilities with JSON import and export functionality, and comprehensive performance analysis under various operating conditions.

## II. Related Work

Vision-based gesture recognition for robotic control has received significant research attention. Oudah et al. presented a comprehensive review of hand gesture recognition based on computer vision, covering various approaches from traditional feature extraction to deep learning methods [3]. Their work highlighted the evolution toward more robust real-time systems suitable for human-robot interaction.

Malassiotis et al. explored vision-based hand gesture recognition specifically for human-robot interaction contexts [4]. Their research demonstrated the feasibility of using visual gestures as a natural interface for robot control, though computational constraints limited real-time performance at that time.

More recently, Michels presented real-time hand gesture control of a robotic arm using MediaPipe [1]. This work demonstrated recording hand-guided motions and replaying them for pick-and-place tasks, which directly inspired our teaching-and-playback approach. Angelidis extended gesture-controlled robotic arms to small assembly line applications [2], showing practical industrial potential.

MediaPipe Hands represents a significant advancement in hand tracking technology [5]. The framework employs a two-stage pipeline with palm detection followed by hand landmark localization. The efficient architecture enables real-time performance on mobile and embedded devices without requiring specialized accelerators.

Our work builds upon these foundations by implementing a complete gesture-controlled robotic arm system using readily available components. We adapted the teaching-and-playback concept and implemented it with Arduino microcontroller combined with PCA9685 servo driver for precise multi-channel control. The system emphasizes practical deployment considerations including lighting sensitivity, latency analysis, and repeatability assessment, while providing dual control modes through gesture recognition and graphical slider interface.

## III. System Design

### A. Problem Statement and Objectives

The primary objective is to design and develop a robotic arm capable of performing pick-and-place operations with precision and accuracy through gesture-based control and graphical user interface. Specific goals include creating a responsive control system enabling seamless human-robot interaction through natural hand movements and manual slider controls, implementing real-time hand gesture recognition using computer vision techniques for intuitive manual control, and providing a comprehensive graphical interface for direct servo manipulation, pose sequencing, and trajectory playback.

### B. System Assumptions

The system operates under several key assumptions to ensure reliable performance. Objects handled are lightweight and suitable for servo-based gripper actuation. The robotic arm is placed on a flat stable surface. Adequate lighting conditions are required for reliable hand gesture detection. A stable camera position with unobstructed view of the user's hand is maintained. Stable USB connection enables real-time serial communication between control computer and Arduino with default baud rate of 9600. Sufficient power supply supports simultaneous operation of all six servo motors.

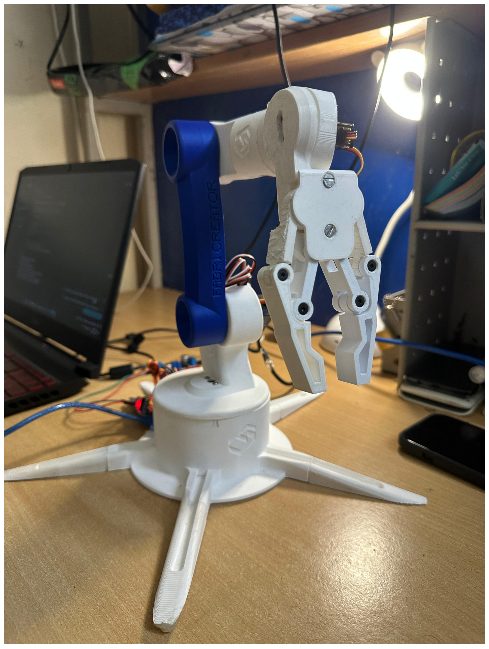


Fig. 1. Assembled robotic arm with 3D printed structure and servo control system

### C. Hardware Components

The mechanical structure comprises a 6-DOF robotic arm fabricated from 3D printed components using publicly available STL files from the Fabri Creator Robotic Arm 4.0 design [8]. The arm incorporates three SG90S metal gear servos and three MG995 metal gear servos positioned to control base rotation, shoulder elevation, elbow flexion, wrist movements, pronation-supination, and gripper actuation.

Power is supplied through a 5V 3A DC adapter with appropriate voltage regulation. An Arduino UNO microcontroller serves as the primary control unit. The Adafruit PCA9685 16-channel 12-bit PWM servo controller [7] interfaces between Arduino and servos, enabling precise simultaneous multi-servo control while offloading PWM generation from the Arduino.

Mechanical assembly utilizes M3 screws in 12mm and 16mm lengths, M3 washers, self-locking M3 nuts, and Allen keys for assembly. Servo cable extensions provide routing flexibility. Various jumper cables facilitate electrical connections. A standard USB webcam captures hand gestures for input.

### D. Software Architecture

The software system is implemented primarily in Python leveraging several key libraries. PySimpleGUI provides a user-friendly graphical interface featuring connection controls, real-time servo position sliders with custom labels for each

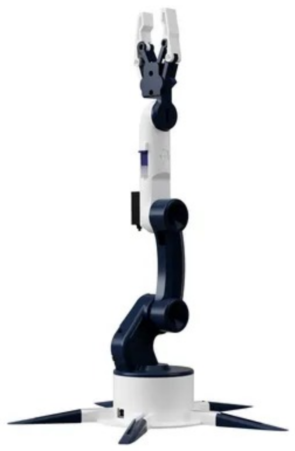


Fig. 2. 6-DOF robotic arm showing base rotation, shoulder, elbow, wrist articulation with pronation-supination and flexion-extension, and gripper mechanism.

joint, pose saving and sequencing capabilities, and comprehensive logging functionality. The GUI implements a DarkBlue3 theme with split-pane layout displaying connection status, reference image, six individual servo sliders with labels for Elbow, Shoulder, Pronation-Supination, Flexion-Extension, Gripper, and Base, along with control buttons for Send All, Home, Save Pose, Run Sequence, and Clear operations.

OpenCV [6] handles real-time video capture and image preprocessing including frame capture, horizontal flipping for mirror-mode interaction, resizing, and color conversion from BGR to RGB format before processing.

MediaPipe Hands [5] performs core gesture recognition through palm detection and extraction of 21 hand landmark coordinates representing fingertips, joints, and palm center. These landmarks enable gesture classification including fist, open hand, point, peace, and pinch configurations.

A custom gesture-to-servo mapping module translates recognized gestures into servo control commands. Hand coordinates map to servo angles within 0 to 180 degree range using slider controls in the graphical user interface. Serial communication via USB with configurable baud rate establishes bidirectional connection between Python application and Arduino for command transmission formatted as servo index followed by angle value with newline terminator.

The pose sequencing system enables recording of arm configurations as timestamped keyframes. Each pose captures all six servo positions and can be stored in memory as a list. Sequences can be exported to JSON format for persistence and imported for later playback. During sequence execution, configurable delay between poses allows control over motion speed.

Arduino firmware implements low-level servo control, receiving position commands over serial communication in format servo index followed by angle, parsing command strings, and updating servo positions through the PCA9685 interface. The firmware handles initialization, calibration, and enforces safety limits within the 0 to 180 degree range.

## IV. Methodology

### A. Sense-Think-Act Control Loop

The system operates according to a sense-think-act paradigm enabling real-time responsive behavior. In the Sense phase, the webcam continuously captures video frames while MediaPipe detects hand presence and extracts landmark coordinates. Gesture classification algorithms analyze spatial relationships between landmarks to identify current hand configurations. Alternatively, users interact with GUI sliders to directly specify desired servo positions.

The Think phase involves two operational modes. In live control mode, hand coordinates or slider values directly map to servo angles for immediate response. Specific gestures map to robot functions such as pinch controlling gripper closure. In playback logic mode, servo angles for each pose are recorded as keyframes with user-configurable timing delays enabling later automatic execution.

The Act phase transmits computed servo commands from Python to Arduino over serial interface using protocol format of servo index concatenated with angle value. Arduino updates servo positions through PCA9685, moving robotic arm joints to commanded positions. During playback mode, Arduino executes stored keyframe sequences to replay recorded trajectories with specified inter-pose delays.

Feedback mechanisms provide situational awareness through the graphical interface displaying detected gestures, current servo angles via both sliders and input boxes, system status, and comprehensive event logging. During playback, progress indicators show execution status and completed steps enabling verification and intervention if needed.

### B. Gesture Recognition and Mapping

Six primary gestures control different aspects of arm motion. Pinch gesture, detected when thumb and index fingertips approach, controls gripper with fingertip distance mapping to gripper opening angle. Three-finger configuration with thumb, index and middle fingers extended controls pronation-supination wrist rotation. Peace gesture with index and middle fingers extended controls flexion-extension wrist motion. Point gesture with only index finger extended controls elbow joint. Open hand configuration with all fingers extended controls shoulder joint. Fist gesture with all fingers closed controls base rotation.

This gesture vocabulary was selected for distinctiveness to minimize classification errors and intuitiveness such that gestures naturally correspond to control functions. The mapping is fully customizable through the user interface sliders and input boxes.

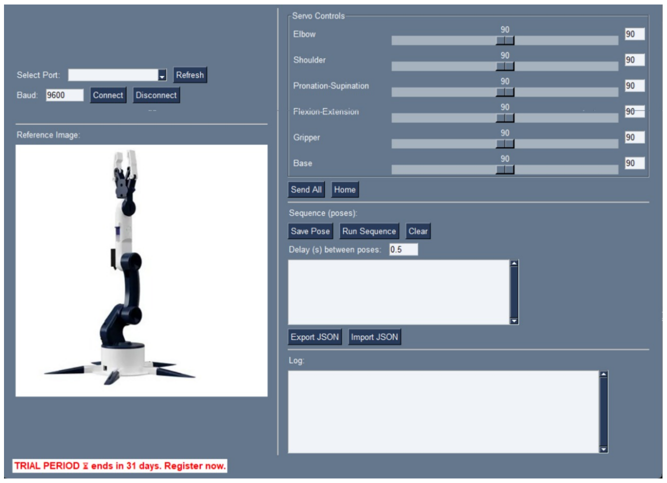
### C. Graphical User Interface Features

The PySimpleGUI-based interface provides comprehensive control capabilities. The connection panel enables COM port selection with refresh functionality, configurable baud rate input defaulting to 9600, and connect and disconnect buttons with status display. A reference image panel displays the robotic arm schematic for user guidance.

The servo control panel features six horizontal sliders ranging from 0 to 180 degrees with real-time position update, custom descriptive labels for each servo joint, synchronized input text boxes for precise numerical entry, and event-driven updates that immediately transmit position changes to Arduino. Additional control buttons include Send All to simultaneously update all servos, and Home to reset all joints to 90 degree neutral position.

The sequence management panel provides Save Pose functionality to capture current arm configuration, Run Sequence to execute recorded poses with configurable inter-pose delay, Clear to reset sequence memory, and a listbox displaying all saved poses. Import and Export buttons enable JSON-based sequence persistence for reusable motion patterns.

A comprehensive logging window displays all system events including connection status changes, transmitted servo commands, pose save operations, sequence execution progress, and error messages providing full operational transparency.



Baud: 9600

Elbow 90
Shoulder 90
Pronation-Supination 90
Flexion-Extension 90
Gripper 90
Base 90

Delay (s) between poses: 0.5

TRIAL PERIOD ⌛ ends in 31 days. Register now.

Fig. 3. PySimpleGUI-based control interface showing connection panel with COM port selection, reference image display, six servo control sliders with custom labels (Elbow, Shoulder, Pronation-Supination, Flexion-Extension, Gripper, Base), pose sequencing controls with JSON import/export functionality, and comprehensive logging window.

### D. Hardware Setup and Integration

The 6-DOF robotic arm is assembled with servo motors connected to Arduino through the PCA9685 servo controller for precise joint control. Mechanical components are secured using appropriate fasteners ensuring rigid connections. Electrical connections route power and control signals properly. The system is calibrated to establish servo angle ranges and neutral positions for each joint defaulting to 90 degrees home position.

### E. Pick-and-Place Execution

Pick-and-place operations utilize the pose sequencing and playback system to automate repeated tasks. The robot is first manually guided through desired motion using gesture control or GUI sliders while operator saves key poses at critical waypoints. During playback, the system interpolates between saved poses executing each servo position with small delays between commands to ensure smooth motion, and waiting for the user-specified delay before advancing to the next pose. This allows automatic repetition of the demonstrated movement without further human input, with all sequences exportable to JSON format for later reuse or modification.

### F. Modules Used

The implementation integrates three key modules. Kinematics module uses inverse kinematics principles for pick-and-place motion planning. Robotic vision module especially OpenCV provides visual processing pipeline. AI module handles gesture recognition and interpretation using MediaPipe's deep learning models. GUI module implemented with PySimpleGUI provides comprehensive human-machine interface for direct control, monitoring, and sequence management.

## V. Experimental Results

Comprehensive testing was conducted to evaluate system performance across multiple metrics under laboratory conditions.

### A. Prototype Validation

The prototype successfully performed real-time gesture-controlled and GUI-controlled pick-and-place operations with high reliability under laboratory conditions with controlled lighting and object placement. The system demonstrated responsive behavior and intuitive control through both interaction modalities.

### B. Gesture Recognition Accuracy

Overall gesture recognition accuracy achieved approximately 90 %, enabling smooth and intuitive human-robot interaction. This high accuracy supports reliable control across the six gesture vocabulary. However, recognition performance showed sensitivity to environmental factors.

Gesture detection performance varied significantly with lighting conditions. Under bright lighting, accuracy reached 94%. Normal indoor lighting yielded 88% accuracy. Poor lighting conditions reduced accuracy to 72%, highlighting sensitivity to illumination. These results emphasize the importance of adequate lighting for reliable gesture recognition.

### C. Pick-and-Place Task Performance

Pick-and-place task success rate measured 88% over 25 trial attempts. Failures primarily resulted from minor gripper misalignment and object slipping rather than gesture recognition errors. This demonstrates that mechanical precision represents a key limitation for task reliability.

### *D. System Latency*

End-to-end system latency was measured at approximately 160 to 180 milliseconds from gesture execution or slider adjustment to robotic arm response. This includes webcam frame capture or GUI event processing, MediaPipe processing or command formatting, serial command transmission with 20 millisecond inter-command delay, and servo movement time. The latency results in near real-time motion response without noticeable delay for typical manipulation tasks. While perceptible, this latency remains acceptable for non-time-critical operations. GUI-based slider control exhibited slightly lower latency of approximately 140 to 160 milliseconds due to elimination of gesture recognition processing overhead.

### *E. Playback Mode Repeatability*

Playback mode testing evaluated trajectory repeatability over multiple cycles using recorded pose sequences. Initial playback showed high fidelity to recorded motions with configurable inter-pose delays providing smooth transitions. However, servo drift accumulated over repeated cycles. After 10 playback cycles, servo position drift measured plus or minus 3 degrees from recorded values. By the 50th cycle, drift increased to plus or minus 8 degrees. This degradation results from servo heating, mechanical backlash, and lack of position feedback sensors. JSON export and import functionality maintained sequence integrity across system restarts with no data loss observed.

## VI. Challenges

Several technical challenges were encountered providing insights for future improvements.

### *A. Error Accumulation Over Cycles*

Small servo inaccuracies accumulated during repeated playback cycles causing gradual path drift. This results from mechanical backlash in servo gears, servo deadband characteristics, and absence of encoder feedback. After multiple repetitions, actual arm position diverged noticeably from intended trajectory compromising repeatability for high-precision applications.

### *B. Hand Tracking Jitter*

Frame-to-frame landmark noise caused small fluctuations in computed servo angles even when the hand remained relatively stationary. This tracking jitter introduced oscillations in servo commands affecting motion smoothness. While software filtering and GUI slider control partially mitigated this issue, residual jitter remained perceptible during gesture-based operation.

### *C. No Real Depth Estimation*

The monocular camera system lacks true depth perception, relying on hand size heuristics for distance estimation. Gesture interpretation changed with user distance from camera requiring users to maintain consistent working distance. Varying distance introduced ambiguity in gesture-to-motion mapping affecting control precision. GUI slider control avoided this limitation entirely.

### *D. Lighting and Background Sensitivity*

Poor lighting or cluttered backgrounds reduced landmark accuracy and gesture reliability. MediaPipe performance degraded under insufficient illumination with reduced landmark detection confidence. Background objects with skin-like colors triggered false detections. Controlled lighting and uncluttered backgrounds were necessary for reliable gesture operation. GUI control mode remained unaffected by lighting conditions.

### *E. Servo Drift and Heating During Playback*

During extended playback sessions, servos did not always return to exact recorded angles due to mechanical backlash and friction leading to small deviations in arm trajectory. Continuous operation caused servo heating potentially affecting holding torque and positioning accuracy. Intermittent cooling periods were required during prolonged testing sessions.

## VII. Applications

The gesture-controlled robotic arm system has potential applications across several domains.

### *A. Industrial Automation*

The system is widely applicable for assembly, welding, painting, and material handling operations due to high precision and repeatability. Gesture control and GUI-based programming enable operators to quickly reprogram tasks without specialized programming knowledge, particularly valuable in small-batch production or custom manufacturing requiring flexibility. The JSON-based sequence export enables sharing of motion programs between multiple systems.

### *B. Pick-and-Place Operations*

Fast and reliable object handling suits packaging, sorting, and manufacturing line applications. The teaching-and-playback capability with pose sequencing supports rapid task reconfiguration. The intuitive dual-mode interface reduces operator training requirements in logistics and warehouse operations.

### *C. Laboratory Automation*

In research laboratories, the system can assist with sample handling, pipetting, and repetitive experimental tasks to improve consistency. Recording and replaying manipulation sequences through saved poses improves experimental reproducibility while freeing researchers from tedious manual procedures.

### *D. Service and Assistive Robotics*

Applications include rehabilitation, elderly assistance, and human-robot interaction tasks. Gesture control provides intuitive interfaces for individuals with mobility impairments. Potential uses include object retrieval, feeding assistance, and household item manipulation enhancing user independence.

## VIII. Future Work

Several enhancements are planned to address current limitations and expand capabilities.

### A. Raspberry Pi Integration

Migrating computer vision, control logic, and GUI to Raspberry Pi would eliminate dependence on external PC creating a compact standalone system. This improves portability and reduces cost while maintaining adequate computational performance for real-time gesture recognition and graphical interface rendering.

### B. Gripper Modification and Advancement

The current gripper has limited adaptability to object shapes. Future versions will incorporate force-sensitive or adaptive grippers enabling secure grasping of wider object ranges with varying geometries and fragility. Tactile feedback integration would prevent object damage during grasping.

### C. Webcam Upgradation for Autonomous Pick-and-Place

Implementation of webcam-based object recognition and localization would enable autonomous pick-and-place without manual guidance for every operation. The robot could automatically locate target objects, plan approach trajectories, and execute grasping based on visual feedback.

### D. Better Motors and Circuitry

Integrating higher-quality motors with encoders would significantly improve positioning accuracy enabling closed-loop control. Better motor drivers would reduce electrical noise and jitter. Addressing mechanical backlash through improved component precision would enhance repeatability and eliminate random jitters and servo current pullback issues.

### E. Enhanced GUI Features

Future GUI enhancements could include real-time 3D visualization of arm configuration, graphical trajectory editing, multiple sequence management with named presets, and integration of computer vision feedback directly into the interface.

### F. Collaborative Robot Features

The system could be upgraded to collaborative robot capabilities to assist in improved pick-and-place and other industrial or service applications. This includes force sensing, compliance control, and safety monitoring enabling safe human-robot interaction in shared workspaces.

## IX. Conclusion

This paper presented Ordimus Bot, a cost-effective gesture-controlled 6-DOF robotic arm system that enables intuitive human-robot interaction through dual control modalities. The key contribution is the integration of MediaPipe-based gesture recognition with a PySimpleGUI interface, providing both natural hand gesture control and precise manual adjustment capabilities within a unified platform.

The system achieved 90% gesture recognition accuracy and 88% pick-and-place task success rate with near real-time response latency of 140 to 180 milliseconds depending on control mode. The pose sequencing feature with JSON-based trajectory storage enables operators to record, save, and replay complex manipulation tasks, making the system particularly suitable for educational environments and small-batch manufacturing where task flexibility is essential.

Experimental validation at Technology Readiness Level 4 revealed important practical considerations including environmental sensitivity of vision-based gesture recognition, mechanical limitations from servo backlash, and trajectory drift during extended operation. These findings provide concrete directions for system refinement.

The demonstrated feasibility of combining affordable components (Arduino, standard servos, webcam) with accessible software tools (Python, MediaPipe, PySimpleGUI) shows that intuitive robotic control interfaces can be developed without significant investment in specialized hardware or proprietary software. This accessibility makes gesture-controlled robotics viable for educational institutions and small enterprises. Future work focusing on embedded processing, improved actuation, and autonomous vision capabilities will enhance the system's practical utility for industrial and service robotics applications.

## References

[1] D. G. Michels, "Real-Time Hand Gesture Control of a Robotic Arm using MediaPipe," International Journal of Robotics Research, 2024.

[2] G. Angelidis, "Gesture-Controlled Robotic Arm for Small Assembly Lines," MDPI Machines Journal, 2025.

[3] M. Oudah, A. Al-Naji, and J. Chahl, "Hand Gesture Recognition Based on Computer Vision: A Review," Human-centric Computing and Information Sciences, 2020.

[4] F. Malassiotis, "Vision-Based Hand Gesture Recognition for Human-Robot Interaction," IEEE Trans. Systems, Man, and Cybernetics, 2019.

[5] "MediaPipe Documentation - Hand Tracking Pipeline and Landmark Model," Google AI, 2023.

[6] "OpenCV Python Documentation," OpenCV.org, 2024.

[7] "Adafruit PCA9685 16-Channel 12-bit PWM/Servo Driver - Technical Datasheet," Adafruit Industries, 2023.

[8] Fabri Creator, "Robotic Arm 4.0 STL Files and Assembly Guide," Cults3D Repository, 2024.