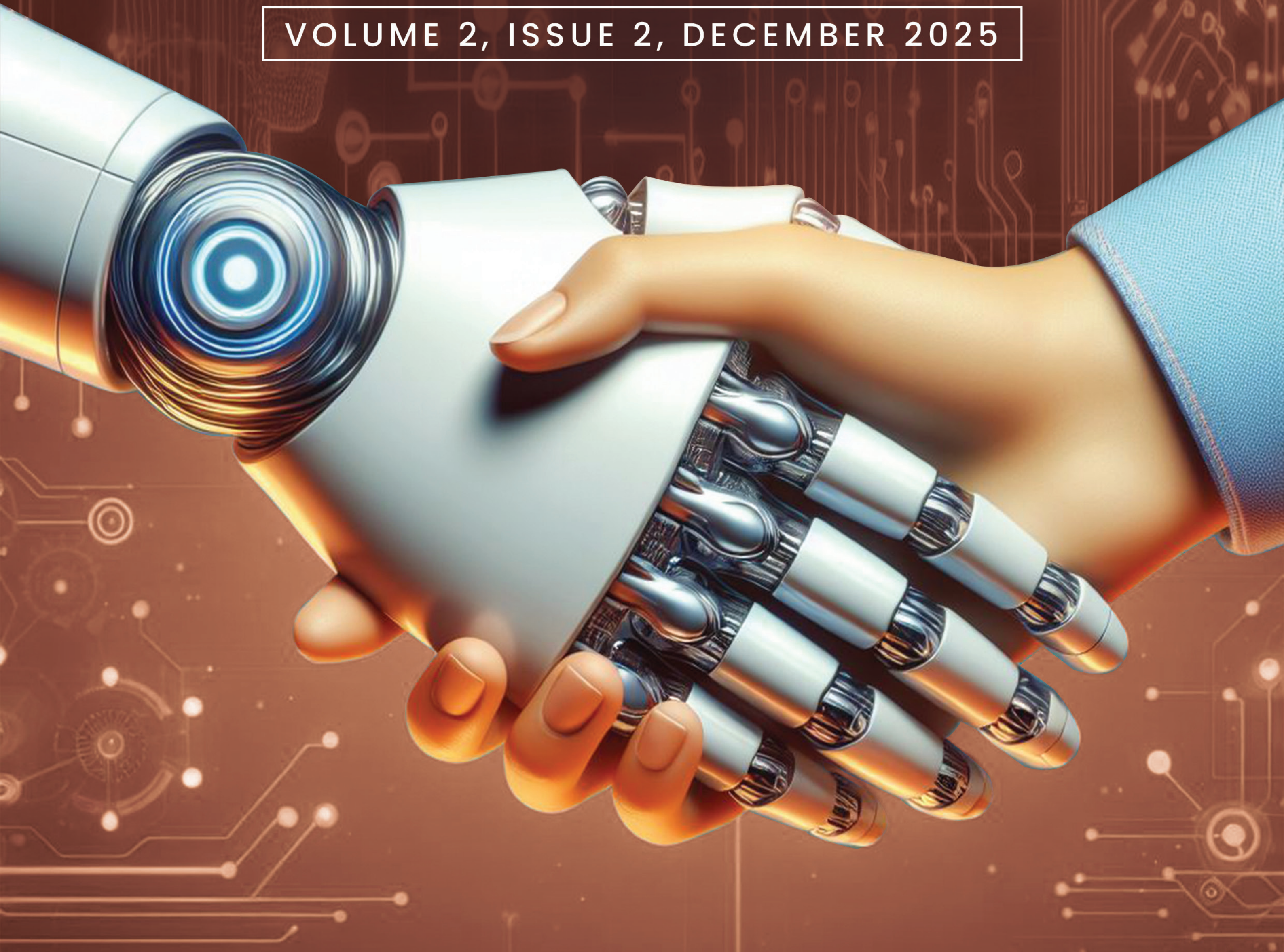
مركز الأكاديمية للنشر

ACADEMY Publishing Center

RIMC

ROBOTICS: INTEGRATION, MANUFACTURING & CONTROL

VOLUME 2, ISSUE 2, DECEMBER 2025



E-ISSN: 3009-7967

ACADEMY PUBLISHING CENTER
Robotics : Integration, Manufacturing and Control (RIMC)
First Edition 2024

RIMC
Volume 2, Issue 2, December 2025
E-ISSN: 3009-7967 Online Version
P-ISSN: 3009-6987 Print Version

Arab Academy for Science, Technology, and Maritime Transport, AASTMT Abu Kir Campus, Alexandria, EGYPT
P.O. Box: Miami 1029
Tel: (+203) 5622366/88 – EXT 1069 and (+203) 5611818
Fax: (+203) 5611818
Web Site: http://apc.aast.edu

RIMC Robotics : Integration, Manufacturing and Control

Journal of Robotics: Integration, Manufacturing and Control is an open access biannual international peer-reviewed publication providing a global platform for the dissemination of research articles, case studies, and reviews across diverse disciplines, focusing on robotics integration and manufacturing.

RIMC strives to contribute essential research findings to the international community, aiding researchers, scientists, institutions, and societies in staying abreast of new developments in theory and applications. We welcome experimental, computational, and theoretical studies that enrich the understanding of robotics related challenges.

RIMC also publishes survey and review articles in the scope. The scope of RIMC includes but is not limited to the following topics: technologies for manufacturing applications, Control algorithms for robots, Sensor technologies, Human-robot collaboration, Robotics for Industry 4.0 and sustainable manufacturing.

The journal is financially supported by the Arab Academy for Science, Technology, and Maritime Transport AASTMT in order to maintain quality open-access sources of research papers on Computing and Engineering.

RIMC has an outstanding editorial and advisory board of eminent scientists, researchers and engineers who contribute and enrich the journal with their vast experience in different fields of interest to the journal.

Editorial Committee:

Advisory Board:

Table of Contents:

Rescue-Bots: A proposed Multi-robot Architecture for Rescue Missions

Yehia Abdulghafar [1], Feras Hamdan [2], Husain AlQuraini [3], Ali Bohamad [4], Ahmed AlRokhami [5], and Yehia Kotb [6*]

[1,2,3,4,5,6*] American University of the Middle East, Egila, Kuwait.

eng.yehia_abdelghafar@hotmail.com, ferasohamdan@gmail.com, husain.alquraini@gmail.com, alihamadbohamad@gmail.com, a.alrokhami@gmail.com, yehia.kotb@aum.edu.kw

Received: 03 June 2025 Accepted: 18 November 2025 Published: 11 December 2025

Abstract

The growing frequency of natural disasters and armed conflicts has created an urgent need for rapid, reliable, and autonomous rescue solutions, particularly in situations where traditional methods become inefficient or unsafe for human responders. This work presents the design and implementation of a fully autonomous rescue system aimed at detecting survivors and delivering immediate assistance without exposing rescue personnel to risk. The system integrates three main components: an aerial drone, a control center, and a ground vehicle. The drone performs autonomous search operations and transmits detected survivor locations to the control center, which handles decision-making and dispatches commands to the ground vehicle. The vehicle then navigates to the identified location to provide essential aid. A comprehensive high-level and low-level design is developed, detailing the system architecture, detection algorithm, communication framework, and hardware components. The implementation of the drone platform, ground vehicle, pre-trained detection model, and inter-device communication is presented based on this design. The system undergoes multiple tests evaluating drone search patterns, communication reliability, and detection performance. Results demonstrate accurate human detection and effective guidance of the ground vehicle to target locations, confirming the feasibility and robustness of the proposed autonomous rescue solution.

Key-words: Autonomous, Rescue System, Self-Centralized, Low-Level Design, High-Level Design, Pre-trained Model

I. Introduction

In recent times, the world has experienced significant loss of lives due to a combination of natural disasters[3] and armed conflicts[4,5]. A notable example is the devastating earthquake that impacted Turkey and Syria. This event resulted in the deaths of approximately 50,000 people, with many more injured. Additionally, tens of thousands of individuals were missing, and over 100,000 have been displaced, facing a lack of shelter. Figure 1 depicts the aftermath of the earthquake in Turkey and Syria, highlighting the destruction [1].

* Corresponding Author

Figure 1: Aftermath of the Turkey–Syria earthquake, illustrating large-scale structural destruction and debris. Adapted from [1].

Additionally, Figure 2 depicts the escalating trend of natural disasters year by year, attributed to global warming and other sources of climate change. It also categorizes the types of disasters [2].

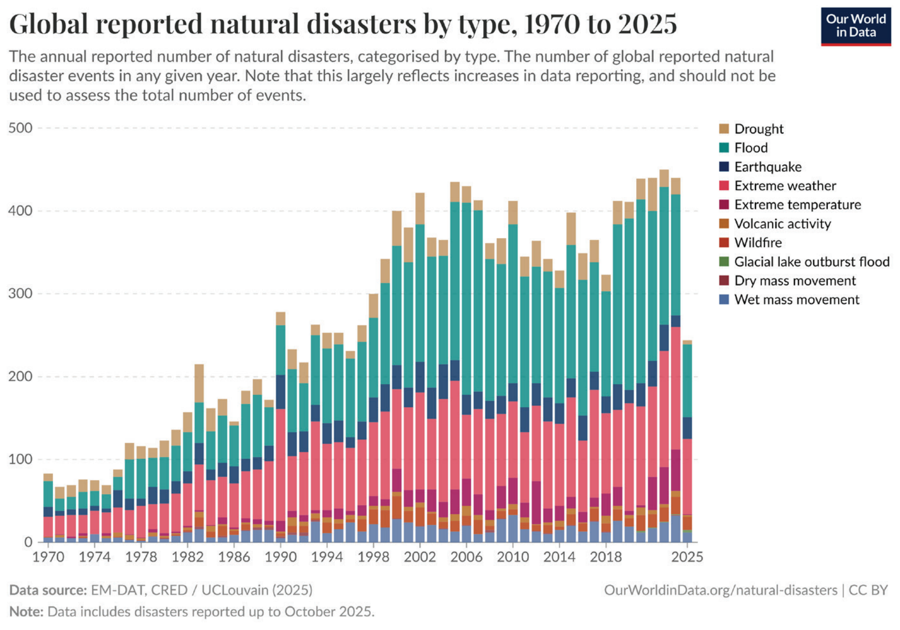


Figure 2: Reported natural disasters by type from 1970 to 2023, showing long-term variation in disaster frequency. Data sourced from [2].

To address the increasing number of natural and human-made disasters, it's important to recognize the urgency caused by rising global temperatures. This will likely face many challenges from these disasters, and it's crucial to find ways to reduce the number of people harmed or killed.

Modern technology plays a crucial role in disaster management [6-10]. Its primary strength lies in its ability to spread awareness and knowledge. This report focuses on leveraging technological advancements to reduce human casualties and losses in disaster situations. Recent developments in artificial intelligence [11,12] and robotics [13,14] have significantly enhanced the disaster response capabilities [15,16]. These technological innovations offer substantial promise in transforming search and rescue operations, providing hope in dire situations. Integrating advanced AI systems into disaster management procedures is a promising strategy for saving lives and mitigating the effects of these disastrous events.

In response to urgent and challenging situations, this study's primary goal is to protect lives at risk from being trapped under collapsed structures, lost in vast deserts, or stranded in harsh terrains. Utilizing advanced technology, it was proposed to develop a robot capable of locating missing individuals and transmitting their location. This approach enables effective tracking and rescue, either by the robot directly or through the automatic and adaptive dispatch of necessary assistance, thus reducing the need for human involvement. This innovative approach is central to this research and has the potential to save not only the lives of rescue personnel but also those of the survivors.

In Table 1, it can be seen that, in disaster response, humans and robots each have their own strengths and weaknesses. Humans are adaptable and experienced, but they face risks in dangerous environments and get tired. They may also struggle in tight spaces. Robots, on the other hand, can work continuously without getting tired and can go into risky areas. They can be equipped with special tools. However, they may have trouble quickly changing situations and rely on programming for decisions. Humans are good at communication, while robots use devices. By combining human skills with robot capabilities, it can improve disaster response efforts.

Table 1: Comparison of human and robotic capabilities in rescue missions, highlighting key strengths and limitations of each.

Aspect	Humans	Robots
Risk to Rescuers	High risk in hazardous environments	Low risk, can navigate dangerous areas
Physical Limitations	Limited, fueled by strength and endurance	No physical limitations, operates continuously
Adaptability	Can quickly adapt to changing situations	May face challenges in rapidly changing environments
Specialized Capabilities	Limited by human capabilities	Can be equipped with specialized sensors and tools
Decision-Making	Based on experience, intuition, and training	Rely on programming and sensor data for decision-making
Access to Confined Spaces	Limited by size and physical constraints	Can navigate tight or small spaces
Remote Operation	Not applicable	Can be operated remotely
Surveillance	Visual and auditory senses	Equipped with cameras for visual information
Communication	Verbal and non-verbal communication	Equipped with communication devices

II. Design development

A. *Proposed design*

The proposed solution is a self-centralized autonomous rescue system consisting of a drone, a central unit, and a ground vehicle. The drone will continuously stream its camera feed and location data to the central unit, which acts as the operation's brain. This central unit will receive data from the drone, perform decision-making, and conduct object recognition on the drone's stream. If a survivor is detected, the central unit will send the survivor's location to the ground vehicle, which will then proceed to provide the necessary aid. Utilizing UAVs and AI significantly enhances capabilities in post-disaster scenarios beyond human abilities. Consequently, the disaster response time will be reduced, and the probability of rescuing survivors will be much higher.

Solution requirements and criteria:

1. **Computer Vision System:** The drone must have an advanced computer vision system to identify people effectively in various conditions.

2. **Navigation and Autonomy:** Both the drone and the ground vehicle require sophisticated navigation systems for autonomous operation in unpredictable areas.

3. **Communication:** Robust communication is essential between the drone, vehicle, and base station, capable of efficient data transfer even in low connectivity areas.

4. **Safety and Reliability:** The system should include fail-safe mechanisms and thorough safety testing to ensure reliability and prevent accidents.

5. **Natural Language Processing:** Advanced natural language processing is needed for clear communication with and assessment of found individuals.

6. **Operational Endurance:** The drone and vehicle should have long battery life and energy efficiency for extended missions.

7. **Simplified Controls:** The system should feature straightforward and easy controls for efficient operation and quick response by the team.

8. **Scalability and Adaptability:** Design should be scalable and adaptable for future technological integrations.

Solution constraints:

9. **Financial Constraints**: Operating within a limited budget restricts the ability to acquire top-tier equipment and software, potentially impacting the overall effectiveness of the rescue system.

10. **Technological Limitations**: The drones and autonomous vehicles have inherent limitations in terms of weight capacity, battery life, and physical dimensions, which can limit their operational capabilities.

11. **Regulatory Compliance**: Adherence to existing laws and regulations governing the use of drones and autonomous vehicles is mandatory, which can complicate deployment in certain areas.

12. **Environmental Challenges**: Diverse weather conditions, difficult terrain, and natural obstacles pose significant challenges to the efficiency and success of rescue missions.

13. **Communication Barriers**: Limited or unreliable internet and phone connectivity in certain areas can hinder communication with the drones.

14. **Interaction with Survivors**: Challenges in interacting with survivors, including language barriers, fear, and injuries, can complicate rescue efforts.

15. **Public Perception and Trust**: Gaining public acceptance and trust is a challenge due to concerns about privacy, safety, and reliance on technology in life-saving scenarios.

16. **Operational Constraints**: The physical weight and size of the drones and vehicles may restrict their access and utility in certain environments, particularly in confined areas.

B. *Detailed high-level specifications*

After starting the operation, the central control unit (PC server) establishes communication with both the drone and vehicle to enable the transmission and reception of requests. Initially, both the drone and vehicle are instructed to calibrate their current locations as the home position $(X, Y, Z = 0, 0, 0)$. Afterwards, the vehicle enters a standby mode, awaiting further instructions. Simultaneously, the drone is tasked to perform reconnaissance within a 300m2 area in the vicinity of the central unit. During this operation, it continuously streams video back to the central unit along with real-time positional data (X, Y, Z). As illustrated in Figure 3.

The central unit, equipped with AI image analysis capabilities, specifically utilizes a Convolutional Neural Network (CNN) algorithm [17], processes the incoming video stream by capturing a snapshot every 20 seconds and analyzing these images. Upon identifying a survivor within these images, the precise coordinates are transmitted to both the drone and vehicle. In response, the drone may either broadcast a prerecorded message or engage its speaker system as determined by the situation, then maintain a hovering position near the identified survivor to scout for additional survivors.

At the same time, the vehicle is navigating through the terrain towards the survivor's location. The vehicle will either wait for a preset duration or visual confirmation via onboard cameras to ensure the survivor has entered it. Following the successful entry of the survivor, both the drone and vehicle are instructed to return to their designated home position (0, 0, 0).

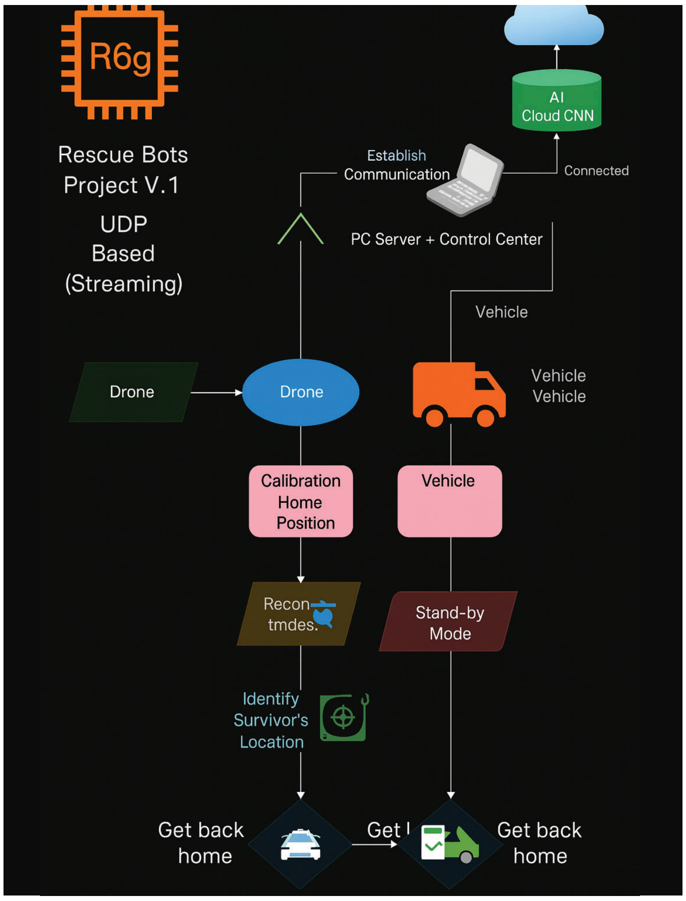


Figure 3: High-level architecture of the autonomous rescue system, showing interactions among the drone, central unit, and ground vehicle.

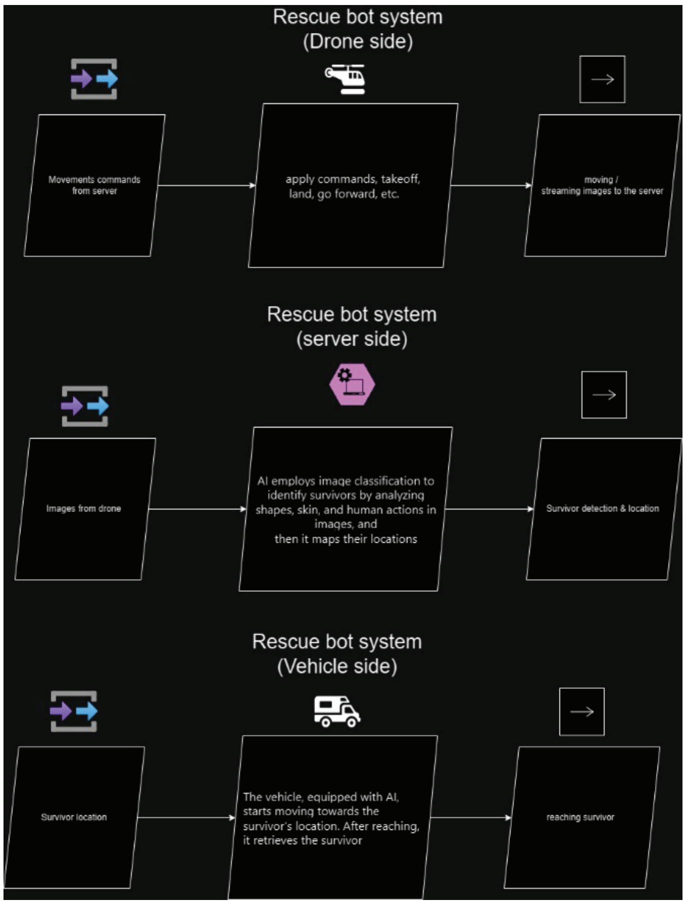

Rescue bot system (Drone side)
Movements commands from server
apply commands, takeoff, land, go forward, etc.
moving / streaming images to the server
Rescue bot system (server side)
Images from drone
AI employs image classification to identify survivors by analyzing shapes, skin, and human actions in images, and then it maps their locations
Survivor detection & location
Rescue bot system (Vehicle side)
Survivor location
The vehicle, equipped with AI, starts moving towards the survivor's location. After reaching, it retrieves the survivor
reaching survivor

Figure 4: Input–output diagram illustrating communication flows among the server, drone, and ground vehicle.

The vehicle system is designed around four key components: a Raspberry Pi 4, a webcam, a motor driver, and a pair of DC motors, as shown in Fig. 4. The Raspberry Pi 4 serves as the central processing unit, orchestrating the vehicle's overall functionality. It operates as the main controller, managing both the reception and transmission of commands to and from the central server.

Communication with the server is facilitated through a secure SSH port, enabling the Raspberry Pi to receive coordinates and other operational commands. Upon receipt of these coordinates, the Raspberry Pi directs the vehicle, through the integration of the motor driver and DC motors, to navigate towards the designated survivor's location, as shown in Fig. 5. This motor assembly is crucial for controlling the vehicle's movements, allowing for precise adjustments in direction and speed as required.

Once the vehicle arrives at the specified region or coordinates, the webcam camera, acting as an optical sensor, is activated to commence the search for the nearest survivor. The camera, in conjunction with image processing algorithms running on the Raspberry Pi, scans the area to detect and identify any individuals in distress. This combination of hardware and software enables the vehicle to fulfill its mission of reaching and assisting survivors effectively.

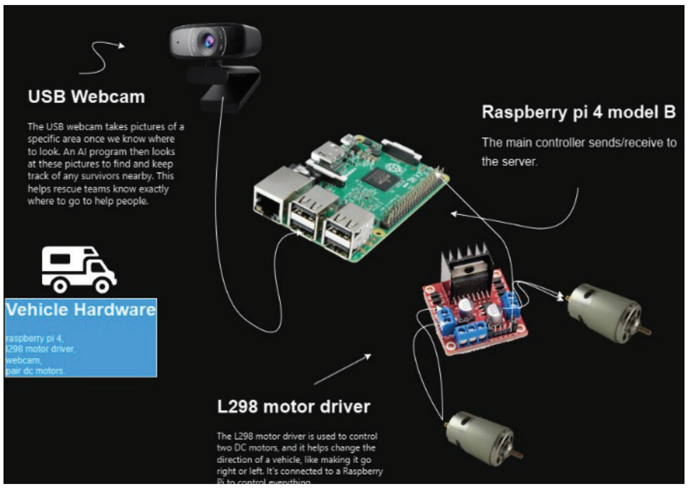

USB Webcam
The USB webcam takes pictures of a specific area once we know where to look. An AI program then looks at these pictures to find and keep track of any survivors nearby. This helps rescue teams know exactly where to go to help people.
Raspberry pi 4 model B
The main controller sends/receive to the server.
Vehicle Hardware
raspberry pi 4,
l298 motor driver,
webcam,
pair dc motors.
L298 motor driver
The L298 motor driver is used to control two DC motors, and it helps change the direction of a vehicle, like making it go right or left. It's connected to a Raspberry Pi to control everything

Figure 5: Hardware configuration of the ground vehicle, including the Raspberry Pi 4, motor driver, DC motors, and onboard camera.

The drone constitutes a central component of the rescue system, operating as the primary field unit responsible for environmental scanning and real-time data acquisition, as demonstrated in Fig. 6. It performs systematic aerial reconnaissance across the designated area and transmits continuous image streams to the server for processing and survivor identification. The drone executes flight commands issued by the control server while simultaneously returning visual data, thereby forming an essential link in the overall search-and-rescue workflow.

The drone's software architecture integrates autonomous navigation, visual detection, and communication functions into a unified operational framework. It establishes a stable connection with the aerial platform, manages live video acquisition, and employs advanced image-recognition algorithms to detect potential survivors. Upon detection, the system activates predefined alert mechanisms and initiates coordination procedures with the ground vehicle. The drone's programmed flight pattern enables systematic coverage of the search area, while the detection subsystem provides continuous analysis of captured frames to identify human figures or distress signals.

The ground vehicle is controlled through a dedicated software module designed to execute remote navigation commands securely. Communication is established via an SSH-based protocol that enables the transmission of movement instructions and mission parameters to the vehicle's onboard processor. The vehicle navigates toward specified coordinates while maintaining communication with the control server and subsequently activates its own detection procedures upon arrival at the target location. This software framework ensures reliable coordination between the aerial and ground units and supports the system's overall goal of autonomous survivor localization and assistance. This logic is exemplified in Appendices 1 and 2 by a flowchart.

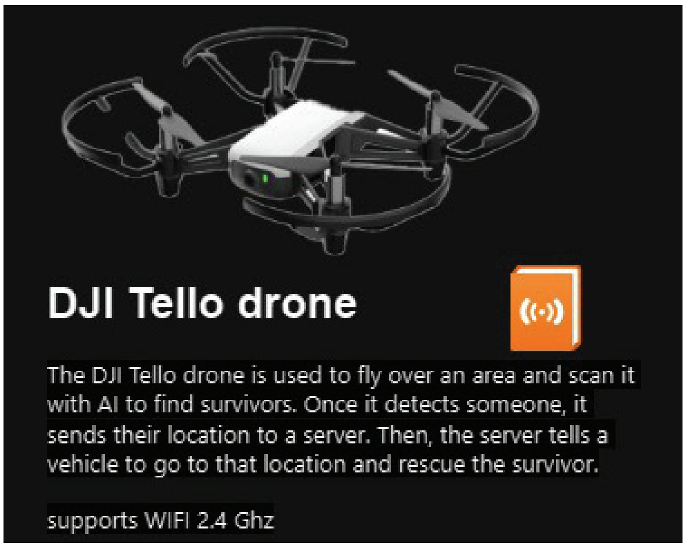

DJI Tello drone
The DJI Tello drone is used to fly over an area and scan it with AI to find survivors. Once it detects someone, it sends their location to a server. Then, the server tells a vehicle to go to that location and rescue the survivor.
supports WIFI 2.4 Ghz

Figure 6: Drone platform used for aerial imaging, showing the mounted camera and live-streaming components.

III. Realization & performance optimization

A. Planned implementation and experiments

In this innovative work, the drone plays a crucial role as a primary tool for survivor detection. It is equipped with a camera whose angle can be adjusted from 0 to 90 degrees, as illustrated in Figure 7, to provide a comprehensive bird's eye view for horizontal detection. Additionally, a specially designed vehicle, powered by a Raspberry Pi 4 and programmed using Python, acts as a secondary element. This vehicle is responsible for retrieving the survivor and transporting them to the rescue station. To implement this modification, the drone's casing was opened, the original camera connection was detached, and the camera was repositioned vertically to obtain a downward-facing, bird's-eye view. This configuration was required to accommodate the YOLO-based detection method [19] and the characteristics of the custom dataset used.

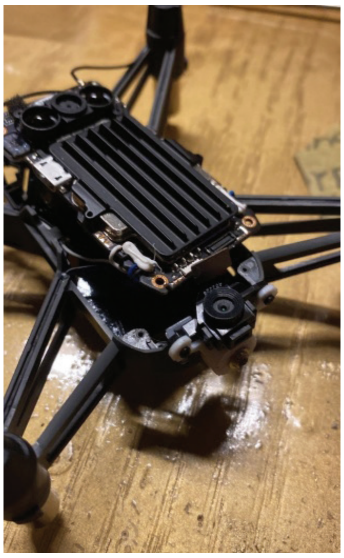


Figure 7: Modified drone prototype with a repositioned downward-facing camera to enable vertical image capture for YOLO-based detection.

The operational plan for the search and rescue system seamlessly integrates three pivotal devices: the drone *in Figure 8,* the vehicle in *Figure 9*, and a central server, each playing a critical role. Upon initiating the Python script on the server, automatic connections are established with both the drone and vehicle via a LAN network. The drone then takes off, embarking on a search within a specific randomized area, utilizing computer vision techniques. Leveraging advanced technologies like artificial intelligence and convolutional neural networks (CNN), the drone intelligently identifies the survivor's location, sending the images back to the server for further analysis.

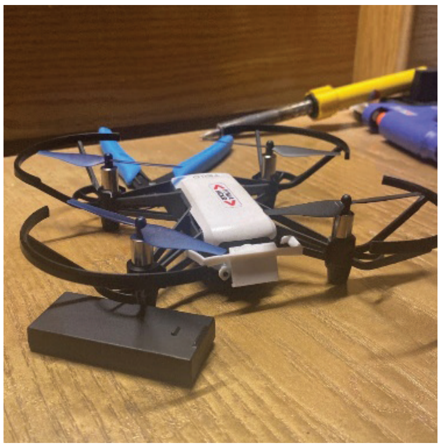


Figure 8: Final drone prototype used in field testing, incorporating the integrated camera and communication modules.

On the server side, advanced image-processing operations are executed to interpret the visual data captured by the mobile application. The incoming frames are analyzed using the YOLOv8 object-detection model, which provides rapid and highly accurate classification of pedestrians and other relevant scene elements. This stage relies on a custom dataset developed through the Roboflow platform to ensure robust detection performance. The subsequent section presents a detailed description of the dataset construction process, including annotation procedures, augmentation strategies, and quality-control measures. It also outlines the complete training pipeline implemented on Google Colab, covering model configuration, hyperparameter optimization, and evaluation methodologies.

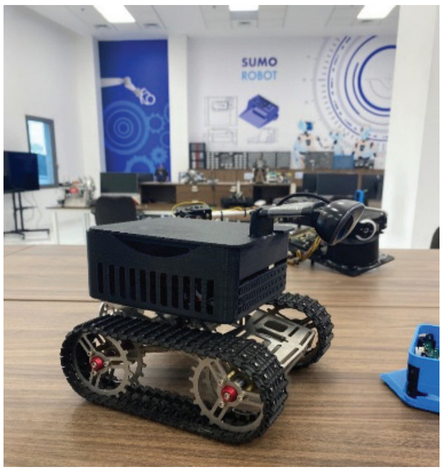
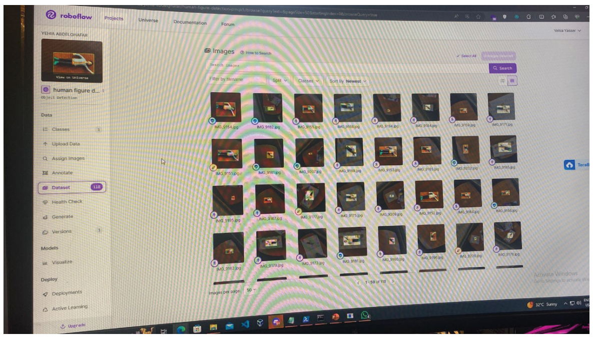


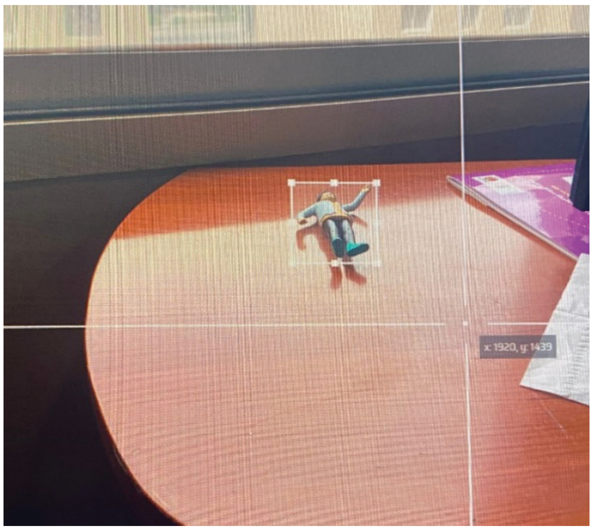
Figure 9: Ground vehicle prototype equipped with a Raspberry Pi 4, motor system, and optical sensing module.

Once the survivor's coordinates are pinpointed, the server promptly transmits this information to the vehicle. Embedded within the vehicle is a Raspberry Pi, which, upon receiving the coordinates via an SSH port, executes another Python script. This enables the vehicle to autonomously navigate towards the survivor, facilitate their rescue, and safely return to base. This sophisticated interplay between the drone, vehicle, and server exemplifies a highly interdisciplinary approach, merging robotics, computer vision, and artificial intelligence to significantly enhance the efficacy of search and rescue operations.

In this paper, the YOLOv8 deep-learning model is employed as the primary method for survivor detection. A dedicated dataset consisting of 118 images of a survivor figure, as shown in Figure 10, was developed to ensure robust model performance. Dataset construction followed a structured workflow: images were first captured under varied angles and environmental conditions to improve generalization capability; subsequently, precise annotations were performed using the Roboflow platform. Model training was conducted on Google Colab, utilizing its available GPU resources to accelerate computation and optimize learning efficiency. Following training, the model underwent extensive evaluation to verify performance in realistic operational scenarios. The overall pipeline—from dataset collection and annotation to training and validation—was computationally demanding yet essential for achieving high accuracy and reliability in survivor detection.



(a)

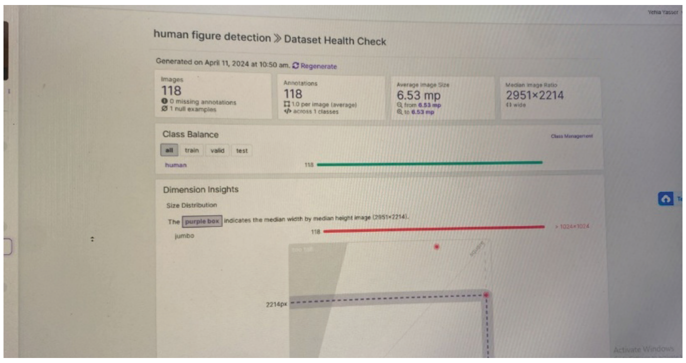


(b)

Figure 10: Annotation process used in dataset preparation: (a) sample of annotated training images; (b) example of a single annotated survivor instance.

Figure 10 displays a variety of images annotated with different poses, ensuring both accuracy and diversity in the dataset. This approach enhances the model's ability to recognize the target object survivors, in this case, across a wide range of scenarios and positions. By incorporating multiple poses, the dataset effectively trains the deep-learning model to identify survivors in various conditions and orientations, significantly improving the model's robustness and performance in real-world applications. Figure 10 shows the overall annotation process and dataset results, including Dimension Insights and the total number of trained and annotated images. This visualization offers key insights into the diversity of image sizes and the scale of data preparation, crucial for assessing the dataset's robustness and effectiveness in training the model.

Figure 11 is valuable for assessing the quality and comprehensiveness of the dataset used for training machine learning models, ensuring that the model is trained on well-rounded and representative data.



human figure detection ≫ Dataset Health Check

Generated on April 11, 2024 at 10:50 am. Regenerate

Images	Annotations	Average Image Size	Median Image Ratio
118	118	6.53 mp	2951×2214
0 missing annotations	1.0 per image (average)	from 6.53 mp	wide
1 null examples	across 1 classes	to 6.53 mp	

Class Balance

all train valid test

human 118

Dimension Insights

Size Distribution

The purple box indicates the median width by median height image (2951×2214).

jumbo 118 > 1024×1024

2214px

Figure 11: Dataset statistics generated through Roboflow, including image dimensions, class distribution, and annotation completeness.

B. *Design analysis and feedback*

Comprehensive testing has been conducted on the prototype to ensure it meets all the requirements. The first set of tests focused on key aspects of the system:

Quick response is crucial in rescue operations. Extensive testing ensured the system's quick response to disasters. As a self-centralized rescue system, decision-making regarding the survivor's location and the vehicle's movement initially caused delays. To address this, improvements in detection speed and vehicle pathing were tested as shown in Figure 12.

The prototype uses a map, so it was essential to ensure the drone's search paths stayed within the map boundaries. Multiple tests confirmed the drone remained within these borders and thoroughly searched every necessary section of the map.

Comprehensive tests checked for issues in communication between the drone, server, and vehicle. Centralized systems can fail at a single point, so the server's operation must be robust to prevent system-wide failures. The prototype was tested under connection obstacles, such as crowded connections, to assess how much connection disruption it could handle before failing.

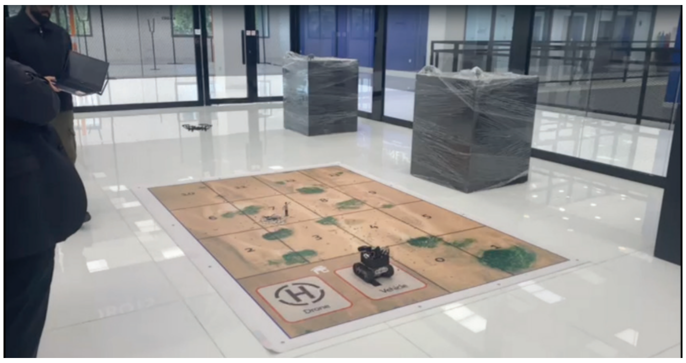
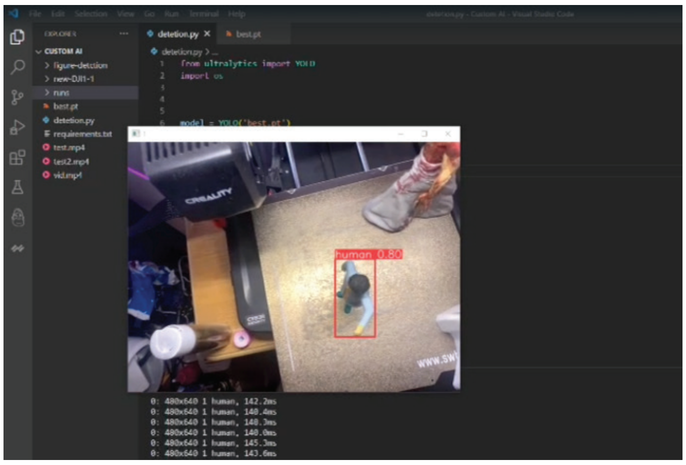


Figure 12: Experimental setup used to evaluate drone detection accuracy, communication latency, and vehicle navigation.

The drone was put in a series of trials to check the accuracy of survivor detection in various environments, as shown in Figure 13. The vehicle, using the same pre-trained model but detecting survivors from different angles, was also tested. These tests assessed how the camera angles of both the drone and the vehicle, as well as different poses of the survivor on the map, affected detection accuracy.

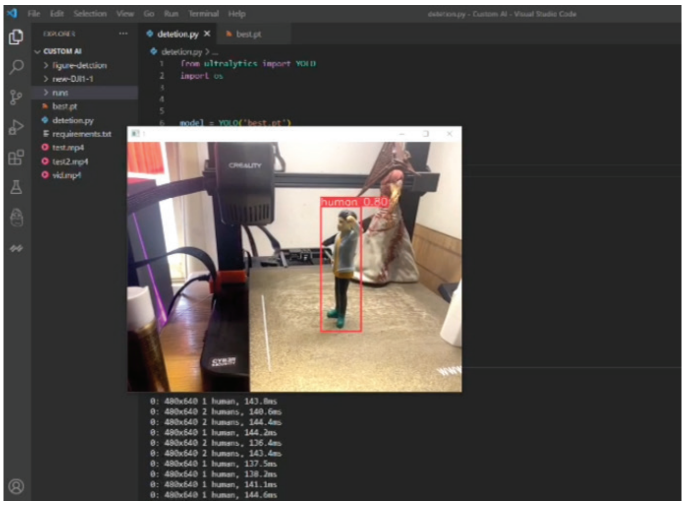


```
from ultralytics import YOLO
import os


model = YOLO('best.pt')
```

```
0: 480x640 1 human, 142.2ms
0: 480x640 1 human, 140.4ms
0: 480x640 1 human, 140.3ms
0: 480x640 1 human, 140.0ms
0: 480x640 1 human, 145.3ms
0: 480x640 1 human, 143.6ms
```

(a)



```
from ultralytics import YOLO
import os


model = YOLO('best.pt')
```

```
0: 480x640 1 human, 143.8ms
0: 480x640 2 humans, 140.6ms
0: 480x640 2 humans, 144.4ms
0: 480x640 1 human, 144.2ms
0: 480x640 2 humans, 136.4ms
0: 480x640 2 humans, 143.4ms
0: 480x640 1 human, 137.5ms
0: 480x640 1 human, 138.2ms
0: 480x640 1 human, 141.1ms
0: 480x640 1 human, 144.6ms
```

(b)

Figure 13: Experimental detection results for two test scenarios: (a) trial 1 showing successful aerial detection; (b) trial two under different survivor pose and lighting conditions.

C. Design optimization and improvements

Using Yolov9 significantly improved detection speed. However, vehicle pathing still needs refinement. Testing various pathing methods revealed that directly approaching the survivor is the fastest, but this can be problematic if obstacles are present, since the vehicle lacks a collision avoidance system. Further tests are needed to develop a reliable path assessment.

After extensive testing, the fourth pattern proved the fastest for finding survivors and returning home, especially since the map is not square. Further tests are needed to integrate SLAM [22-23], which would eliminate the need for predefined search patterns in a localized map.

Testing various communication methods showed that a LAN network connecting the vehicle and server to the central unit was the most stable. However, interference from multiple people using Wi-Fi or cellular networks can disrupt communication, necessitating further testing with alternative methods.

Several algorithms were tested as demonstrated in Figure 14, including pretrained Yolov8, a pretrained TensorFlow CNN model, custom-trained Yolov8, and a custom-trained TensorFlow CNN. The custom-trained Yolov8 was superior for the drone's stream, offering great accuracy and detection speed. For the vehicle, the pretrained TensorFlow model performed better due to the Raspberry Pi 4 limitations. Further testing on different algorithms and datasets is required.

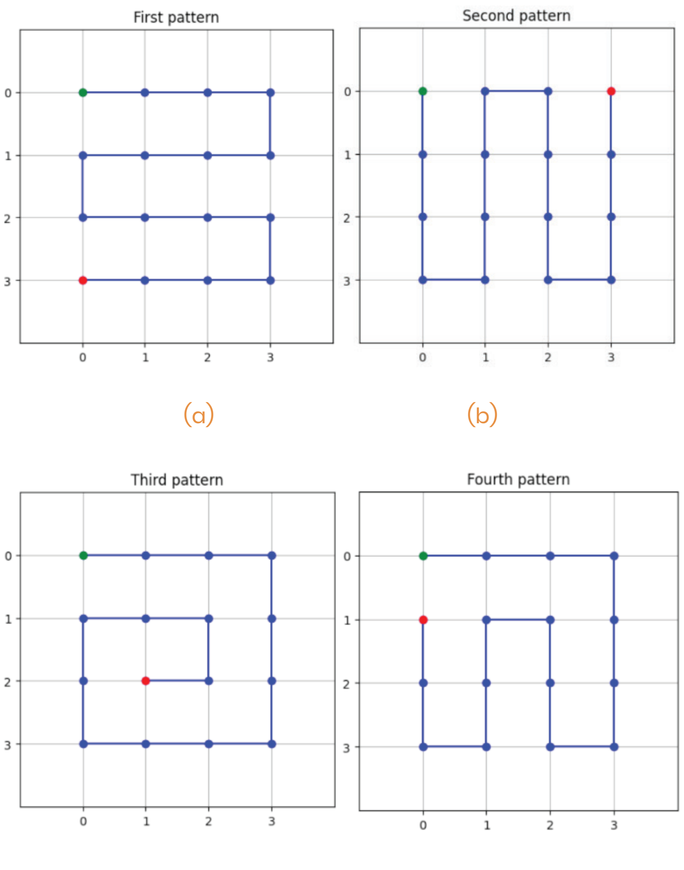


Figure 14: evaluated search strategies for drone reconnaissance: (a) row-by-row search; (b) column-by-column search; (c) outside-inward pattern; (d) hybrid approach where the drone scans the first column followed by row-by-row coverage.

IV. Conclusion & Future works

In conclusion, a self-centralized autonomous rescue system has been developed, consisting of an aerial drone, a central control unit, and a ground vehicle. The drone performs reconnaissance, identifies potential survivors, and transmits their coordinates to the control center, which subsequently directs the ground vehicle to the detected location. A custom YOLOv8 model is employed for detecting survivors or visual signals indicating a need for assistance. The ground vehicle is equipped with a Raspberry Pi 4, providing the computational capability necessary for real-time processing and navigation.

Future enhancements include the integration of drone swarms to improve search coverage and overall system efficiency. Additional sensors will be incorporated to enhance detection performance under poor visibility or challenging environmental conditions. Furthermore, the use of Simultaneous Localization and Mapping (SLAM) is planned to enable operation in previously unmapped or unknown environments, allowing the drone to construct and navigate a map in real time.

V. Appendix

Appendix1:

Autonomous drone rescue system flowchart for YOLO human detection.

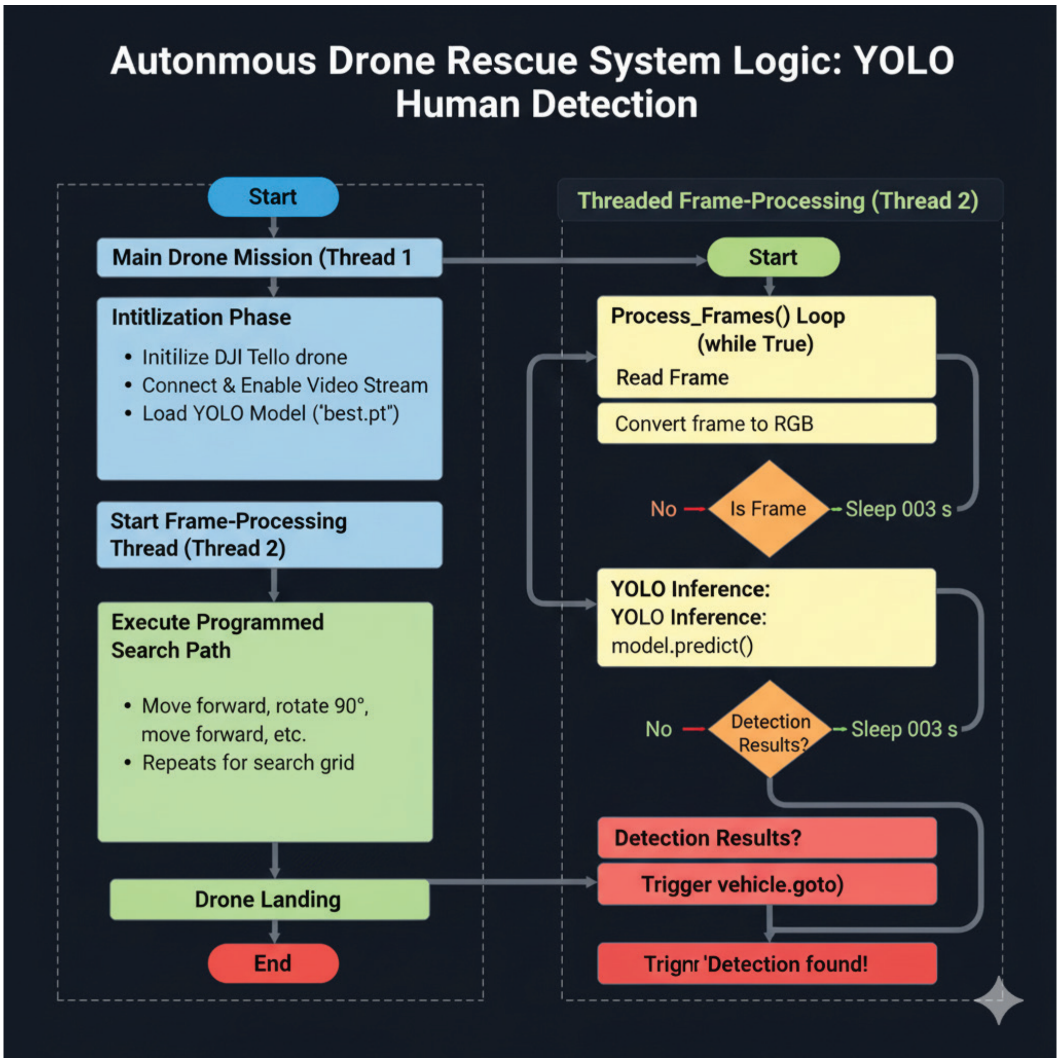

Autonmous Drone Rescue System Logic: YOLO Human Detection
Start
Main Drone Mission (Thread 1
Intitlization Phase
• Initilize DJI Tello drone
• Connect & Enable Video Stream
• Load YOLO Model ("best.pt")
Start Frame-Processing Thread (Thread 2)
Execute Programmed Search Path
• Move forward, rotate 90°, move forward, etc.
• Repeats for search grid
Drone Landing
End
Threaded Frame-Processing (Thread 2)
Start
Process_Frames() Loop (while True)
Read Frame
Convert frame to RGB
No
Is Frame
Sleep 003 s
YOLO Inference:
YOLO Inference:
model.predict()
No
Detection Results?
Sleep 003 s
Detection Results?
Trigger vehicle.goto)
Trignr 'Detection found!

Figure A1

Appendix 2:

Raspberry Pi Ground Vehicle flowchart for pedestrian detection and avoidance.

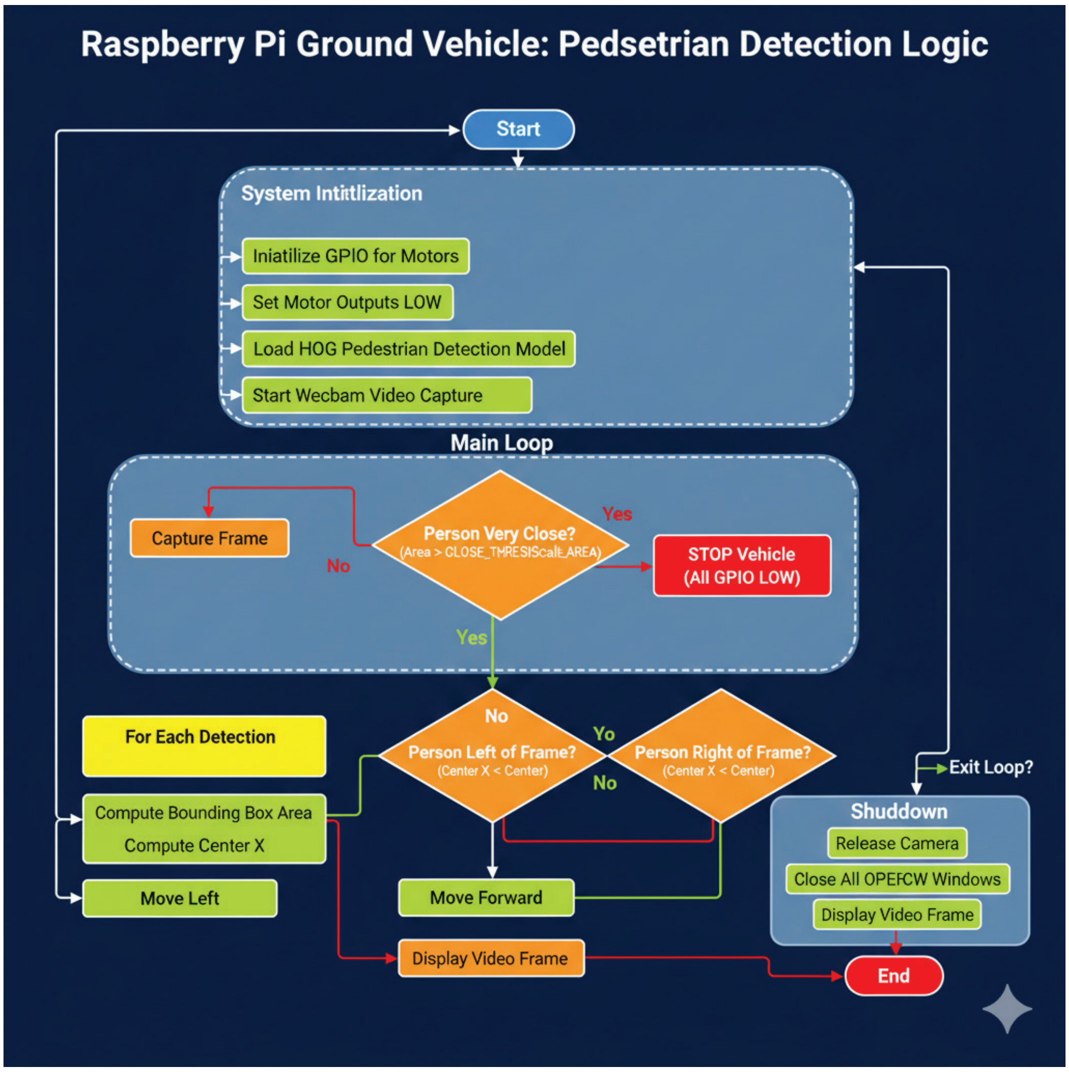


Figure A2

REFERENCES

[1] I. S. E. K. C. Liakos Amarachi Orie, "Death toll climbs to 33,000 people in Turkey-Syria earthquake," Nov. 2023. [Online]. Available: https://edition.cnn.com/2023/02/12/middleeast/deaths-turkey-syria-earthquake-intl/index.html

[2] "Global reported natural disasters by type." [Online]. Available: https://ourworldindata.org/grapher/natural-disasters-by-type

[3] B. R. Cross, "Turkey and Syria earthquakes: latest news," Nov. 2023, *British Red Cross*. [Online]. Available: https://www.redcross.org.uk/stories/disasters-and-emergencies/world/turkey-syria-earthquake

[4] K. McIntosh, "Deaths from global conflicts hit 30-year high, since 2021," *The Guardian*, 2024.

[5] J. Erdemir, M. Ghassany, N. Fernandez, and et al, "The impact of wars and natural disasters on the emergence of communicable diseases," *Ann Clin Microbiol Antimicrob*, vol. 22, no. 1, 2023.

[6] M. Sakurai and Y. Murayama, "Information technologies and disaster management – Benefits and issues -," *Progress in Disaster Science*, vol. 2, p. 100012, Jul. 2019, doi: 10.1016/j.pdisas.2019.100012.

[7] B. Mukhopadhyay, "Use of Information Technology in Emergency and Disaster Management," *American Journal of Environmental Protection*, vol. 4, no. 2, p. 101, 2015, doi: 10.11648/j.ajep.20150402.15.

[8] Y. Murayama, H. J. Scholl, and D. Velev, "Information Technology in Disaster Risk Reduction," *Information Systems Frontiers*, vol. 23, no. 5, pp. 1077–1081, Sep. 2021, doi: 10.1007/s10796-021-10204-x.

[9] R. Gaire *et al.*, "Internet of Things (IoT) and Cloud Computing Enabled Disaster Management," 2018. [Online]. Available: https://arxiv.org/abs/1806.07530?utm

[10] A. Adeel *et al.*, "A Survey on the Role of Wireless Sensor Networks and IoT in Disaster Management," 2019, pp. 57–66. doi: 10.1007/978-981-13-0992-2_5.

[11] W. Sun, P. Bocchini, and B. D. Davison, "Applications of artificial intelligence for disaster management," *Natural Hazards*, vol. 103, no. 3, pp. 2631–2689, Sep. 2020, doi: 10.1007/s11069-020-04124-3.

[12] S. K. Abid *et al.*, "Toward an Integrated Disaster Management Approach: How Artificial Intelligence Can Boost Disaster Management," *Sustainability*, vol. 13, no. 22, p. 12560, Nov. 2021, doi: 10.3390/su132212560.

[13] H. Surmann *et al.*, "Lessons from robot-assisted disaster response deployments by the German Rescue Robotics Center task force," *J Field Robot*, vol. 41, no. 3, pp. 782–797, May 2024, doi: 10.1002/rob.22275.

[14] A. Sebastian, "Soft Robotics for Search and Rescue: Advancements, Challenges, and Future Directions," *arXiv preprint arXiv:2502.12373*, 2025.

[15] J. Cani *et al.*, "TRIFFID: Autonomous Robotic Aid For Increasing First Responders Efficiency," *arXiv ID: arXiv: 2502.09379*, 2025.

[16] R. V. Kumar, M. V. D. P. Kumar, Mamatha B, and A. V. Hanuman, "Swarm Robotics for Disaster Management," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 4, pp. 5584–5589, 2024.

[17] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Trans Neural Netw Learn Syst*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[18] Ryze Tech, "Tello Drone Specifications," *Ryze Robotics*, 2025.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[21] S. Thrun and M. Montemerlo, "The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures," *Int J Rob Res*, vol. 25, no. 5–6, pp. 403–429, May 2006, doi: 10.1177/0278364906065387.

[22] Y. Bai, Y. Wang, and S. Shen, "Deep Visual-Inertial Odometry with Semantic Features for Indoor Navigation," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 1–14, 2020.

[23] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," *Int J Rob Res*, vol. 34, no. 7, pp. 1–19, Jul. 2015, doi: 10.15607/RSS.2014.X.007.

Real-time Object Detection and Diagnosis of Tomato Quality using YOLO

Youssry A. Mokhtar [1], and Essam H. Seddik [2]

[1] Arab Academy for Science, Technology, and Maritime Transport, Department of Mechanical Engineering, Alexandria, Egypt.

[2] Arab Academy for Science, Technology, and Maritime Transport, Department of Mechanical Engineering, Alamein, Egypt.

y.a.mokhtar@student.aast.edu, essam.seddik@aast.edu

Received: 04 November 2025 Accepted: 16 December 2025 Published: 23 December 2025

Abstract

Tomato quality plays a critical role in both customer satisfaction and the efficiency of post-harvest processing. Traditional sorting and grading methods are labor-intensive, subjective, and unsuitable for high-throughput operations. This study proposes a smart AI-based vision system capable of real-time classification of tomato quality into three classes: fresh, damaged, and unripe. The system employs advanced deep learning techniques, specifically a custom-trained YOLO object-detection model, to analyze key visual attributes such as color, texture, and surface defects. A diverse, labeled custom dataset of tomato images was collected to train and evaluate the model. This dataset included tomato images with the three health conditions according to the output classes of the neural network. The results show that the system has achieved high accuracy and strong robustness across varying lighting conditions and backgrounds, making it suitable for deployment in real agricultural and industrial environments. By enabling fast, automated, and objective quality assessment, the proposed system significantly enhances the reliability and efficiency of tomato grading and contributes to improved food supply chain management.

Key-words: Tomato Quality Classification, Computer Vision, YOLO, Deep Learning

I. Introduction

A. Problem statement

With an annual production of about 186 million metric tons, tomatoes are one of the most consumed vegetables in the world (FAO, 2022). Food safety, waste reduction, and consumer pleasure all depend on maintaining the quality after harvest. Sorting and grading tomatoes according to their ripeness, freshness, and physical flaws is still quite difficult, though, particularly in high-volume production settings like farms, warehouses, and supermarkets. With an annual production of about 186 million metric tons, tomatoes are one of the most consumed vegetables in the world (FAO, 2022). Food safety, waste reduction, and consumer pleasure all depend on maintaining the quality after harvest. Sorting and grading tomatoes according to their ripeness, freshness, and physical flaws is still quite difficult, though, particularly in high-volume production settings like farms, warehouses, and supermarkets [1].

B. Literature review

Since Deep Learning (DL) showed the best results in the literature review, a modified YOLOv5 object detection model was used to solve this problem. The DL model categorizes tomatoes according to their color, outer look, damage symptoms, and obvious defects after being trained on a dataset of more than 2,000 annotated tomato photos. The suggested system can be used in robotic arms or conveyor belts in agricultural environments and is built for real-time response [2]. Mukesh Dalal and Payal Mittal systematically reviewed recent advancements in using deep learning models (YOLO v9, v10, EfficientDet, Transformer-based models, and hybrid frameworks) for real-time object detection in agriculture (crops, fruits, diseases), noting enhanced precision but emphasizing challenges like data scarcity and the need for edge computing [3]. Campos Soto, Rojas Pino, and Aguilera Carrasco systematically review the use of deep learning models (CNNs, R-CNN, YOLO) for fruit classification and detection, emphasizing challenges like data scarcity and occlusion while proposing future research into multi-modal integration and computational efficiency [4]. Tan et al. review the application of deep learning in fruit and vegetable picking robots, summarizing key technologies in visual perception, path planning, and end effector control to highlight the shift towards intelligent, automated harvesting [5]. Dewi, Thiruvady, and Zaidi propose a novel Fruit Classification System that applies Neural Architecture Search (NAS) to automatically refine the deep learning network topology, achieving a superior 99.98% mAP for classifying 15 distinct fruit categories [6].

The study by Ioannis D. Apostolopoulos, Mpesi Tzani, and Sokratis I. Aznaouridis proposes a novel, generalizable machine learning (ML) model—specifically leveraging Vision Transformers (ViT)—to objectively assess fruit quality (distinguishing between good and rotten) across various fruit types [7]. The review by Ignacio Rojas Santelices, Cano, Moreira, and Peña Fritz thoroughly analyzes Artificial Vision Systems for fruit inspection and classification, categorizing methodologies by algorithms (e.g., CNN, ANN) and features (Color, Shape, and Texture) to detail the field's current state and common practices for automated quality control [8]. Sarron et al. utilized a YOLOv5 network for mango yield estimation across diverse orchards, finding that correcting for occlusion and detection errors required incorporating categorical covariates (region, cropping system) into the linear model to significantly improve generalization (R^2 from 0.34 to 0.66) [9]. Wang, Fang, Mo, Gan, and Sun reviewed deep learning models for tomato ripeness detection, noting that while existing YOLO-based methods offer high accuracy in controlled settings, they often fail in complex outdoor environments due to reliance on substantial memory and computational resources. This led them to propose the lightweight YOLOv11-MHS model to achieve high precision with reduced overhead in challenging agricultural scenes [10]. Wu, Huang, Song, and Zhou (2025) reviewed deep learning models for tomato ripeness detection, noting that while existing YOLO-based methods offer high accuracy in controlled settings, they often fail in complex outdoor environments due to reliance on substantial memory and computational resources. This led them to propose the lightweight YOLO-PGC model to achieve high precision with reduced overhead in challenging agricultural scenes [11]. The literature review by Wang, Xu, Hu, Zhang, Li, Zhu, and Liu identifies that accurate tomato yield estimation is challenging in field environments due to fruit occlusion and overlap, which limit the performance of traditional and older machine vision techniques. Consequently, the study emphasizes the need for lightweight, deep learning-based networks (like their improved YOLO11n) that can maintain high detection accuracy while being efficient enough for real-time deployment on edge computing devices [12].

II. Dataset

A. *Data collection*

In this study, a custom image dataset of tomatoes was collected and annotated to add to the YOLO's COCO dataset to include different tomato health conditions. The modified dataset was used to train and evaluate the proposed AI vision system. The dataset is divided into three quality-based classes.

Figure 1 displays an example of each of the three conditions.

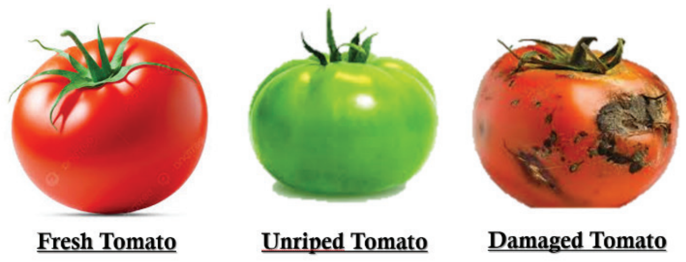


Figure 1: Dataset classes.

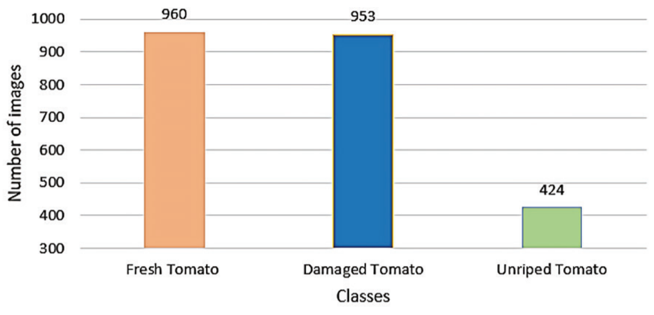


Figure 2: Dataset distribution.

The dataset was designed to reflect real-world conditions, including variations in lighting, background, and angle. As shown in the distribution, the number of unripe tomato samples is relatively smaller than the other two categories. This imbalance is due to the distinct visual features of unripe tomatoes, particularly their color and surface characteristics, which differ significantly from both fresh and damaged tomatoes. These unique features result in distinct feature extraction patterns, allowing the model to effectively learn their representations with fewer samples. Nonetheless, data augmentation techniques such as flipping, rotation, and contrast adjustment were applied to improve model robustness and mitigate class imbalance.

Figure 2 displays a bar chart titled "Dataset Classes", which illustrates the distribution of images across three tomato categories: *f*resh, *d*amaged, and *u*nripe *t*omatoes. The dataset comprises 960 images of fresh tomatoes and 953 images of damaged tomatoes, indicating that these two classes are nearly equivalent in size. Conversely, the unripe tomato class contains only 424 images, representing a substantially smaller proportion of the dataset. This disparity reveals a noticeable class imbalance, with the unripe tomato category being significantly underrepresented.

The use of color-coded bars, clearly labeled axes, and explicit numerical values enhances the clarity of this distribution.

Emphasizing the importance of applying data augmentation or other balancing techniques prior to model development.

B. Annotation

All photos were labelled using the Roboflow annotation platform, which provides a robust toolset and user-friendly interface for labelling computer vision datasets, to prepare the dataset for training and evaluation. A bounding box and one of three class labels: fresh, damaged, or unripe, were manually applied to each tomato instance in the pictures. Trained annotators carried out the labelling, visually examining each tomato to determine its class based on color, surface texture, and obvious flaws. While built-in features like auto-zoom, keyboard shortcuts, and error checking expedited the annotation workflow, Roboflow's integrated label management system helped guarantee consistent labeling throughout the dataset. To reduce human bias and mislabeling, annotation quality was confirmed by manual inspection and cross-validation by a second reviewer. A total of 2,337 labelled images were produced. Roboflow's dataset management capability was then used to export and version-control these annotations, ensuring repeatability and traceability during the model training procedure. Before exporting the finished dataset, Roboflow was also utilized to carry out data augmentation tasks, including flipping, rotation, and brightness/contrast correction, directly within the platform.

III. Training

Building dependable and broadly applicable machine learning models requires dividing the dataset into subsets for testing, validation, and training. To measure the model's performance in the actual world, this procedure makes sure that it learns from one piece of data, is adjusted using another, and is then tested on entirely unseen data.

The model was trained to identify characteristics and patterns linked to each class (Fresh, Damaged, and Unripe) using the training data. During training, the validation set was used to track the model's performance and adjust hyperparameters to assist in avoiding overfitting. In order to ensure an objective assessment of the model's performance in the real world on unknown data, the testing set was finally set aside for the final review.

A. Checking integrity and fairness of the evaluation process

The dataset was divided stratified by class in order to preserve the fairness and integrity of the assessment procedure. This indicates that, as opposed to having any subset dominated by a single tomato quality class, each subset (training, validation, and test) includes a representative portion of all three tomato quality classes. For instance, the training set is not skewed towards fresh tomatoes, and the test set is not limited to unripe tomatoes. The model's robustness and practical application are enhanced by this balanced distribution, which guarantees that it learns equally from each category and is assessed on its capacity to generalise across all tomato quality kinds.

IV. Computer vision model

A. Model selection

When developing an image-based AI system, choosing the right computer vision model is essential since it has a direct impact on the solution's accuracy, speed, and overall performance. The model must be able to reliably detect small items, identify subtle surface flaws, and function consistently in a variety of lighting and background settings for this project, which entails the real-time classification of tomatoes into fresh, damaged, and unripe categories.

A variety of model families is frequently employed in computer vision tasks, such as object identification models that integrate localization and classification, semantic segmentation networks for pixel-level comprehension, and convolutional neural networks (CNNs) for picture classification. Object detection models are most suited for this task because they involve recognizing and categorizing several tomatoes in a single image.

Among these, state-of-the-art models such as YOLO (You Only Look Once), Faster R-CNN, and SSD (Single Shot Multibox Detector) are widely adopted.

B. YOLO

YOLO (You Only Look Once) is a state-of-the-art, real-time object detection algorithm that reframes object detection as a single regression problem, rather than the traditional two-step approach of region proposal followed by classification. Unlike older methods like R-CNN or Faster R-CNN that generate multiple candidate regions and then classify them separately, YOLO processes the entire image in one forward pass through a neural network, making it extremely fast and efficient.

The workflow of this research began by splitting the input image into a 13 x 13 grid, depending on the model version. A set number of bounding boxes within each grid cell was then predicted. The model produced a few parameters for each box, including the width and height of the box in relation to the image, the (x, y) coordinates of the box's centre in relation to the cell, and a confidence score that indicates the likelihood of an object being present as well as the accuracy of the predicted box. Furthermore, class probabilities for the detected object—such as whether it is an unripe, damaged, or fresh tomato—are output by each grid cell. The final detection confidence for each object is calculated by multiplying these class probabilities by the bounding box confidence scores.

Once all grid cells and bounding boxes have been predicted, YOLO uses a method known as Non-Maximum Suppression (NMS) to remove overlapping or redundant boxes, leaving only the ones with the highest confidence scores. This procedure minimizes false positives and guarantees that each object is detected once. The model learns by minimizing a mixed loss function that has elements for objectless confidence (determining whether an object exists), classification accuracy (making the right class prediction), and localization accuracy (bounding box regression).

Compared to region-based methods, YOLO better captures global context and spatial linkages since it examines the entire image at once. Because of its architecture, YOLO is incredibly quick, able to operate in real-time (30+ FPS), and appropriate for uses such as security monitoring, autonomous driving, and, in this instance, real-time tomato quality detection. YOLO is the perfect option for smart agricultural systems that need real-time automated decision-making because of its speed-accuracy balance and capacity to identify several items in a single frame.

C. YOLOv5

YOLOv5, created by Ultralytics, is a highly favoured and extensively used object identification model because of its performance, speed, and adaptability. In order to balance accuracy with inference speed, YOLOv5 is implemented in PyTorch and comes in five primary pre-configured model sizes: YOLOv5n (Nano), YOLOv5s (Small), YOLOv5m (Medium), YOLOv5l (Large), and YOLOv5x (Extra Large). Although the depth and width of each version vary, they are all based on the same architecture, which ultimately influences the number of layers and channels as well as the size, accuracy, and speed of the model.

D. YOLOv5n

The YOLOv5n (Nano) model was chosen for this project because it strikes a great mix between speed, efficiency, and tolerable accuracy. With roughly 1.9 million parameters, YOLOv5n is the lightest version of the YOLOv5 family. This makes it ideal for real-time applications on devices with limited resources, like the Raspberry Pi, NVIDIA Jetson, or other edge computing platforms frequently found in agricultural settings. The key architectural advantages of the YOLO family, such as quick inference, end-to-end object recognition, and high spatial awareness, are still present in YOLOv5n despite its diminutive size. Even with different lighting and backdrop conditions, it can identify and categorize several tomato instances in a single image. Even while it is not as accurate as larger models like YOLOv5m or YOLOv5x, in well-structured, high-quality datasets, the performance loss is negligible. YOLOv5n is an ideal solution for on-site, automated quality assessment where speed and deployability are more important than slight precision gains. In this application, it successfully detected and classified tomatoes into fresh, damaged, and unripe categories with dependable accuracy and low latency.

The provided ***Figure 3*** illustrates the process of object detection using a system like YOLO (You Only Look Once). It shows an input image divided into an $ S\times S$ grid, where each cell predicts bounding boxes with confidence scores. A class probability map determines the object class for each grid cell, ultimately leading to the final tomato detection results with precise bounding boxes.

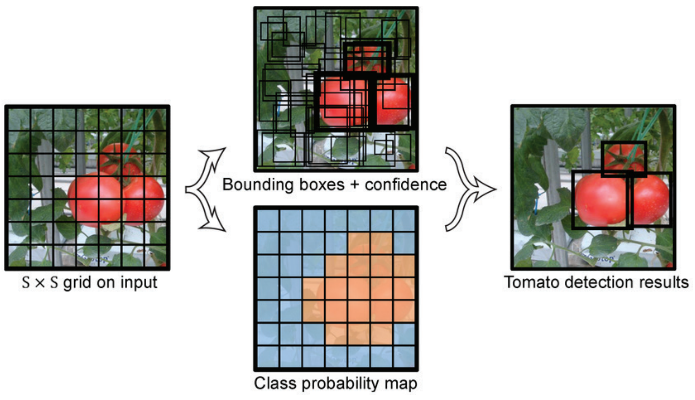


Figure 3: YOLO feature extraction.

E. YOLOv5n Layer architecture

The YOLOv5n (Nano) architecture is suitable for edge devices and embedded systems used in agricultural applications, as shown in ***Figure 3,*** because it offers real-time object detection with low computational overhead. The three main parts of it are the head, neck, and Backbone. To improve early-stage efficiency, the Backbone starts with a special Focus layer that slices the image, boosts the channel depth, and reduces spatial dimensions. The Backbone oversees extracting visual information from the input image.

Convolutional layers used in conjunction with Batch Normalization and the SiLU activation function—which provides a smoother gradient flow than ReLU—come next. The usage of Cross Stage Partial (CSP) Bottlenecks, which divide the feature map into two paths—one undergoing transformation and the other preserved—and then concatenate them to improve gradient flow and minimize computation, is a crucial component of the Backbone. A layer called Spatial Pyramid Pooling (SPP) completes the Backbone. It uses max-pooling at various scales to gather local and global information for improved object representation. The next step involves the use of convolutional layers in combination with Batch Normalisation and the SiLU activation function, which offers a smoother gradient flow than ReLU.

A key part of the Backbone is the use of Cross Stage Partial (CSP) Bottlenecks, which split the feature map into two paths, one of which is undergoing transformation and the other of which is maintained. These paths are then concatenated to enhance gradient flow and reduce computation. The Backbone is completed by a layer known as Spatial Pyramid Pooling (SPP). For better object representation, it collects local and global data using max-pooling at different sizes.

Figure 4 illustrates two key architectural modules of YOLOv5: the C3 (Cross Stage Partial Bottleneck) module and the SPPF (Spatial Pyramid Pooling - Fast) module. The C3 module is the primary processing unit, designed to improve gradient flow and reduce computation by splitting and concatenating feature paths. The SPPF module, used in the Backbone, efficiently gathers multi-scale global and local contextual information via a sequential process. pooling operations.

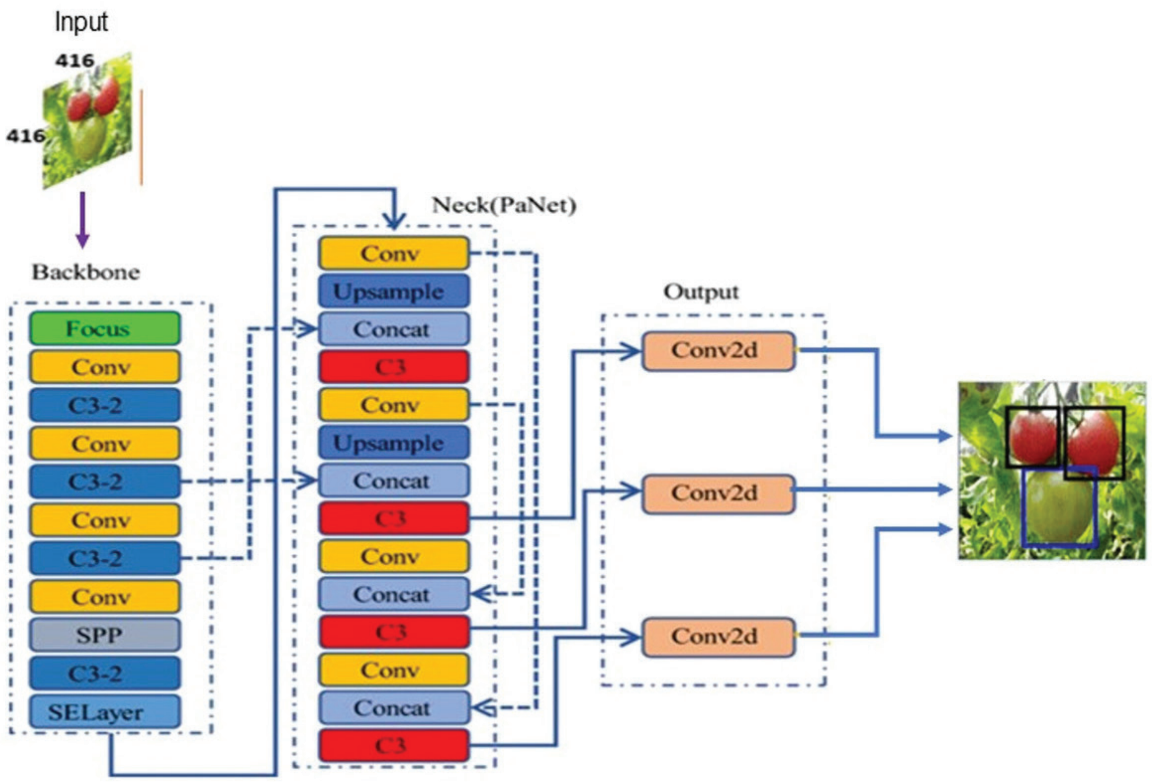


Figure 4: YOLOv5 layers.

This Precision–Confidence Curve shown in ***Figure 5*** shows how the precision of your tomato classification model changes as you adjust the confidence threshold. The x-axis represents the model's confidence level, and the y-axis represents precision, which measures how many predictions made above that confidence level is correct. The curves for Fresh and Unripe tomatoes (orange and green) stay high across almost all confidence levels, meaning the model identifies these classes very accurately even when its confidence is low. In contrast, the Damaged Tomato curve (light blue) remains low at lower confidence levels and increases only gradually, showing that this class is more difficult for the model and produces more false positives. The thick blue curve represents overall precision across all classes, and it reaches perfect precision (1.00) at a confidence of about 0.924. Overall, the plot highlights strong performance for Fresh and Unripe tomatoes and weaker consistency for Damaged tomatoes, helping you choose an appropriate confidence threshold depending on whether you prefer higher accuracy or more detections.

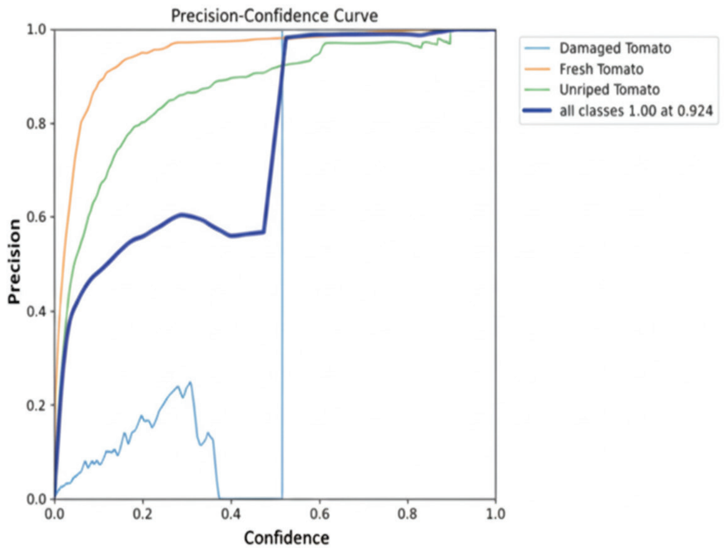


Figure 5: Precision-confidence curve.

The figure below (***Figure 6***) presents a normalized confusion matrix that evaluates the performance of a classification model across four categories: Damaged Tomato, Fresh Tomato, Unripe Tomato, and Background. The diagonal values represent correct predictions, showing that the model performs exceptionally well for Fresh Tomato, with a precision of 0.98, and achieves strong accuracy for Unripe Tomato and Damaged Tomato, with scores of 0.86 and 0.70, respectively. The matrix also indicates that a small proportion of Damaged Tomato images (0.03) and Unripe Tomato images (0.11) were misclassified as background, while minor confusion is observed between the tomato classes, such as 0.01 of unripe tomatoes predicted as damaged and 0.02 of fresh tomatoes predicted as background. Overall, the matrix highlights that while the model demonstrates high accuracy for most categories, some misclassification occurs, particularly with background and visually similar tomato classes—indicating areas where further model refinement or dataset enhancement may be beneficial.

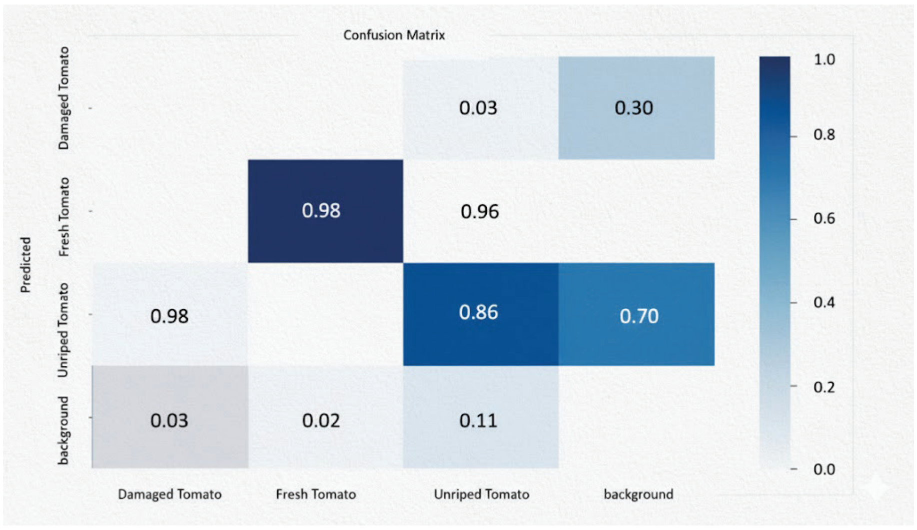


Figure 6: Confusion matrix.

V. Model performance evaluation

A. YOLOv5n model performance evaluation

Precision – 97.3% The percentage of anticipated positive cases (detections) that turn out to be accurate is known as precision. In this case, a precision of 97.3% indicates that the model is **97.3% accurate** in predicting whether a tomato is fresh, damaged, or unripe. Because of its great precision, the model seldom mislabels or mistakenly classifies non-tomato objects as tomatoes (false positives). This high precision is vital, as it ensures that when the model signals a detection, the **robotic arm** can trust the classification and act with high confidence, minimizing false picks and maximizing process reliability.

Recall – 63.1% Recall measures the model's ability to detect all actual objects. A Recall of 63.1% means that the model successfully detects about 63 out of every 100 actual tomatoes present in the images. This suggests that some objects are being missed. This low Recall reveals a significant limitation: the **robot** is missing nearly 4 out of every 10 available tomatoes due to challenges like occlusion or lighting. Therefore, while the harvested crop will be accurately sorted, the overall yield collection will be substantially incomplete, necessitating further optimization before full **robot** deployment.

B. Model performance test

Figure 7 shows a single tomato with visible signs of damage. The tomato surface has dark spots, bruising, and areas of decay, indicating biological or physical deterioration. This crucial detection is necessary to prevent the robotic arm from selecting and harvesting a compromised fruit. These features—such as discoloration, wrinkling, and surface lesions—are commonly associated with damaged or spoiled tomatoes.

Figure 8 shows the output of the tomato detection model, where two tomatoes are identified and classified using bounding boxes. The tomato on the left is labeled "Fresh Tomato" with 79% confidence, indicating it is ready for the robot to pick, while the tomato on the right is labeled "Unripe Tomato" with 94% confidence, signaling it should be ignored. This demonstrates the model's ability to detect multiple tomatoes in one image and accurately distinguish between different ripeness stages, effectively acting as the **robot's** 'eyes' for selective harvesting.

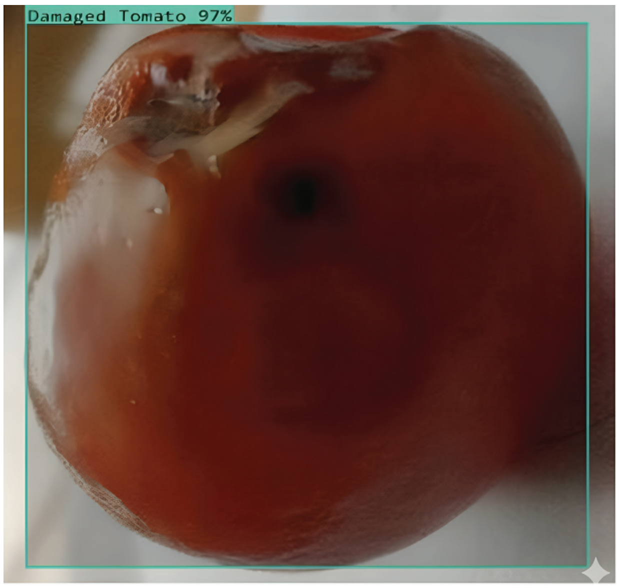


Figure 7: Damaged tomato detection.

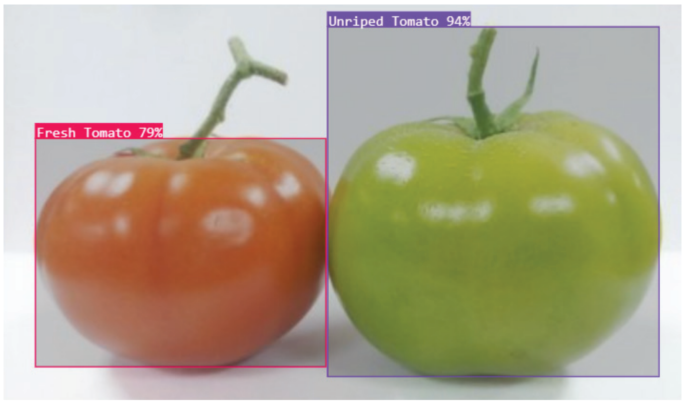


Figure 7: Multi-label tomato detection.

VI. Conclusion

This paper presented a real-time AI-based computer vision system for automated tomato quality detection using a lightweight YOLOv5n object detection model. The proposed approach addressed the limitations of traditional manual grading by enabling fast, objective, and scalable classification of tomatoes into fresh, damaged, and unripe categories under diverse lighting and background conditions. A custom annotated dataset reflecting real-world variability was developed to train and evaluate the model. Experimental results demonstrated high precision (97.3%), indicating strong reliability in correct classification and low false-positive rates, which are critical for automated harvesting and sorting applications. While the recall rate (63.1%) revealed challenges in detecting all tomato instances—particularly under occlusion and complex scenes—the system proved effective as a real-time decision-support tool for quality assessment in smart agricultural environments.

Future improvements will focus on enhancing Recall and overall robustness through dataset expansion, improved class balancing, and additional data augmentation using images collected from real harvesting fields. Further research may explore advanced or hybrid deep learning architectures, including newer YOLO variants and attention-based models, to better capture subtle surface defects while maintaining real-time performance on edge devices. Additionally, model optimization techniques such as pruning and quantization can be employed to reduce computational overhead. Integrating the proposed vision system with robotic manipulators or conveyor-based sorting platforms, along with depth sensing and grasp planning, represents a key step toward fully autonomous tomato harvesting and grading systems, contributing to reduced post-harvest losses and improved efficiency in smart agriculture.

REFERENCES

[1] J. A. Clark, "Automation in horticulture: The future of crop grading and handling," 2022. [Online]. Available: https://www.cabidigitallibrary.org/doi/pdf/10.5555/20220163943

[2] M. D. Forecast, "Tomato Market Size, Share, Trends & Growth Report, 2033," 2025. [Online]. Available: https://www.marketdataforecast.com/market-reports/tomato-market

[3] M. Dalal and P. Mittal, "A Systematic Review of Deep Learning-Based Object Detection in Agriculture: Methods, Challenges, and Future Directions," *Computers, Materials & Continua*, vol. 84, no. 1, pp. 57–91, 2025, doi: 10.32604/cmc.2025.066056.

[4] S. Espinoza, C. Aguilera, L. Rojas, and P. G. Campos, "Analysis of Fruit Images With Deep Learning: A Systematic Literature Review and Future Directions," *IEEE Access*, vol. 12, pp. 3837–3859, 2024, doi: 10.1109/ACCESS.2023.3345789.

[5] Y. Tan, X. Liu, J. Zhang, Y. Wang, and Y. Hu, "A Review of Research on Fruit and Vegetable Picking Robots Based on Deep Learning," *Sensors*, vol. 25, no. 12, p. 3677, Jun. 2025, doi: 10.3390/s25123677.

[6] C. Dewi, D. Thiruvady, and N. Zaidi, "Fruit Classification System with Deep Learning and Neural Architecture Search," 2024.

[7] I. D. Apostolopoulos, M. Tzani, and S. I. Aznaouridis, "A General Machine Learning Model for Assessing Fruit Quality Using Deep Image Features," *AI*, vol. 4, no. 4, pp. 812–830, Sep. 2023, doi: 10.3390/ai4040041.

[8] I. Rojas Santelices, S. Cano, F. Moreira, and Á. Peña Fritz, "Artificial Vision Systems for Fruit Inspection and Classification: Systematic Literature Review," *Sensors (Basel)*, vol. 25, p. 1524, Dec. 2025, doi: 10.3390/s25051524.

[9] J. Sarro *et al.*, "Fruit yield estimation using image analysis is also about correcting the number of detections | International Society for Horticultural Science," 2023. [Online]. Available: https://www.ishs.org/ishs-article/1360_42

[10] D. WANG, Z. FANG, M. MO, J. GAN, and Z. SUN, "Tomato ripeness detection method based on improved YOLOv11 lightweight model," *Front Agric Sci Eng*, vol. 13, 2025, doi: 10.15302/j-fase-2025657.

[11] Q. Wu, H. Huang, D. Song, and J. Zhou, "YOLO-PGC: A Tomato Maturity Detection Algorithm Based on Improved YOLOv11," *Applied Sciences*, vol. 15, no. 9, p. 5000, Apr. 2025, doi: 10.3390/app15095000.

[12] A. Wang *et al.*, "Tomato Yield Estimation Using an Improved Lightweight YOLO11n Network and an Optimized Region Tracking-Counting Method," *Agriculture (Switzerland)*, vol. 15, no. 13, 2025, doi: 10.3390/agriculture15131353.

مركز الأكاديمية للنشر

ACADEMY Publishing Center

RIMC

Robotics : Integration, Manufacturing and Control

VOLUME 2, ISSUE 2, DECEMBER 2025 - E-ISSN 3009-7967