DOI: 10.20535/1970.70(2).2025.348019
UDC 631.362:004.896

CONTROL SYSTEM FOR MOBILE AGROBOT BASED ON NEURAL NETWORK OBJECT DETECTION

[1)]O. I. Tsompel, [1)]M. O. Bezuhlyi, [2)]Andrzej Dzierwa
[1)]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine;
[2)]Rzeszow University of Technology, Rzeszów, Poland
E-mail: sashatsompel@gmail.com; m.bezuglyi@kpi.ua; adzierwa@prz.edu.pl

The paper addresses the urgent scientific and technical problem of automating crop care processes within the framework of the Precision Agriculture 4.0 paradigm. The primary objective of the study is to develop the architecture and investigate the efficiency of a cost-effective cyber-physical system (CPS) for autonomous crop monitoring and targeted physical destruction of weeds in real-time. This approach minimizes the reliance on chemical herbicides, thereby addressing the issues of weed resistance and environmental soil degradation. The authors propose and implement a two-level hierarchical control system for a mobile agricultural robot. The high-level computing layer is based on the Raspberry Pi 4 Model B single-board computer, which handles computer vision tasks and strategic path planning. The YOLOv8 Nano neural network architecture was selected and justified for semantic segmentation of vegetation cover. A set of optimization methods for Edge devices was applied, specifically model conversion to ONNX format and dynamic weight quantization to INT8 format. This reduced the model size to 6 MB and ensured stable inference on the CPU without hardware acceleration. The network training was conducted on a dataset of 10,000 images using a loss function that combines the IoU metric, binary cross-entropy, and Distribution Focal Loss. The low-level control is implemented on the ESP32 microcontroller (Dual Core architecture) running the FreeRTOS real-time operating system. Multi-threaded software was developed to separate communication tasks, inertial sensor (IMU) polling, and PWM signal generation. A discrete PID controller was implemented to stabilize the angular velocities of the differential drive platform wheels, compensating for errors caused by soil slippage. Inter-level communication is established via a UART interface (115200 baud) using a custom JSON-based protocol. A Direct Mapping method is proposed for manipulator control, eliminating the need for resource-intensive inverse kinematics calculations. Field test results confirmed the high efficiency of the system: a detection accuracy of mAP@0.5 at 92.4% was achieved with an average frame processing speed of 65–70 ms (14.5 FPS). The total latency in the control loop does not exceed 75 ms, and the platform positioning error is within ±2.5 cm. Energy monitoring indicated power consumption at the level of 18–22 W, providing up to 60 minutes of autonomous operation.

Keywords: *agrobot, YOLOv8, Precision Agriculture 4.0, ESP32, Raspberry Pi, Computer Vision, PID controller, Boustrophedon, FreeRTOS.*

Introduction

The contemporary stage of development in the agro-industrial complex is characterized by the transition to the "Agriculture 4.0" concept. This paradigm relies on the integration of the Internet of Things (IoT), robotics, and Artificial Intelligence (AI) into high-tech cycles for crop monitoring, real-time decision-making, and the execution of precision agrotechnical operations [1, 2].

One of the persistent global challenges is weed control, traditionally performed through the broadcast application of herbicides. This approach is increasingly recognized as inefficient due to the development of herbicide resistance in weeds, the accumulation of toxic residues in produce, and groundwater contamination. This results not only in significant economic costs but also in soil degradation and ecological crises, as detailed in the works of leading researchers in agroecology and precision farming [3]. According to FAO estimates, despite the use of plant protection products, global crop losses caused by weeds amount to approximately 34% [4].

The response to these challenges lies in the concept of precision agriculture, which entails a shift from blanket field treatment to site-specific intervention [2]. Autonomous robotic platforms constitute a pivotal element of such technologies. Market analysis conducted in [5] indicates that existing commercial solutions are characterized by high costs, which hinders their widespread implementation in the small and medium-sized farming segment.

Scientific literature extensively reviews methods for automatic plant identification. The modern scientific community is actively advancing the application of computer vision in agronomy [6, 7]. While early approaches were based on spectral analysis, the advent of deep learning has established Convolutional Neural Networks (CNNs) as the standard [8]. Studies [9, 10] convincingly demonstrate the advantages of the YOLO family detectors for high-speed object detection and classification in dynamic field conditions, particularly when operating on resource-constrained on-board computing units.

However, the practical application of modern neural network architectures, as discussed in [11], faces limitations regarding the computational resources of on-board systems. A key challenge in designing autonomous agricultural robots is not merely detection accuracy, but ensuring real-time synchronization between the computer vision system and the actuation mechanism, which necessitates the development of integrated control approaches.

The objective of this work is to develop and investigate a cost-effective control system for a mobile agricultural robot that ensures the coordinated operation of the neural network detection loop and the platform motion stabilization loop.

2. Materials and Methods

2.1. Mathematical Model of Kinematics

To facilitate efficient maneuvering within narrow inter-row spaces and to negotiate challenging soil conditions, the selection of a four-wheeled platform utilizing a differential drive was justified. In contrast to conventional steering configurations, this architecture enables the implementation of a zero turn radius [12]

$$q=[x,y,\theta]^T,$$

where *x* and *y* denote the coordinates of the platform's geometric center, and θ represents the orientation angle of the body relative to the abscissa axis.

The mathematical model of the motion kinematics is presented in matrix form by the expression [13]:

$$\dot{q}=\begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}, \qquad (1)$$

In the presented model, V denotes the linear velocity of the platform's geometric center, while ω represents the angular velocity of the chassis rotation. Kinematic control is achieved by calculating and applying the requisite angular velocities for the right, ω_R and left, ω_L drive wheels.

The relationship between the target motion parameters and the angular velocities of the wheels is defined by the system of equations (2):

$$v=\frac{R}{2}(\omega_R+\omega_L),\omega=\frac{R}{L}\alpha(\omega_R-\omega_L), \qquad (2)$$

where R denotes the wheel radius, and L represents the distance between the wheels (wheel base).

The incorporation of the coefficient μ is necessitated by the specific operational dynamics of the agricultural robot. In contrast to locomotion on rigid surfaces, movement over deformable terrain induces longitudinal and lateral slippage of the drive wheels during maneuvering. It has been experimentally established that for dry soil of medium density, the optimal value of this coefficient is $\alpha\approx 0.85$.

2.2. Hardware

The hardware component of the complex is implemented based on a two-level hierarchical architecture, which ensures the decomposition of control tasks and the optimization of computational load. The schematic diagram of the electrical connections of the system components is presented in Fig. 1.

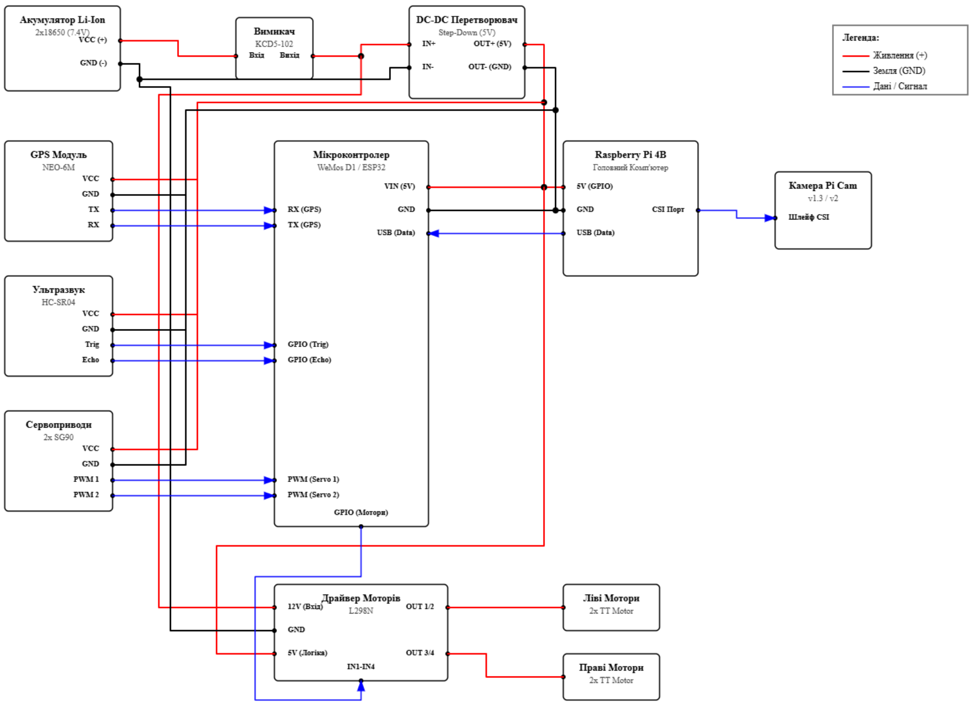

Fig. 1. Schematic diagram of the electrical connections of the system components

A Raspberry Pi 4 Model B single-board computer (Raspberry Pi Foundation, UK) was selected as the central control module. The choice of this platform is justified by its optimal balance of computational power (ARM Cortex-A72 architecture) and energy efficiency. The availability of a hardware Camera Serial Interface (CSI) provides direct access to the video core, thereby minimizing latency during video stream processing and reducing the load on the central processing unit (CPU) during the execution of computer vision algorithms.

Real-time control functions for the actuation equipment are assigned to an ESP32 DevKit V1 microcontroller (based on the ESP32-WROOM-32 chip by Espressif Systems, PRC). The system architecture provides for load distribution between the two microcontroller cores: the first core handles communication (WiFi/UART), while the second is dedicated to generating PWM signals for the drivers and sensor polling. This approach guarantees the determinism of the control process and the operational stability of the actuators, even in the event of latency within the data transmission channel.

Power supply for the mobile platform is provided by a battery pack consisting of 18650 lithium-ion cells (2S configuration, nominal voltage 7.4 V, capacity 3000 mAh). To ensure a stable power supply for low-voltage logic circuits, a step-down switching DC-DC converter based on the LM2596 IC with a 5 V output voltage was utilized [14]. Control of the electric motor power circuits is implemented using an L298N driver, which provides current switching of up to 2 A per channel.

2.3. Features and Algorithm for End-Effector Positioning Control

A key task of the system following the detection of a target object (weed) is the transformation of its screen coordinates into a corresponding control signal for the actuation mechanism. This requires the alignment of two reference frames: the Cartesian coordinate system of the digital image, where position is defined in pixels (with the axis directed to the right), and the polar coordinate system of the manipulator, where the angle is measured relative to the robot's longitudinal axis.

To ensure high targeting speed without employing resource-intensive iterative inverse kinematics algorithms, a Direct Mapping coordinate method is proposed. The rotation angle of the servo drive output shaft, α_{servo} is determined via an affine transformation of the horizontal coordinate of the object's center, x, within the image coordinate system. The calculation formula is expressed as follows:

$$\theta = \theta_{max} - \left(\frac{x}{W} \times \left(\theta_{max} - \theta_{min} \right) \right), \qquad (3)$$

where W – represents the frame width in pixels; θ_{min} , θ_{max} – denote the limiting angles of servo drive rotation (physical operating range); and x corresponds to the current horizontal coordinate of the detected object's center of mass [15].

The physical interpretation of expression (3) is as follows: the ratio $\frac{x}{W}$ normalizes the object's coordinate, mapping it to the dimensionless range [0; 1]. The multiplier $\theta_{max} - \theta_{min}$ scales the obtained value according to the manipulator's operating sector. The operation of subtraction from the upper limit θ_{max} implements the necessary mirror inversion of the control signal. This compensates for the directional discrepancy between the axes, since, in the standard image coordinate system, the X-axis is oriented from left to right, whereas an increase in the servo rotation angle mechanically corresponds to counter-clockwise rotation.

3. SOFTWARE IMPLEMENTATION

3.1. Selection of Neural Network Architecture

The computer vision module constitutes a pivotal component of the system. A comparative analysis of contemporary lightweight neural network architectures was conducted to select the optimal model (Table 1).

Table 1. Comparative analysis of neural network architectures for edge devices

Architecture	**Model Size (FP32), MB**	**mAP@0.5 (COCO), %**	**Inference Time (RPi 4 CPU), ms**
MobileNet SSD v2	14.0	22.1	~85
YOLOv5 Nano	3.9	28.0	~110
YOLOv8 Nano	6.2	37.3	~135

Based on the analysis presented in [16, 17], the YOLOv8 Nano architecture was selected, as it provides the highest accuracy while maintaining acceptable processing speed. To optimize execution on the CPU, the model was converted to the ONNX format employing dynamic quantization (INT8). This transformation resulted in a reduction of inference time to 65–70 ms.

3.2. Neural Network Detection (YOLOv8)

The core of the computer vision subsystem is the YOLOv8 Nano architecture. The selection of this model is justified by its high computational efficiency on embedded systems (Edge Devices). It is a one-stage detector utilizing an anchor-free approach, which simplifies the training process and accelerates inference.

Model training was conducted on a specialized dataset comprising 10,000 images representing various vegetative stages of weeds and crop plants. To enhance the robustness of the detector, data augmentation techniques were employed, including affine transformations, brightness histogram equalization, noise injection, and Mosaic augmentation.

The objective Loss Function is defined as the weighted sum of three components:

$$\mathcal{L}_{total} = \lambda_{box}\mathcal{L}_{box} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{dfl}\mathcal{L}_{dfl}\ , \quad (4)$$

where $\mathcal{L}_{box}$ – represents the bounding box regression loss (based on the IoU metric), $\mathcal{L}_{cls}$ – enotes the binary cross-entropy for the classification task, and $\mathcal{L}_{dfl}$ – corresponds to the Distribution Focal Loss utilized for object boundary refinement [18].

In the practical implementation, the weighting coefficients $\lambda_{box}, \lambda_{cls}, \lambda_{dfl}$ in formula (4) were defined in the hyperparameter configuration file hyp.yaml as follows: box: 7.5 (prioritizing boundary precision), cls: 0.5 (classification), and dfl: 1.5. This configuration allowed the training emphasis to be shifted toward the precise positioning of the end-effector.

For deployment on the Raspberry Pi platform, the model was exported to the **ONNX** format utilizing dynamic weight quantization (INT8). This resulted in a reduction of the model size to 6 MB and a 2.5-fold increase in inference speed on the CPU compared to the original PyTorch implementation.

3.3. Communication Protocol and Low-Level Control

The interaction between the high-level system (Raspberry Pi) and the low-level controller (ESP32) is implemented via a UART interface (Baud Rate: 115200). To ensure data integrity, an application-layer protocol based on the JSON structure was developed:

```
{
"cmd": "move",
"val": [0.5, -0.2],
 "tool": 1,
"id": 1024,
"crc": "a1b2"
}
```

The ESP32 firmware is built upon the FreeRTOS real-time operating system. The firmware architecture entails the parallelization of processes into three prioritized tasks:

1. **MotionTask** (High Priority): Execution interval of 10 ms. Implements a digital PID controller for wheel rotational speed based on feedback from quadrature encoders.
2. **SensorTask** (Medium Priority): Acquisition and filtration of data from the inertial measurement unit (MPU-6050) utilizing a Kalman filter for orientation estimation.
3. **CommTask** (Low Priority): Asynchronous processing of the UART input buffer, parsing of JSON packets, and updating of state variables.

The discrete implementation of the motor control law is described by the following equation:

$$u[k] = K_p e[k] + K_i \sum_{j=0}^{k} e[j] + K_d\big(e[k] - e[k-1]\big)\ , \quad (5)$$

where e(k) denotes the velocity error at the k-th time step. The coefficients $K_p, K_i,$ and K_d were experimentally determined using the Ziegler-Nichols method to ensure an aperiodic transient response.

The software implementation of this control law in C++ within the FreeRTOS environment for the ESP32 microcontroller is provided below:

```
// PID controller implementation (MotionTask fragment)
float compute_pid(float target, float current, float dt) {
    float error = target - current;
    static float integral = 0, prev_error = 0;

    integral += error * dt;
    float derivative = (error - prev_error) / dt;

    // Equation (5) implementation
    float output = (Kp * error) + (Ki * integral) + (Kd * derivative);

    prev_error = error;
    return constrain(output, -255, 255);
}
```

3.4. Human-Machine Interface (HMI)

To facilitate system monitoring and control, a web-based interface built upon the Flask framework (Python) was developed. The utilization of WebSocket technology (SocketIO library) enables the transmission of the video stream featuring detection overlay masks, alongside telemetry data (battery charge, coordinates, status), to the client browser with a latency of less than 100 ms.

3.5. System Operation Algorithm

To implement the autonomous execution of agrotechnical operations, a control algorithm functioning on the principle of a closed-loop feedback system was developed. The system logic encompasses a sequence of data processing stages, ranging from sensor data acquisition to the generation of control inputs for the actuators.

The initial stage involves loading the YOLOv8 neural network weights into the onboard computer's RAM, configuring UART interface parameters for communication with the low-level controller, and calibrating the Inertial Measurement Unit (IMU) to establish the zero horizon.

The system captures a frame from the camera via the CSI interface. To ensure compatibility with the neural network's input layer, image resizing to a resolution of

320×320 pixels, pixel value normalization, and color space conversion are performed.

Subsequently, model inference is executed for object detection. The algorithm filters results based on a confidence threshold ($P > 0.6$). Upon detection of an object of the "Weed" class, the coordinates of its geometric center are computed, the manipulator rotation angle is calculated (Equation 3), and an action: destroy command is generated. Conversely, the detection of a "Crop" class object activates a Safety Mask. Based on the analysis of free space within the inter-rows, a vector of linear (v) and angular (ω) velocities is calculated for course correction. The data are serialized into JSON format and transmitted to the ESP32 for execution by the PID controller. A graphical representation of the developed algorithm is presented in Fig. 2.

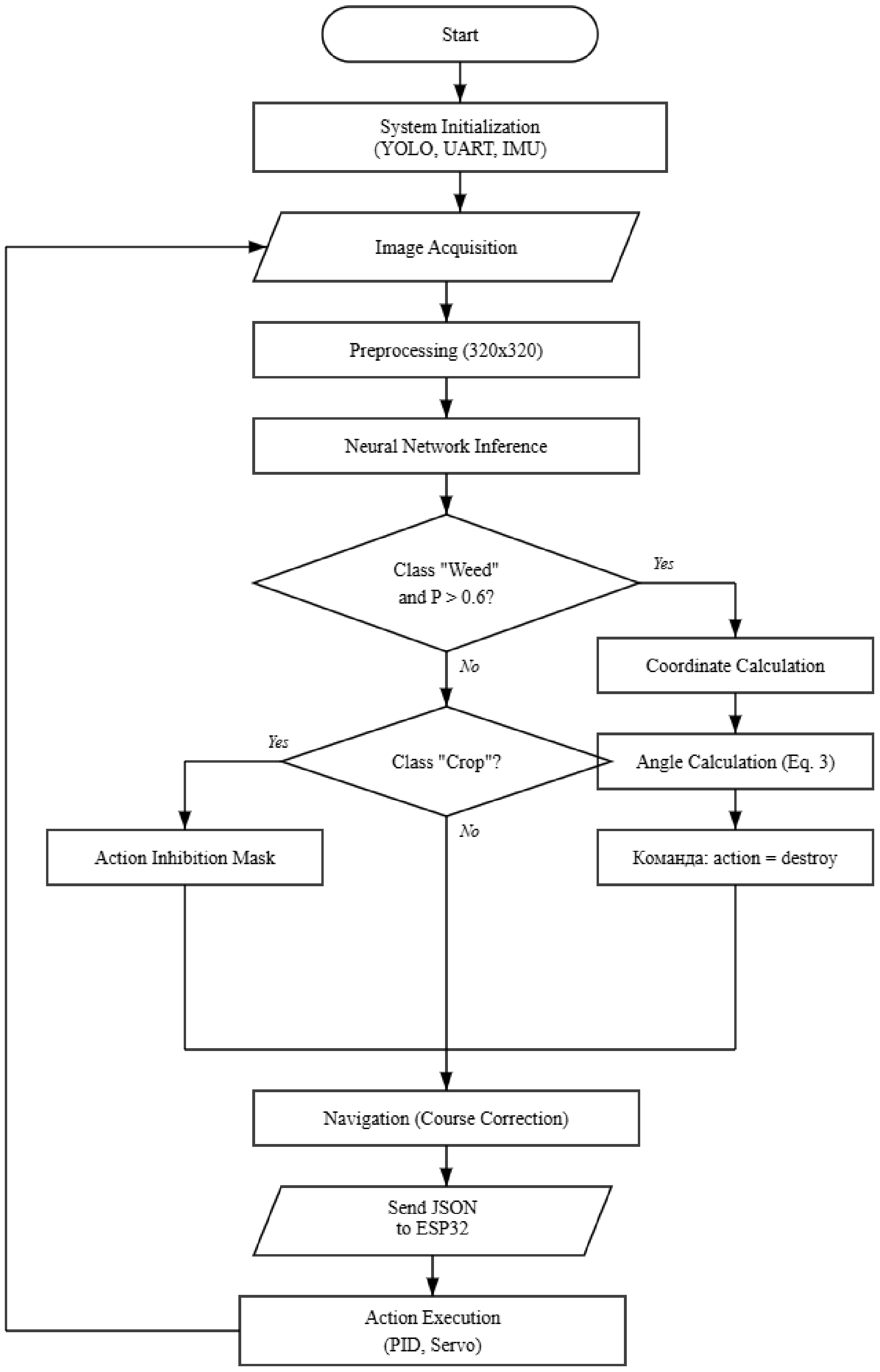
Start
System Initialization (YOLO, UART, IMU)
Image Acquisition
Preprocessing (320x320)
Neural Network Inference
Class "Weed" and P > 0.6?
Yes
No
Coordinate Calculation
Angle Calculation (Eq. 3)
Команда: action = destroy
Class "Crop"?
Yes
No
Action Inhibition Mask
Navigation (Course Correction)
Send JSON to ESP32
Action Execution (PID, Servo)

Fig. 2. Flowchart of the agricultural robot control system operation algorithm

A Python code fragment performing this coordinate mapping on the Raspberry Pi onboard computer is presented below:

```
def get_servo_angle(bbox_center_x, frame_width=320):
    # Practical implementation of formula (3)
    # Coordinate normalization and inversion for the servo drive
    norm_x = bbox_center_x / frame_width
    angle = THETA_MAX - (norm_x * (THETA_MAX - THETA_MIN))
    return int(angle)
```

3.6. Structural Implementation of the Physical Prototype

To experimentally validate the theoretical calculations and the proposed control algorithms, a functional physical prototype of the mobile agricultural robot was developed and fabricated. The general view of the constructed prototype and the layout of its primary components are presented in Fig. 3.

Structurally, the robot is based on a four-wheel drive (4WD) chassis utilizing differential steering. This configuration ensures high maneuverability and the capability to perform zero-radius turns within confined inter-row spaces. The overall dimensions of the platform are 260×160×150 mm.

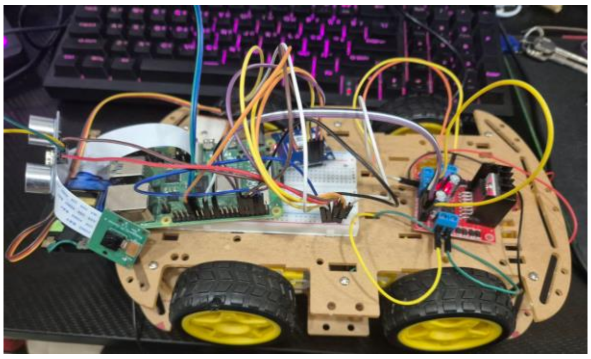
Fig. 3. Hardware layout of the experimental agricultural robot prototype

The component layout is implemented using a two-tier configuration, enabling the separation of power and logic circuits:

- **Lower Level (Power):** This tier houses four brushed DC motors equipped with gearboxes (gear ratio 1:48) and the L298N motor control driver. This arrangement, combined with the placement of the battery pack in the lower section of the chassis, lowers the center of mass, thereby ensuring the platform's kinematic stability when traversing uneven terrain.
- **Upper Level (Computational):** This tier contains the control electronics, specifically the Raspberry Pi 4 single-board computer and a prototyping board for signal line routing. As illustrated in Fig. 3, peripheral connections are established using flexible jumper wires, facilitating rapid reconfiguration of the circuit during the debugging process.

The optical system (Raspberry Pi Camera v3 module) is mounted at the frontal section of the robot. It is connected via a flexible ribbon cable (CSI), minimizing data transmission latency, which is critical for real-time neural network operation. Additionally, an ultrasonic rangefinder is installed on the front panel to implement an emergency stop mechanism upon obstacle detection.

4. Experimental Results

Field trials were conducted on a test site featuring mixed vegetation under natural lighting conditions (sunny/cloudy). The objective of the experiments was to verify detection accuracy, system response speed, and energy efficiency.

4.1. Analysis of Computer Vision Efficiency

The performance results of the detection algorithm are presented in Fig. 4. The model demonstrated a high generalization capability, successfully detecting weeds even under conditions of partial occlusion by crop leaves.

Red bounding boxes indicate weed classification, while green bounding boxes designate crop plants.

To evaluate classification quality, the mAP@0.5 metric (mean Average Precision at an Intersection over Union (IoU) threshold of 0.5) was utilized. Validation results on the test dataset demonstrated an mAP value of 92.4%. The Confusion Matrix revealed that the majority of errors were attributed to False Positives (FP) involving grass species with similar textures; this issue can be mitigated by expanding the dataset [16].

An analysis of temporal characteristics indicated that the transition to the ONNX Runtime execution environment with an input resolution of 320×240 pixels (optimal for inference) enabled the achievement of a stable frame rate of 14.5 FPS.

Concurrently, detection accuracy on the validation dataset was maintained at mAP@0.5 = 92.4%. The total system latency within the control loop (Camera → RPi → ESP32 → Motor) is approximately 65–75 ms, which permits the robot to travel at velocities of up to 0.5 m/s without compromising processing quality.

Due to the implementation of sensor fusion algorithms combining odometry and Inertial Measurement Unit (IMU) data, the platform positioning error does not exceed ±2.5 cm.

4.2. Analysis of Energy Efficiency

Power consumption measurements taken during field trials recorded an average load of 18–22 W in the active patrolling mode. The primary energy expenditure is attributed to the operation of the traction motors during locomotion over soil and the power supply of the Raspberry Pi computing module. The installed battery capacity (3000 mAh, ~22 Wh) is sufficient for 45–60 minutes of autonomous operation. While this metric is adequate for maintaining homestead plots, industrial implementation requires the optimization of the power supply system, specifically through the integration of photovoltaic panels or an automated charging station.

4.3. Analysis of Accuracy and Neural Network Training

In the second stage, the training and validation of the neural network were conducted. The dynamics of the Mean Average Precision (mAP) over 100 training epochs are illustrated in Fig. 5.

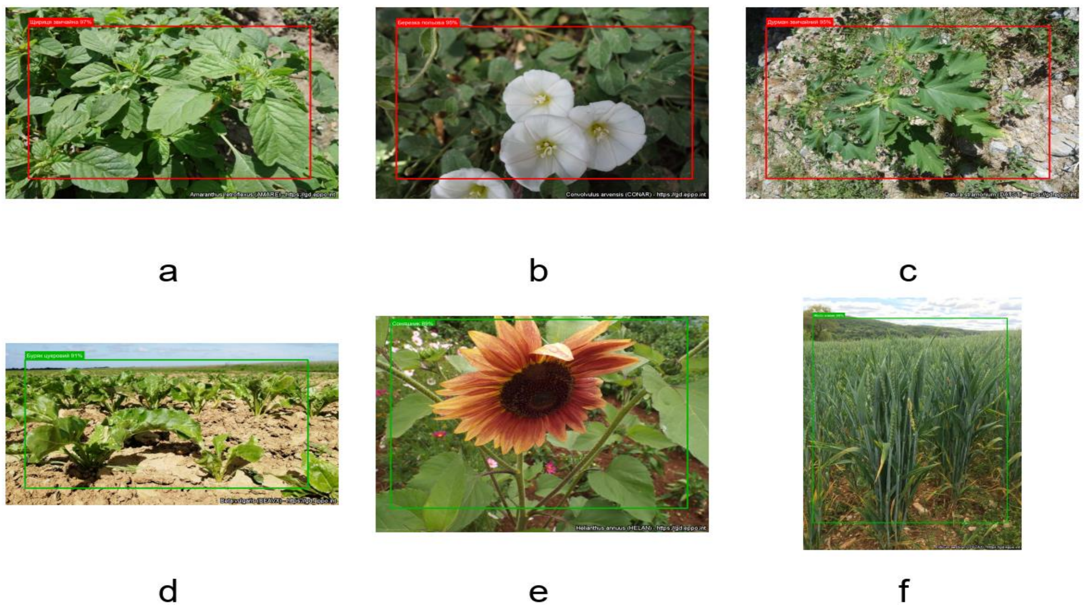

Fig. 4. Results of object detection by the neural network under real-world conditions: (a) Redroot pigweed (*Amaranthus retroflexus*), 97%; (b) Field bindweed (*Convolvulus arvensis*), 95%; (c) Jimsonweed (*Datura stramonium*), 95%; (d) Sugar beet (*Beta vulgaris*), 91%; (e) Sunflower (*Helianthus annuus*), 89%; (f) Winter rye (*Secale cereale*), 89%.

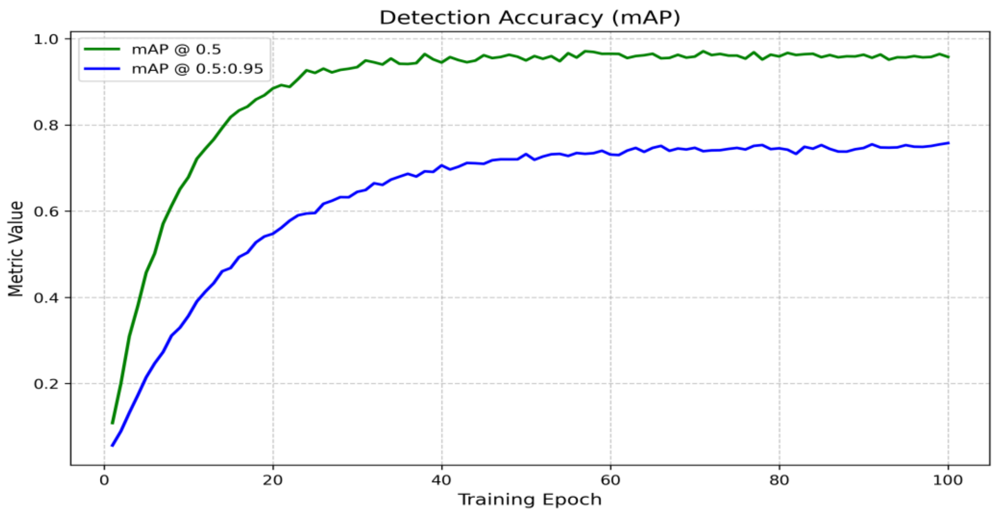

Fig. 5. Neural network training dynamics (metrics mAP@0.5 and mAP@0.5:0.95).

The mAP@0.5 curve (green line) exhibits a plateau at the 0.96–0.97 level subsequent to the 60th epoch, indicating the stability of the training process and the absence of overfitting. This validates the appropriateness of the selected hyperparameters and the data augmentation strategy employed [19].

To conduct a detailed analysis of classification errors, a normalized Confusion Matrix was generated, as presented in Fig. 6. This facilitates the assessment of recognition accuracy for each individual class.

The diagram indicates that the recognition accuracy for the target class "Weed" is 92%, while false negatives (misclassifying a weed as a crop) constitute only 3%.

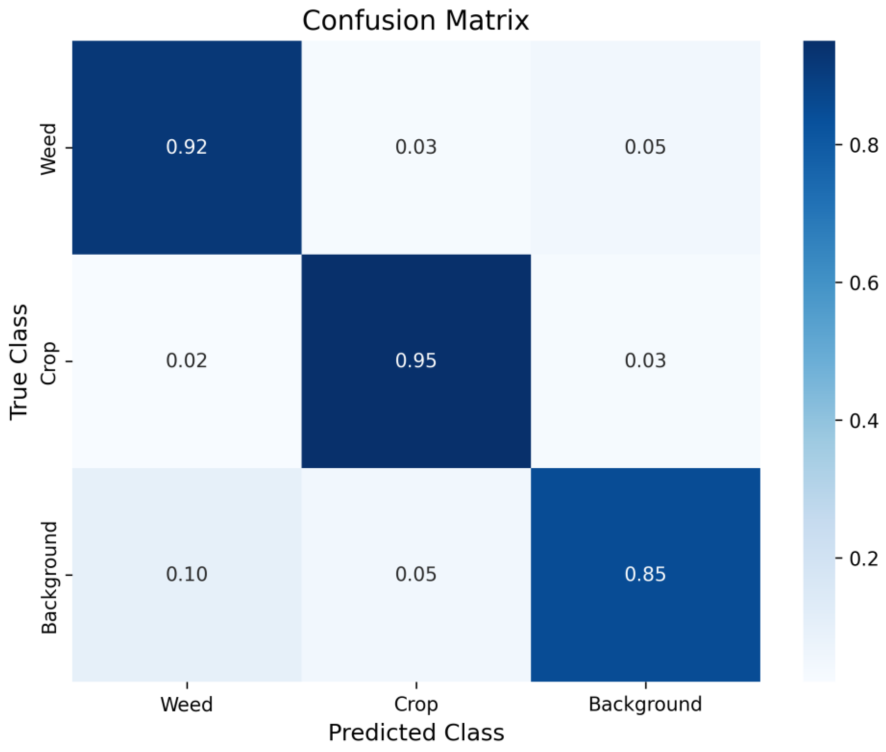

Fig. 6. Normalized Confusion Matrix

The misclassification of a crop as a weed (False Positive) stands at 2%, which is an acceptable metric for automated weeding systems [20], as it minimizes the risk of crop damage.

Based on the obtained data, the integral performance metrics of the system were calculated using the test dataset: Precision was 0.934, and Recall was 0.964. The overall Mean Average Precision (mAP@0.5) reached 92.4%.

Conclusions

This paper proposes a comprehensive solution to the topical scientific and technical problem of weeding automation, implemented through the creation of a cyber-physical system that combines deep learning methods with embedded control algorithms.

The primary scientific outcome of the study is the substantiation and implementation of a two-level control architecture, wherein the distribution of computational load between the high-level module (Raspberry Pi) and the real-time controller (ESP32) addressed the resource-intensive nature of computer vision algorithms. This configuration ensured the requisite performance for neural network inference while maintaining the strict determinism of control signal generation, which is critical for platform motion safety.

The efficacy of utilizing quantized models has been experimentally proven: the application of the YOLOv8 Nano architecture in INT8 format enabled a video stream processing speed of 14.5 FPS on general-purpose processors without the involvement of specialized hardware accelerators. This confirms the feasibility of real-time system operation while maintaining low hardware costs. Concurrently, the implementation of the kinematic model in conjunction with PID speed control ensured motion stability over uneven terrain and mitigated the impact of wheel slippage during maneuvering in confined spaces.

Future development of the project envisions the integration of an RTK-GPS module to achieve centimeter-level global positioning accuracy, as well as the development of a laser system for the non-contact elimination of weeds, which will minimize mechanical and chemical impact on the soil.

References

[1] S. G. Vougioukas, “Agricultural robotics,” *Annu. Rev. Control Robot. Auton. Syst.*, vol. 2, pp. 365–392, 2019. DOI: 10.1146/annurev-control-053018-023617.

[2] Y. Liu, X. Ma, L. Shu, G. P. Hancke, and A. M. Abu-Mahfouz, “From Industry 4.0 to Agriculture 4.0: Current status, enabling technologies, and research challenges,” *IEEE Trans. Ind. Informatics*, vol. 17, no. 6, pp. 4322–4334, 2021. DOI: 10.1109/TII.2020.3003910.

[3] S. B. Powles and Q. Yu, “Evolution in action: Plants resistant to herbicides,” *Annu. Rev. Plant Biol.*, vol. 61, pp. 317–347, 2010. DOI: 10.1146/annurev-arplant-042809-112119.

[4] FAO, *The Future of Food and Agriculture – Trends and Challenges*. Rome, Italy: Food and Agriculture Organization of the United Nations,

2017.
[5] R. R. Shamshiri, C. Weltzien, I. A. Hameed, I. J. Yule, T. E. Grift, and S. K. Balasundram, et al., “Research and development in agricultural robotics: A perspective of digital farming,” *Int. J. Agric. Biol. Eng.*, vol. 11, no. 4, pp. 1–14, 2018. DOI: 10.25165/j.ijabe.20181104.4278.
[6] A. Kamilaris and F. X. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” *Comput. Electron. Agric.*, vol. 147, pp. 70–90, 2018. DOI: 10.1016/j.compag.2018.02.016.
[7] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, “Machine learning in agriculture: A review,” *Sensors*, vol. 18, is. 8, 2674, 2018. DOI: 10.3390/s18082674.
[8] S. M. Hasan, F. Sohel, D. Diepeveen, H. Laga, and M. G. Jones, “~~Image-based weed detection:~~ A survey of deep learning techniques,” *Comput. Electron. Agric.*, vol. 184, 106067, 2021. DOI: 10.1016/j.compag.2021.106067.
[9] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” *arXiv* 1804.02767, 2018. DOI: 10.48550/arXiv.1804.02767.
[10] J. Terven and D.-M. Córdova-Esparza, “A comprehensive review of YOLO architectures in computer vision,” *Mach. Learn. Knowl. Extr.*, vol. 5, is. 4, pp. 1680–1716, 2023. DOI: 10.3390/make5040083.
[11] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A survey of quantization methods for efficient neural network inference,” *arXiv* 2103.13630, 2021. DOI: 10.48550/arXiv.2103.13630.
[12] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA, USA: MIT Press, 2011.
[13] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. London, U.K.: Springer, 2010. DOI: 10.1007/978-1-84628-642-1.
[14] Texas Instruments, “LM2596 SIMPLE SWITCHER® power converter 150 kHz 3A step-down voltage regulator,” Datasheet SNVS124G, 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2596.pdf; STMicroelectronics, “L298 — Dual full-bridge driver,” Datasheet CD00000240, 2016. [Online]. Available: https://www.st.com/resource/en/datasheet/l298.pdf
[15] Arduino, “Servo library,” Documentation. [Online]. Available: https://www.arduino.cc/reference/en/libraries/servo/
[16] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics YOLOv8,” GitHub repository, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics
[17] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 7464–7475. DOI: 10.1109/CVPR52729.2023.00721.
[18] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, and J. Yang, “Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection,” *arXiv* 2006.04388, 2020. DOI: 10.48550/arXiv.2006.04388.
[19] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Inf. Process. Manag.*, vol. 45, is. 4, pp. 427–437, 2009. DOI: 10.1016/j.ipm.2009.03.002.
[20] A. Binch and C. W. Fox, “Controlled comparison of machine vision algorithms for *Rumex* and *Urtica* detection in grassland,” *Comput. Electron. Agric.*, vol. 140, pp. 123–138, 2017. DOI: 10.1016/j.compag.2017.05.018

УДК 631.362:004.896
1)О. І. Цомпель, 1)М. О. Безуглий, 2)Анджей Дзєрва
1)National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine;
2)Rzeszow University of Technology, Rzeszów, Poland

СИСТЕМА КЕРУВАННЯ МОБІЛЬНИМ АГРОРОБОТОМ НА ОСНОВІ НЕЙРОМЕРЕЖЕВОЇ ДЕТЕКЦІЇ ОБ'ЄКТІВ

У роботі вирішується актуальна науково-технічна задача автоматизації процесів догляду за сільськогосподарськими культурами в контексті парадигми Precision Agriculture 4.0.
Метою дослідження є розробка архітектури та дослідження ефективності бюджетної кіберфізичної системи (CPS) для автономного моніторингу посівів та точкового фізичного знищення бур'янів у режимі реального часу. Це дозволяє мінімізувати використання хімічних гербіцидів, вирішуючи проблему резистентності шкідливих рослин та екологічного навантаження на ґрунти.
Авторами запропоновано та реалізовано дворівневу ієрархічну систему керування мобільним агророботом. Верхній рівень обчислень базується на одноплатному комп'ютері Raspberry Pi 4 Model B, який виконує задачі ком-

п'ютерного зору та стратегічного планування. Для семантичної сегментації рослинного покриву обґрунтовано вибір нейромережевої архітектури YOLOv8 Nano.
Застосовано комплекс методів оптимізації для Edge-пристроїв: конвертацію моделі у формат ONNX та динамічне квантування ваг до формату INT8, що дозволило зменшити розмір моделі до 6 МБ та забезпечити стабільний інференс на CPU. Навчання мережі проводилося на датасеті з 10 000 зображень із використанням функції втрат, що комбінує метрику IoU, бінарну крос-ентропію та Distribution Focal Loss. Нижній рівень керування реалізовано на мікроконтролері ESP32 (архітектура Dual Core) під керуванням операційної системи реального часу FreeRTOS. Розроблено багатопотокове програмне забезпечення, що розділяє задачі комунікації, опитування інерціальних датчиків (IMU) та генерації ШІМ-сигналів.
Реалізовано дискретний PID-регулятор для стабілізації кутових швидкостей коліс платформи з диференціальним приводом, що нівелює похибки від проковзування на ґрунті. Взаємодія між рівнями здійснюється через UART-інтерфейс (115200 бод) за авторським протоколом на базі JSON. Для керування маніпулятором запропоновано метод прямого відображення координат (Direct Mapping), що виключає необхідність ресурсомістких обчислень оберненої кінематики. Результати польових випробувань підтвердили високу ефективність системи: досягнуто точність детекції mAP@0.5 на рівні 92.4% при середній швидкості обробки кадру 65–70 мс (14.5 FPS). Загальна затримка в контурі керування не перевищує 75 мс, а похибка позиціонування платформи становить ±2.5 см.
Енергетичний моніторинг показав споживання на рівні 18–22 Вт, що забезпечує до 60 хвилин автономної роботи.
Ключові слова: агроробот, YOLOv8, Precision Agriculture 4.0,, ESP32, Raspberry Pi, Computer Vision, PID-регулятор, Boustrophedon, FreeRTOS.

Надійшла до редакції
27 жовтня 2025 року

Рецензовано
19 листопада 2025 року