Vision-Based Control of a Mobile Platform Using Edge Artificial Intelligence

Mecanum platform Case Study

Isma AKLI [(1)*], Asma AMRANI[(2)], Manel AIT YALA[(3)], Inas Oumnia MAIZI[(3)] , Rihab TOUMI[(2)]
[(1)] Centre for Development of Advanced Technologies (CDTA), Baba Hassen, Algiers, Algeria.
[(2)] Saad Dahleb University (USDB), Blida, Algeria.
[(3)] University of Science and Technology Houari Boumedien, (USTHB), Algiers, Algeria.
*Corresponding author : iakli@cdta.dz

Abstract: This paper presents a vision-based motion generation system for mobile platforms. The robot is controlled through human hand gestures. A lightweight camera captures video streams from real-world environments, while an artificial intelligence-based gesture recognition model identifies the operator's hand gestures and generates the corresponding motion commands. The proposed solution is implemented and validated on a real Mecanum mobile platform. Two lightweight artificial intelligence models (Faster Objects, More Objects and MobileNetV1) for hand gesture recognition are embedded and evaluated on Expressive Systems ESP32 microcontrollers**.**

Keywords: Mobile platform, Embedded systems, Edge Artificial Intelligence, Hand gesture, Robotic vision.

1. INTRODUCTION

Mobile robotics is founded on several fundamental fields, including perception, navigation, and cognition [1].

Mobile robots, equipped with dedicated locomotion mechanisms, are capable of moving in multiple directions while perceiving their surroundings through various sensory systems. Among these systems, robotic vision plays a central role in enabling autonomous behavior. Recent advances in artificial intelligence (AI), machine learning algorithms, and camera technologies have significantly improved the reliability and practicality of robotic vision systems for real-world mobile robot applications. As a result, robotic systems are becoming increasingly autonomous, intelligent, and efficient [2].

Computer vision has enabled a wide range of robotic applications, including object detection and recognition [3], object pose estimation [4], inspection and quality control [2], as well as human–robot interaction [5]. Visual information can also support many technical functionalities, such as, path planning [6]. In addition, vision data might be combined with other sensing technologies, such as RFID systems [7], to further enhance the control, perception, and autonomy of robotic platforms. Altogether, these developments illustrate the evolution of robotic vision from simple sensing approaches toward more intelligent and context-aware perception systems.

The AI models have become fundamental tools in computer vision, enabling machines to perform complex visual perception tasks. Among the different possible applications, gesture recognition was selected in this work due to its importance in natural and intuitive human-robot interaction.

This article presents a solution enabling the control a Mecanum drive mobile platform through a vision-based gesture recognition module. Two AI models are embedded on a low-power microcontroller, to recognize American Sign Language (ASL).

This paper is organized as follows: Section 2 provides an overview of related works in connection with the presented topic. Section 3 describes the system architecture and methodology adopted to deal with the presented issue. Section 4 describes the experimental setup. Section 5 presents some pertinent results.

2. RELATED WORKS

Human hand gestures can be recognized through different kinds of sensors with different technics [8]. In this article, a vision sensor supplemented with an AI algorithm is used to control the movement of a mobile platform.

Hand Gesture recognition with Vision based AI methods

Gesture recognition based on computer vision is a topic widely discussed in literature. Zaman et *al*. [9] designed a teleoperation control system for a Mecanum wheeled mobile robot. Hand gesture recognition strategy was adopted. Five AI models as: GoogLeNet, VGG19, SqueezeNet, AlexNet, MobileNetV2 were implemented. Fuad et *al.* [10] developed a body gesture recognition system for human robot communication. Convolutional Neural Network (CNN) algorithm was implemented. A dataset composed of 450 images was used for training the AI model. Laplaza et *al.* [11] proposed a gesture-based algorithm for human robot interaction. MediaPipe tool extracts the full body skeleton of a human operator. Neural networks are used for human gesture recognition. Static and dynamic gestures are considered. Sahoo et *al.* [12] deployed a vision based CNN model for real-time hand gesture recognition. Finger information are emphasized through an attention multiscale representation mechanism. American sign language finger-spelling (ASL-FS) dataset estimates the performance of the system. Chen et *al.* [13] developed a gesture recognition system using a LeapMotion sensor for a mobile robot control. A GestureMoRo algorithm associating human gestures recognition and mobile robots were designed. Weng et *al.* [14] integrated a visual system with hand gesture recognition algorithm for human-robot interaction. The visual-perception system was implemented on a dual-arm mobile robot. The YOLO (You Only Look Once) trained AI model identifies the object to be tracked. Suryavanshi et *al.* [15] proposed proof of concept of hand gesture guided wheeled mobile robot prototype. Data fusion from sensors mounted on human hand created a scheme for motor control of the wheels.

Mobile robots and edge vision

Many Hard/Soft ways are possible for vision-based control of a mobile platform. Indeed, the AI algorithm could be implemented on a distant computer for teleoperated control, or embedded on the robotic system. Mahmud et al [16] used computer vision algorithm for controlling an autonomous fire-fighting robot. The proposed solution based on an ESP32-CAM microcontroller utilizing a stereoscopic vision system. The FOMO (Faster Objects, More Objects) image classifier detects in real-time fires. Gholami et *al.* [17] designed a human detection robot. The YOLO (You Only Look Once) framework was deployed on the STM32MP157F microprocessor for security and surveillance scenarios. Galarza-Falfan et *al.* [18] integrated an Odometry and artificial vision based on deep learning (DL) methods for Simultaneous Localization and Mapping implemented on a Mobile Robot. The algorithms were deployed on a Jetson Nano embedded system. The ResNet18 and YOLOv3 algorithms were selected for pattern recognition tasks. Wang et al [19] collected a thermal imaging gesture dataset and implemented an AI lightweight model on a microcontroller STM32 through STM32CubeMX6.9.2. The gesture based controller of the robot is based on thermal imaging sensors. Ikegami et *al.* [20] proposed a user's hand gesture detection method for mobile robot control. The hand is detected by comparing skin colors with user's face. The gesture is detected by calculating the movement of hand gravity center. Four classes of gestures are considered to command robot. Raspberry pi camera with a Raspberry pi 3 Model are used.

This paper investigates the use of computer vision and AI for controlling the motion of a mobile platform. In this work, FOMO and MobileNetV1 AI models are comparatively evaluated on an ESP32-S3. The selected models are then integrated into an ESP32-CAM-based vision and control architecture and experimentally deployed on a real Mecanum-wheeled mobile platform. This approach enables the assessment of both the recognition capabilities of the two models and their practical suitability for embedded robotic applications. The proposed system uses a mobile platform capable of interpreting American Sign Language (ASL) gestures through a lightweight camera, enabling users to control the robot using hand gestures without the need for speech or conventional input devices such as keyboards.

3. SYSTEM ARCHITECTURE AND METHODOLOGY

Problem Statement

For controlling mobile robots using vision sensors, one of the main challenges lies in designing an appropriate hardware and software architecture. Vision sensors should be lightweight and embedded to facilitate their integration into mobile platforms. In addition, an efficient communication framework must be implemented to ensure the reliable transmission of visual data acquired by the sensors. The overall system should also

incorporate AI models to provide the robot with perception and decision-making capabilities. These capabilities enable the robot to interpret its environment and make intelligent decisions, thereby enhancing its autonomy and operational efficiency. The main components and requirements of a vision-based hand gesture-guided control system for mobile platforms are illustrated in Fig. 1.

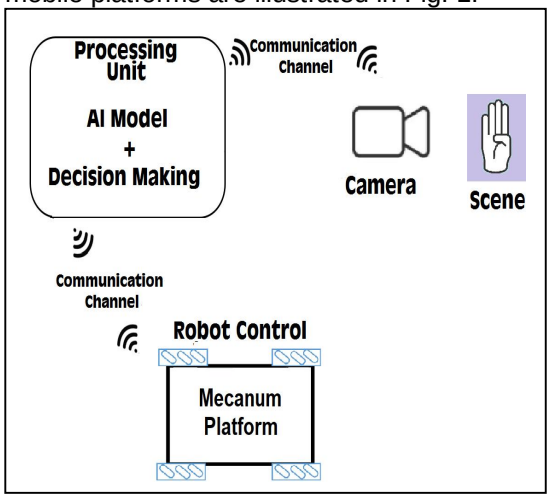


Fig. 1 *Requirement for vision based hand guided control system* for mobile platforms.

System Requirements

Solving the aforementioned problem requires defining the main system components and requirements. The proposed system is divided into three main components: the visual sensor (the "eye"), the central processing unit (the "brain"), and the robot control layer, as described below.

A. **Image processing module and visual sensor:** This module captures images of the environment and transmits them for further processing. It may also perform lightweight local processing tasks, effectively acting as the robot's "eye." It acquires visual information, performs initial preprocessing when necessary, and sends the resulting data to the main processing unit.
B. **Processing Unit (Brain):** The processing unit is responsible for running the AI-based vision models. Two options can be considered: a PC-based platform capable of executing computationally intensive detection models, and an embedded platform designed for deploying lightweight AI models.
C. **Communication:** All components should be connected to the same network.
D. **Robot:** For development and validation purposes, either virtual models or real mobile robots can be used to evaluate the proposed approach.

Global Architecture

The proposed vision-oriented mobile robotic system is designed using a modular architecture. The system operation begins with image acquisition from the vision sensor (Fig. 2). The camera captures images of the environment and wirelessly streams them to a processing unit using its assigned IP address. The received data are then processed by an image-processing module. Subsequently, the selected AI model is configured and executed to detect the recognize hand gestures. Based on the detection results, a decision-making module determines the appropriate robot behavior. Finally, the robot is connected through a communication protocol, and the corresponding control commands are transmitted to a mobile platform to perform the required actions.

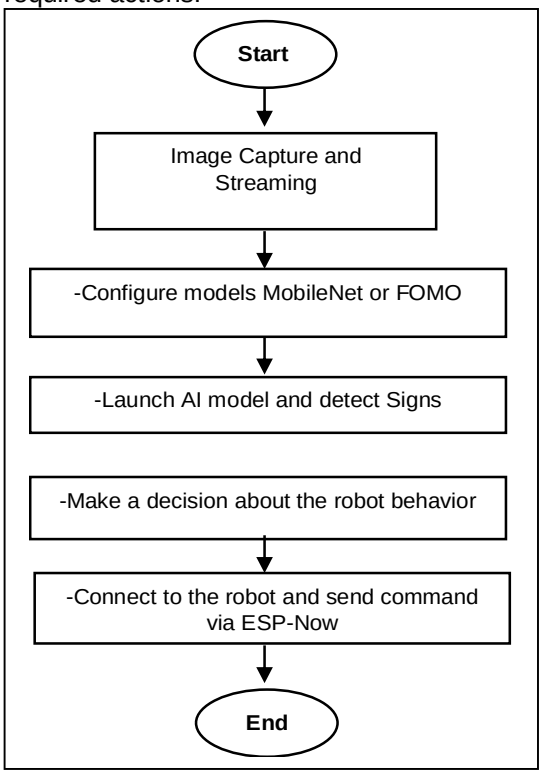


Fig. 2 Flowchart of the overall system architecture

In this article a processing architecture is considered to ensure flexibility between computational performance and embedded deployment. The embedded architecture is designed to support lightweight models, including FOMO and MobileNet V1, suitable for real-time execution on low-power hardware platforms. The ESP32-CAM is

selected as the main visual sensor. Different tools used in this article are presented below.

American Sign Language

American Sign Language (ASL) is a natural language that is used by the Deaf community in the United States and parts of Canada with a grammatical structure comparable to that of spoken languages [21]. Communication in ASL relies on hand shapes, hand movements, facial expressions, and body postures.
In this article, ASL are used in form of hand gestures , serving as the control commands for the mobile robot. Each unique ASL gesture is a distinct class.

Edge-based AI Models

In this article, two principal models are tested and compared.

A. FOMO Model : Edge Impulse FOMO is a lightweight object detection algorithm designed for resource-constrained embedded devices. It enables real-time localization and counting of multiple objects within an image while requiring significantly less computational power and memory than YOLOv5, with reported reductions of up to 30 times [22].

B. MobileNet Models: MobileNet is a family of convolutional neural network architectures specifically designed for mobile and embedded vision systems. These models employ depth wise separable convolutions, which significantly reduce the number of parameters and computational operations while maintaining competitive performance [23].

Mecanum Mobile platform

Mecanum mobile platforms are equipped with small rollers mounted around the circumference of each wheel at a 45-degree angle. Owing to this design, the robot is capable of omnidirectional motion (Fig. 3). Mecanum wheels provide excellent maneuverability, which is particularly advantageous in confined or crowded environments [4].
In this work, the Mecanum-wheel platform is controlled by an ESP32 microcontroller (Fig. 3) interfaced with two H-bridge motor drivers. Four DC motors are independently actuated to enable omnidirectional movement. The platform is powered by dedicated power-supply modules and communicates wirelessly with the processing unit. In addition, the system integrates several sensors, including an IMU for localization and ultrasonic sensors for obstacle detection.

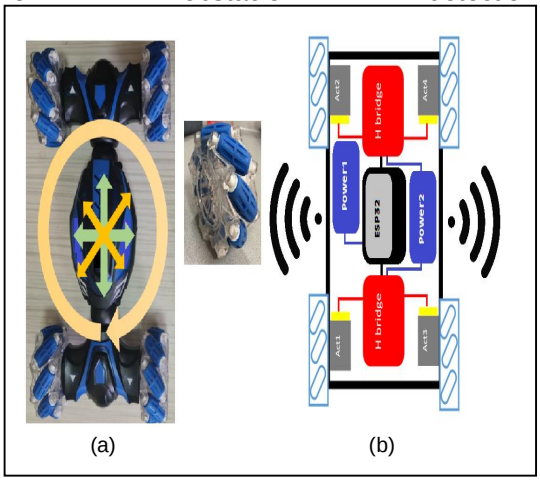


Fig. 3 *Me*canum platform: (a) possible movements and Mecanum wheels (b) hardware components of the mobile robot.

4. EXPERIMENTAL SETUP

System Architecture

The proposed embedded architecture is designed to satisfy real-time requirements while ensuring fully onboard processing (Fig.4).
Lightweight deep learning models, are deployed on resource-constrained embedded platforms such as the ESP32-CAM and ESP32-S3.
The system leverages ASL hand signs (Fig.5) as intuitive commands to control a Mecanum-wheel mobile platform.
A wireless communication framework enables the real-time transmission of recognized commands, allowing the robot to react instantly to the detected hand gestures and perform the corresponding movements.

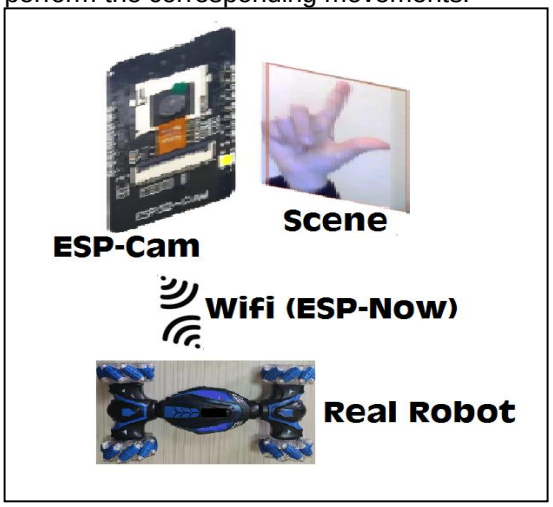


Fig.4 Embedded Architecture for Real-Time ASL hand gesture recognition using lightweight models.

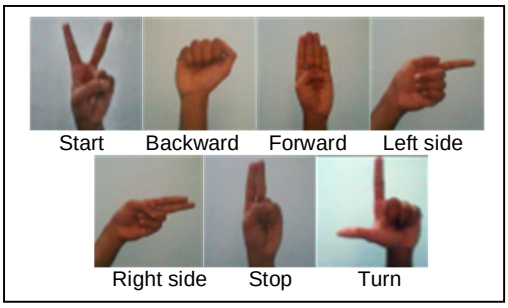


Fig.5 The signs used in the presented work

Model training and Evaluation

In this work, a custom image dataset was created through manual image acquisition to ensure its suitability for the targeted application. The collected images were subsequently filtered and enhanced using data augmentation techniques to improve both the quality and diversity of the dataset. Dataset annotation and management were performed, where each class was manually labeled and the dataset was organized into dedicated training and testing subsets.

A total of 306 unique image items were captured across the classes. The dataset is partitioned into a 79% training set (240 samples) and a 21% testing set (60 samples) using Edge Impulse's automated split mechanism, while validation is handled dynamically in-loop during the training phase. The raw images are processed through an image pipeline configured with Grayscale color depth, scaled down to match the memory and processing constraints of the ESP32 microcontroller. To prevent overfitting and enhance generalization, offline data augmentation was performed using the Albumentations library. A custom stochastic pipeline was applied, including geometric transformations (Horizontal/Vertical Flips, RandomRotate90, and Rotations), photometric adjustments, as well as kernel-based blurring/sharpening filters and Gaussian/Additive Noise injections.

The FOMO and MobileNetV1 architectures were trained using the Edge Impulse platform, which provides an integrated environment for model training and performance evaluation. The resulting models were then exported to embedded platforms for efficient on-device deployment and execution.

The lightweight models were deployed and evaluated on two embedded systems. The ESP32-S3 N16R8 platform was used to assess real-time performance in terms of inference latency, memory usage, and operational stability. It also served to determine the operational limits of the proposed approach by evaluating inference speed and identifying the maximum number of gesture classes that could be reliably handled under embedded resource constraints. In addition, the ESP32-CAM platform was employed to validate the suitability of the proposed approach for real-time robotic applications and autonomous robot control.

Deployment of the gesture-based commands on the Mecanum platform

The ESP32-CAM served both as the vision sensor and the embedded processing unit For robotic deployment. The ESP32-CAM supports video streaming over Wi-Fi using the HTTP protocol with M-JPEG encoding. During the experimental phase, this capability was exploited to transmit commands to an external processing unit for controlling the robot motion. The board captures hand gestures, executes the lightweight deep learning model locally, and generates motion commands in real time. These commands are transmitted to the Mecanum robot through the ESP-NOW communication protocol, ensuring low-latency wireless control.

The robot localization is performed using an Inertial Measurement Unit (IMU). The robot heading is estimated from the yaw angle provided by the IMU, while the traveled distance is estimated from the commanded motion and the elapsed time (equ. 1). For each control interval, the displacement D_k is calculated as:

$$D_k = V_f . \Delta t_k \quad (1)$$

where V_f is the predefined robot velocity and Δt is the elapsed time during the k^{th} control interval. The displacement D_k is then transformed from the robot frame to the global reference frame using the estimated yaw angle θ_k. The robot position is consequently updated according to the following equations (equ.2. and equ. 3.):

$$X_{k+1} = X_k + D_k \cos(\theta_k) \quad (2)$$

$$Y_{k+1} = Y_k + D_k \sin(\theta_k) \quad (3)$$

5. RESULTS AND DISCUSSION

This section presents the experimental results and a comprehensive analysis of the proposed vision-based architectures for the mobile robotic system. The performance of the lightweight embedded models FOMO and MobileNetV1, is evaluated for ASL gesture recognition. The analysis covers training outcomes, classification performance metrics, on-device resource utilization, scalability

assessments, and real-world validation experiments conducted on a Mecanum-wheeled mobile platform. The obtained results highlight the trade-offs among recognition accuracy, inference speed, memory requirements, and suitability for deployment on resource-constrained embedded platforms.

Performance study of the edge-AI models

The gesture recognition models are executed directly on the ESP32-S3 board to investigate the feasibility of deploying AI algorithms on embedded hardware. The primary objective is to implement lightweight models capable of performing inference autonomously on the microcontroller without relying on external computing resources. To assess the storage capacity and computational limits of the ESP32-S3 platform. A series of scalability experiments are conducted by progressively increasing the number of ASL gesture classes from five to seven. These experiments aim to evaluate how model complexity and resource consumption evolve with the number of classes and to identify the threshold at which hardware limitations begin to affect system performance. All the models are compiled and deployed on the ESP32-S3 board using the Arduino IDE.

A. FOMO with 5 Classes: The FOMO model was trained using five-class custom ASL dataset (Left, Right, Start, Stop, Turn).

The confusion matrix confirmed 100% correct classification across all five gesture classes (Fig. 6) with no misclassifications. The feature explorer showed well-separated clusters for all classes, indicating a discriminative feature learning.

The FOMO model initialized and ran correctly, repeatedly. FOMO demonstrates efficiency for real-time embedded inference.

The FOMO model with 5 classes achieves perfect accuracy while consuming minimal hardware resources. These results confirm that FOMO is an efficient model for embedded gesture recognition, making it well-suited for real-time robot control applications on resource-constrained microcontrollers.

B. FOMO with 7 Classes: To test the scalability limits of the ESP32-S3 board, two additional gesture classes (Backward and Forward) were added to the dataset,



Fig.6 Training performance, confusion matrix, and on-device metrics of the FOMO model with 5 classes .

bringing the total to seven classes. The FOMO model is retrained on this expanded dataset and redeployed on the board. With 7 classes, The confusion matrix (Fig. 7) revealed that most classes were classified perfectly (100%), with the exception of the Backward class, which showed an 11.1% misclassification rate into the Background category. The feature explorer showed generally well-separated clusters, though with some overlap between the newly added classes.

Increasing the number of classes from 5 to 7 had a notable impact on FOMO's performance and resource usage. While the model still deployed successfully, the RAM consumption with Inferencing time stay the same for the two models. This experiment demonstrates that FOMO can scale to 7 classes while remaining deployable on the ESP32-S3.



Fig.7 Training performance and confusion matrix of the FOMO model with 7 classes.

C. MobileNetV1 with 5 Classes : MobileNetV1 was tested on the same 5-class ASL dataset. Unlike MobileNetV2, which failed to deploy due to memory overflow, MobileNetV1 is successfully compiled and deployed on the ESP32-S3 board.

MobileNetV1 achieves a good performance (Fig. 8) on the 5-class validation set with an Accuracy of 100%. The confusion matrix showed 100% correct classification across all five gesture classes. MobileNetV1 demonstrated outstanding performance on the 5-class task, achieving perfect scores across all metrics. Compared to FOMO, it requires more flash memory (123.9 KB vs. 81.2 KB) and has a longer inferencing time (812 ms vs. 644 ms), but it delivers higher baseline accuracy. This makes MobileNetV1 a strong candidate when maximum classification precision is the priority and the slight increase in resource usage is acceptable.

D. MobileNetV1 with 7 Classes: MobileNetV1 was then retrained and tested with 7 classes (Backward, Forward, Left, Right, Start, Stop, Turn) to evaluate the impact of increased class complexity on its performance and hardware footprint (Fig. 9). With 7 classes, MobileNetV1 achieved an overall Accuracy of 94.0%. The confusion matrix identified two sources of misclassification: the Forward class was confused with Backward in 20% of cases, and the Stop class was confused with Forward in 12.5% of cases. The remaining five classes (Left, Right, Start, Turn, and Backward) were all classified at 100%. On-device metrics showed an inferencing time of 888 ms, Peak RAM of 182.7 KB, and Flash usage of 159.9 KB. Adding two more gesture classes causes a measurable drop in MobileNetV1's accuracy (from 100% to 94.0%) and increased flash usage from 123.9 KB to 159.9 KB.



Fig.8 Training performance, confusion matrix, and on-device metrics of MobileNetV1 with 5 classes.

The confusion between the Forward and Backward gestures suggests that visually similar hand orientations become harder to distinguish as the number of classes grows. Despite this, the model remained deployable on the ESP32-S3, demonstrating reasonable scalability under hardware constraints.



Fig. 9 Confusion matrix and on-device deployment metrics of MobileNetV1 with 7 classes.

***E.* Comparative Analysis of implemented Models:** To find the best model for real-time use on the ESP32-S3 microcontroller, we measured how fast the models ran (inference times) and how much memory they used (RAM and Flash).

Table 1 : *Performances of all the implemented Models*

Models	Classes	RAM (KB)	Flash (KB)	Inference (ms)
FOMO	5	114.7	81.2	644
MobileNet	5	182.7	123.9	812
FOMO	7	114.7	81.2	644
MobileNet	7	182.7	159.9	888

Table.1 shows the trade-offs between FOMO and MobileNetV1 when both were set up for 7-class gesture recognition.

The experimental results highlight a clear performance advantage of FOMO over MobileNetV1 on the ESP32-S3. FOMO The achieved faster classification time (644 ms vs. 888 ms), thanks to its lightweight grid-based architecture optimized for object detection. Unlike MobileNetV1, which processes the full image through multiple feature extraction layers, FOMO efficiently detects object centers locally, significantly reducing computational load especially noticeable with 7 classes.

In terms of resource efficiency, FOMO demonstrated superior optimization by using less Flash storage and less Peak RAM. The saved memory allows the ESP32-S3 to simultaneously handle motor control, sensor reading, and navigation algorithms without risking system instability or crashes.

Overall, these findings confirm that FOMO is the more suitable model for real-time

gesture detection on resource-constrained devices like the ESP32-S3.

Integration of real mobile platform

The hardware architecture, based on fully embedded models (FOMO and MobileNetV1) running directly on the ESP32-CAM, is tested on the physical Mecanum robot platform.
This architecture eliminates the need for an external PC, offering better autonomy and real-time performance.
The system is built around an ESP32-CAM module, which runs the AI models directly on-device. It ensures real-time gesture recognition from the camera feed without relying on external processing. The ESP32-CAM interfaces with motor drivers to provide direct control of the four Mecanum wheels**,** enabling omnidirectional movement. The onboard inference pipeline was tested with the following command gestures, including “Start” and “Left Side”**.**
The embedded system successfully controlled the physical robot in real indoor environments. Below are the experimental results. As illustrated in Fig. 10, the user presents a gesture to the ESP32-CAM, which captures the image and runs the embedded AI model locally. Once the gesture is recognized, the corresponding motion command is generated and transmitted to the robot, enabling it to perform the requested movement in real time.



Fig.10 Gesture control of the real robot via ESP-CAM.

Both FOMO and MobileNetV1 models were able to run successfully on the ESP32-CAM and control the Mecanum robot in real time.
This validation confirms that the presented architecture is efficient for real-world deployment of the vision-based mobile robotic system.
The robot responded correctly and smoothly to the hand gestures (Fig. 11 and Fig. 12). The FOMO model provided faster response time and lower resource usage, making it more suitable for real-time mobile robotics.

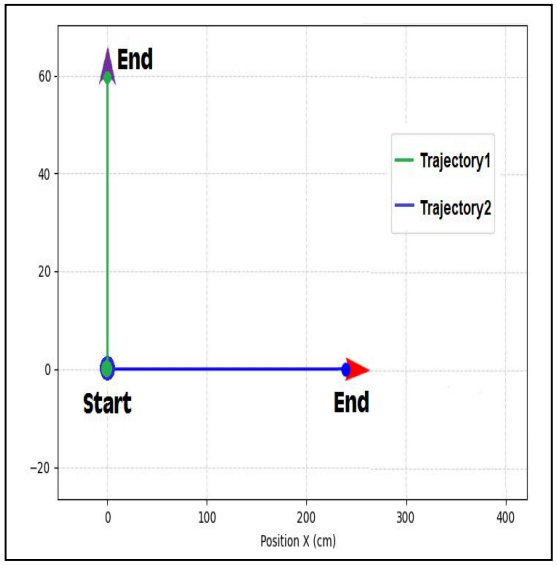


Fig.11. Trajectory of the mobile platform (Trajectory1: lateral movement, Trajectory2: Straight line movement).

The fully embedded solution significantly improved the system’s autonomy, reduced latency, and eliminated dependency on an external computer.

Discussion

The comparative evaluation of FOMO and MobileNetV1 highlights a clear trade-off between classification performance and computational efficiency for embedded deployment. With five gesture classes, both models achieved 100% classification accuracy; however, FOMO required significantly fewer computational resources. Moreover, FOMO achieved a lower inference time.
When the number of gesture classes was increased to seven, the performance gap became more significant. FOMO maintained the same memory requirements and inference time. However, A slight confusion appeared for the Backward gesture, which was misclassified as Background in 11.1% of the cases. This can be mainly attributed to the variability of the dataset. Although the Backward gesture is characterized by a closed fist, some samples exhibit low contrast between the hand and the background, due to image noise, or grayscale representations. These conditions may reduce the visibility of discriminative hand features, causing FOMO to interpret the fist region as Background. MobileNetV1, on the other hand, achieved an overall accuracy of 94.0%, with confusion between Forward and Backward gestures and between Stop and Forward gestures, while its inference time increased to 888 ms. The 20% confusion between Forward and Backward is

likely related to intra-class variability in hand orientation, scale, and illumination. Although the two gestures have distinct canonical configurations, closely aligned fingers in some Forward samples may reduce their discriminative features, making them harder to distinguish from the compact Backward shape. The 12.5% confusion between Stop and Forward may similarly result from variations in imaging conditions. Despite the observed confusion among some classes.

Both models were successfully integrated into the Mecanum mobile platform and operated on the embedded hardware without requiring an external computing system. The FOMO AI model appears to provide the most suitable compromise for the proposed application, as its lower memory footprint and shorter inference time leave more computational resources available for other embedded functions. However, the misclassification observed when using seven classes should be considered when deploying the system in real-world human–robot interaction scenarios, since an incorrect gesture recognition may result in an unintended robot command. Further optimization of the training dataset, particularly by increasing the number and diversity of samples for visually similar gestures, could therefore improve the robustness of the seven-class configuration.

6. CONCLUSION

The embedded FOMO model proved to be highly efficient for real-time deployment on the ESP32 platform with minimal resource usage. The scalability study confirmed that both FOMO and MobileNetV1 can handle up to seven gesture classes within hardware constraints, with FOMO offering superior efficiency. Real robot validation confirmed reliable response to hand gesture commands, validating the complete hardware/software pipeline. These results highlight the feasibility of using lightweight AI models for autonomous vision-based control.

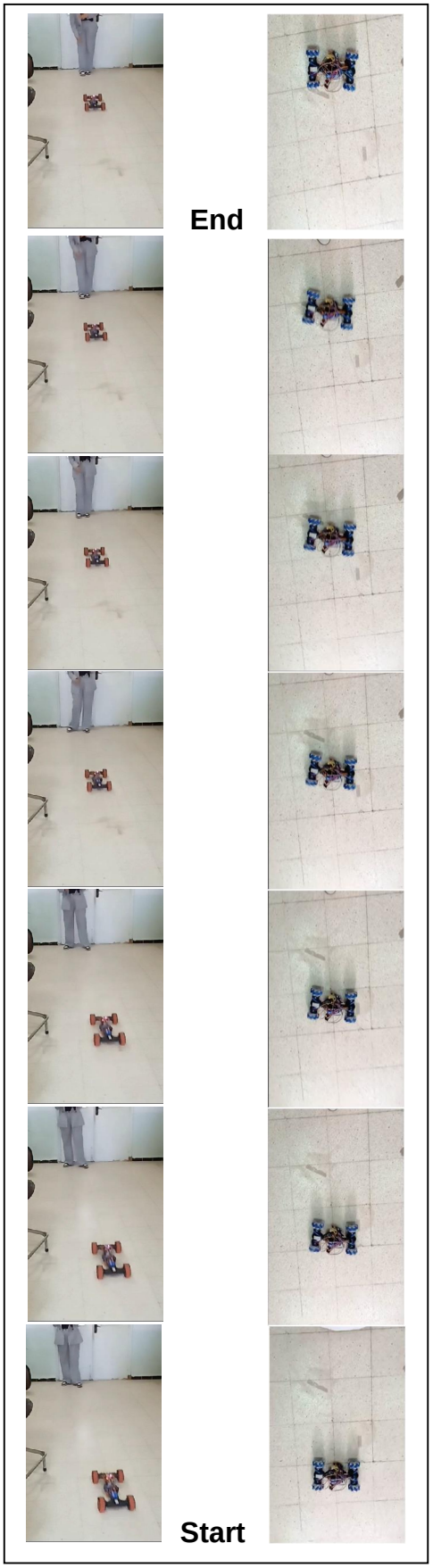


Fig.12. movement of the real robot response to “Start” and “Left Side” Gestures.

References

[1] Rubio, F., Valero, F., & Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, *16*(2), 1729881419839596.

[2] Kim, D., Kwon, J. I., Kim, Y., Kim, D. H., & Choi, C. (2026). AI-native robotic vision systems enabled by in-sensor computing. *npj Unconventional Computing*, *3*(1), 2.

[3] Greco, D., Fasihiany, M., Ranjbar, A. V., Masulli, F., Rovetta, S., & Cabri, A. (2024). Computer vision algorithms on a Raspberry Pi 4 for automated depalletizing. *Algorithms,* 17(8), 363. https://doi.org/10.3390/a17080363

[4] Naik, L., Iversen, T. M., Wilm, J., & Krüger, N. (2025). Multi-view object pose distribution tracking for pre-grasp planning on mobile robots. *Frontiers in Robotics and AI*, *12*, 1683931.

[5] Qi, J., Ma, L., Cui, Z., & Yu, Y. (2024). Computer vision-based hand gesture recognition for human-robot interaction: a review. *Complex & Intelligent Systems*, *10*(1), 1581-1606.

[6] Hentout, A., Maoudj, A., Yahiaoui, D., & Aouache, M. (2019). RRT-A*-BT approach for optimal collision-free path planning for mobile robots. *Algerian Journal of Signals and Systems*, *4*(2), 39-50.

[7] Seray, S. E., Bey, I., & Akli, I. (2024). Waste Sorting Automation with Visual Sensor and RFID. In *2024 3rd International Conference on Advanced Electrical Engineering (ICAEE)* (pp. 1-6). IEEE.

[8] Tchantchane, R., Zhou, H., Zhang, S., & Alici, G. (2023). A review of hand gesture recognition systems based on noninvasive wearable sensors. *Advanced intelligent systems*, *5*(10), 2300207.

[9] Zaman, M. Q., & Wu, H. M. (2023). Hand gesture-based teleoperation control of a mecanum-wheeled mobile robot. *IFAC-PapersOnLine*, *56*(2), 1484-1489.

[10] Fuad, M., Umam, F., Wahyuni, S., Fahriani, N., Nurwahyudi, I., Darwaman, M. I., & Maulana, F. (2023). Towards controlling mobile robot using upper human body gesture based on convolutional neural network. *Journal of Robotics and Control (JRC)*, *4*(6), 856-867.

[11] Laplaza, J., Romero, R., Sanfeliu, A., & Garrell, A. (2023, March). Body gesture recognition to control a social mobile robot. In *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction* (pp. 456-460).

[12] Sahoo, J. P., Sahoo, S. P., Ari, S., & Patra, S. K. (2023). Hand gesture recognition using densely connected deep residual network and channel attention module for mobile robot control. *IEEE Transactions on Instrumentation and Measurement*, *72*, 1-11.

[13] Chen, L., Li, C., Fahmy, A., & Sienz, J. (2024). GestureMoRo: an algorithm for autonomous mobile robot teleoperation based on gesture recognition. *Scientific Reports*, *14*(1), 6199.

[14] Weng, W. T., Huang, H. P., Zhao, Y. L., & Lin, C. Y. (2022). Development of a visual perception system on a dual-arm mobile robot for human-robot interaction. *Sensors*, *22*(23), 9545.

[15] Suryavanshi, L. M., Chandraraj, A. J., Lochan, K., & Nag, P. (2023). Hand Gesture-Controlled Wheeled Mobile Robot for Prospective Application as Smart Wheelchairs. In *Smart Sensors Measurement and Instrumentation: Select Proceedings of CISCON 2021* (pp. 437-448). Singapore: Springer Nature Singapore.

[16] Mahmud, K. A., Ahmed, F., Siddiq, R., & Sobhan, S. (2025, December). Edge-Optimized Firefighting Robot: Autonomous Navigation via PID Control and Fire Detection with FOMO Vision. In *2025 7th International Conference on Electrical Information and Communication Technology (EICT)* (pp. 1-6). IEEE.

[17] Gholami, R., Jahromi, H. D., & Sedaghat, S. (2025). Design and implementation of a near real-time human detection robot using YOLO framework and IoT technologies. *IEEE Access*.

[18] Galarza-Falfan, J., García-Guerrero, E. E., Aguirre-Castro, O. A., López-Bonilla, O. R., Tamayo-Pérez, U. J., Cárdenas-Valdez, J. R., ... & Inzunza-Gonzalez, E. (2024). Path planning for autonomous mobile robot using intelligent algorithms. *Technologies*, *12*(6), 82.

[19] Wang, X., Ma, X., Gao, H., Wang, L., & Song, X. (2025). Thermal Imaging-Based Lightweight Gesture Recognition System for Mobile Robots. *Machines*, *13*(8), 701.

[20] Ikegami, S., Premachandra, C., Sudantha, B. H., & Sumathipala, S. (2018, December). A study on mobile robot control by hand gesture detection. In *2018 3rd international conference on information technology research (ICITR)* (pp. 1-5). IEEE.

[21] Petitto, L. A. (1994). Are signed languages 'real'languages. *Evidence from American Sign Language and Langue des Signes Québécoise. Reprinted from: Signpost (International Quarterly of the Sign Linguistics Association)*, *7*(3), 1-10.

[22] Edge Impulse, "FOMO: Object detection for constrained devices," [Online]. Available at: https://docs.edgeimpulse. com/docs/edge-impulse-studio/learning-blocks/object-detection/ fomo-object-detection-for-constrained-devices. [Accessed: June 25, 2026].

[23] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.