International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

EYE ROBOT: A UNIFIED FRAMEWORK FOR OBJECT DETECTION AND GESTURE-BASED COMMUNICATION

J Vijay Gopal[1], Dr. T.S Subashini[2], and Dr.K. Srinivas Rao[3]

[1]Research Scholar, Department of Computer Science Engineering , Annamalai University , Chidambaram, India.
[2]Professor, Department of Computer Science Engineering , Annamalai University , Annamalai University, Chidambaram, India.
[3]Professor, Department of Computer Science and Engineering , Mlr Institute of Technology , Dundigal,Hyderabad, india.

I Abstract

The EyeRobot is a vehicle that combines a land rover and a drone, designed for applications in challenging war terrains. It is intended to assist troops and commanders in making faster, intelligent decisions while in the field. The autonomous vehicle is capable of covering vast expanses and identifying known and unknown objects, enhancing situational awareness in combat. Here, we present five most important skills of the system: First, it is able to identify common objects—such as trees, automobiles, or streetlights—and assist the robot in navigating through dense or urban surroundings safely. Second, it is able to identify rotations of an object, assisting in the interpretation of hand gestures and other non-verbal signals.

The third function is dedicated to identifying possibly dangerous objects, such as sharp objects or objects resembling weapons. This helps the system identify threats at an early stage and alert alarms to prevent possible harm. The fourth capability includes detecting and identifying faces, which helps in restricting access to secure zones by verifying authorized individuals.

Finally, it is able to read Morse code through flashing eyes or flashing lights, i.e., when too dark or too loud to speak.

These capabilities are combined into a single system, which we evaluated using simulations and testbeds. The EyeRobot generally has clear potential for improving object detection and team communication in harsh combat environments. In short, this module gives the Eye Robot a powerful and complete hearing system that not only hears and understands the sound and also knows where it comes from and reacts accordingly. With the combination on the basis of communication theory, emotional audio interpretation, and geometric signal analysis, we here propose a robust basis for human-aware, future robot interaction.

A. keywords:

EyeRobot, Object Detection, Gesture Recognition, Dangerous Object Identification, Weapon Detection, Face Recognition, Morse Code Interpretation, Blink Pattern Detection, Drone and Rover Integration, Au-

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

tonomous Navigation, War Terrain Deployment, Human-Robot Interaction, Situational Awareness, Emotional Audio Interpretation, Communication Theory, Geometric Signal Analysis, Multimodal Communication.

II Introduction

Autonomous systems have revolutionized contemporary warfare, surveillance, and reconnaissance by allowing real-time object and gesture recognition. The EyeRobot, a hybrid of a drone-land rover, is designed to recognize, analyze, and detect objects in rapidly changing environments—most particularly in battlefields where quick decisions can be a game-changer.

Object detection is at the heart of system. It is used for detecting and classifying objects in the surroundings like trees, cars, roads, buildings, and obstacles in city and harsh environments. While Canny edge detection and morphological processing are adequate with the help of traditional image processing methods, they are prone to fail in high-speed, uncontrolled environments.

To counteract that, EyeRobot combines models such as deep learning models such as YOLO (You Only Look Once) and R-CNNs which are sophisticated . They enable quick and accurate detection by inspecting video streams from in-car cameras. A pipeline based on CNN extracts feature from these images and classifies them using pre-trained models, rendering it more reliable in diverse real-world environments. But

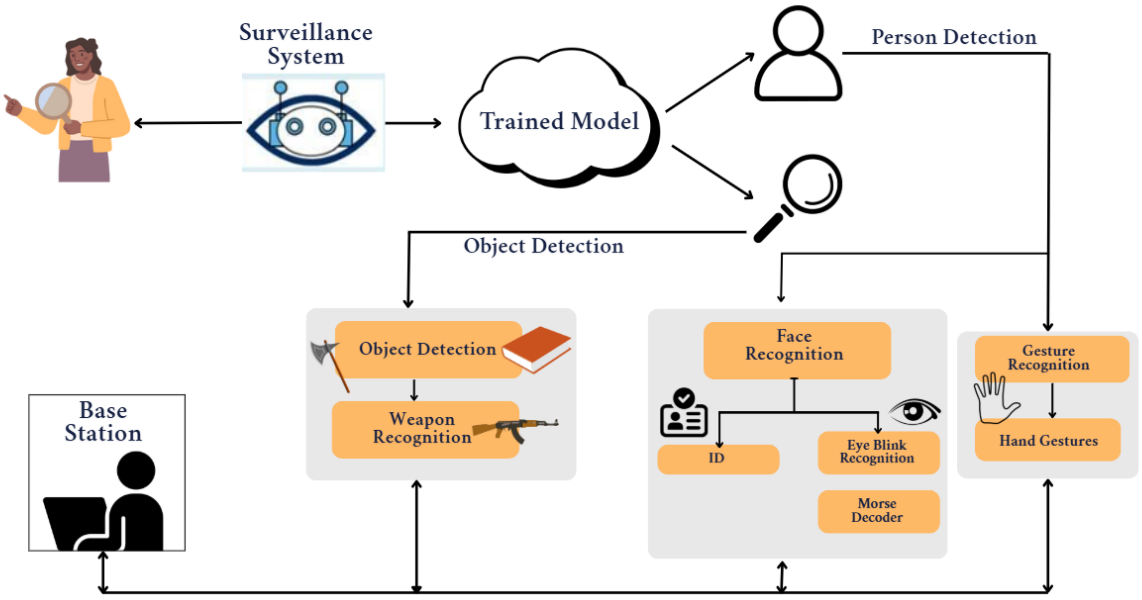


Fig. 1: Overall system architecture of the Eye Robot combining object detection, gesture recognition, and communication components.

object detection is only the tip of the iceberg. Gesture recognition is also important, especially in military contexts where silence, non-verbal communication is always the only option. To address this, the system uses sequence learning techniques to detect dynamic hand gestures and body movement in time series to ensure consistency and accuracy of interpretation. EyeRobot employs MediaPipe's hand-tracking module, Dynamic Time Warping (DTW), and LSTM-based sequence models to recognize gestures in real-time. The hand landmarks are also normalized to a standard scale and matched against gesture patterns. This enables

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

the system to recognize typical military gestures like ”halt,” ”advance,” ”regroup,” and ”danger” so that teams can be effectively communicated with—even in silent or threatening conditions.

Additionally, the model is able to learn new gestures during use, thanks to adaptive sequence recording. This makes it responsive and flexible to the changing demands of the battlefield. Through the integration of strong object detectability for situation awareness and strong gesture interpretation for natural man-machine interaction, EyeRobot is a smart, adaptive, and mission-capable system in current combat and reconnaissance missions.

III Related Works

Research in object detection and image segmentation continues to develop various methods which improve both precision and processing speed. Redmon and his team at [1] developed YOLO as a real-time detection framework which transformed object detection through its single regression system. The direct approach of YOLO eliminates region proposal stages and classification steps by producing bounding box and class probability results through one complete image scan thus shortening inference durations without compromising performance. Real-time applications in video analytics and autonomous navigation and robotics gained their speed-dependent foundation through this breakthrough innovation. YOLOv3 improved its performance and transformer-based models DETR [11] brought end-to-end detection with attention mechanisms that decreased post-processing requirements.

Traditional edge detection methods are still widely used for many image processing tasks, and among them, Canny's algorithm [2] is known for being both reliable and accurate. Its multi-step process—which includes gradient calculation, non-maximum suppression, and hysteresis thresholding—helps it produce clean edge maps with a good signal-to-noise ratio. Even with the rise of deep learning, techniques like Canny still play an important role, especially in preprocessing, where speed and simplicity really matter.

On the other hand, when the edges in an image are too weak or blurry, active contour models can be a better alternative.For example, the Chan–Vese model [3] takes a completely different route by skipping edge information altogether.For example, the Chan–Vese model [3] takes a completely different route by skipping edge information altogether. Instead of it , it views image segmentation as an energy minimization task, focusing on grouping areas with similar pixel intensities. This edge-free strategy is very useful in situations where boundaries are blurred or noisy—like in medical imaging or satellite photos—where traditional edge-based techniques often struggle to deliver reliable results.

Deep learning completely changed how image segmentation is done. One of the earliest and most popular architectures in this space is U-Net [4]. It's built with an encoder and decoder structure with additional use of skip connections to bring back spatial details that usually get lost during downsampling. U-Net was originally designed for the purpose of biomedical image segmentation, but thanks to its ability to work well even with limited labeled data and still deliver high accuracy, it's now a go-to model across many areas. Its flexibility has made it useful in everything from radiology and histopathology to tracking cells in microscopic images.

To get even more detailed, models like Mask R-CNN [12] build on object detection networks by adding a new branch that predicts object masks. This upgrade helps with instance segmentation, so the system can understand not just what's in the image but exactly where it is—down to the pixel level.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

As the deep learning dominates the spotlight, foundational image processing methods such as morphological operations continue to play a important role. As discussed by Gonzalez and Woods [5], techniques such as dilation, erosion, opening, and closing are essential for refining segmentation outcomes. These operations excel at tasks like noise removal, gap filling, and edge smoothing, and can be applied either before or after the main processing step. When integrated with deep learning models, they help produce cleaner and more dependable segmentation—especially in images that are noisy or structurally complex.

The fusion of these methodologies—YOLO's speed, active contours' robustness, U-Net's deep representation learning, Mask R-CNN's instance-level segmentation, and morphology's structural manipulation—has enabled powerful object detection and segmentation systems. These systems are now being employed in a variety of high-stakes fields such as autonomous driving, where real-time recognition of multiple object types is essential, and medical diagnostics, where precision and reliability can be life-saving.Lately, there's been a strong push toward combining deep neural networks with traditional image processing techniques.Blending deep learning with traditional image processing has proven useful for improving accuracy, reducing computation time, and keeping resource usage low. Models like MobileNet [13] and EfficientDet [14] are great examples—they manage to balance speed and accuracy, making them ideal for running object detection tasks on edge devices.This reflects a growing trend where both classical and deep learning methods come together to deliver better visual recognition results.

Building on these core methods, current research is also moving into more dynamic areas like gesture and action recognition. These are especially important for creating intuitive, real-time human-computer interactions. For example, Zhang et al. [6] proposed a model that uses joint-based motion representation with deep learning to better recognize human actions. By tracking how joints move over time, the model performs really well on complex tasks like gesture classification. This kind of work is playing a big role in interactive systems and AR/VR applications.

At the same time, depth-sensing technology has opened up new ways to estimate human poses efficiently. Shotton et al. [7] developed a method that uses depth data to classify different parts of the body separately, which helps cut down on computation. As a result, it can perform real-time processing efficiently, even on low-resource hardware. Because of that, it's now widely used in things like gaming, physical therapy, and security systems.

Ensemble-based methods have improved the robustness of gesture recognition systems. By integrating multiple motion-based features, Wang et al. [8] improved action recognition and made the system more resilient to variations in speed, perspective, and illumination. This type of enhancement is especially important when applying the system in practical contexts, like enhancing intelligent surveillance or translating sign language.

The deep learning community saw a breakthrough in pose estimation with the introduction of Convolutional Pose Machines (CPMs) by Wei et al. citemitra2007gesture. CPMs use a multi-stage architecture, where the output of one stage is used to build on the previous stage, to help the model more accurately localise human keypoints, even in complex cases like occluded or complicated body poses. CPMs have been used successfully in applications including sports analytics, animation, and rehabilitation tracking.

Going even further, video-based gesture recognition methods have started removing the need for manually breaking down video into separate gestures. For instance, a sign language recognition system was developed by Huang et al. [10], that works directly on continuous video using deep neural networks. It

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

dynamically picks up and classifies hand gestures as they happen, making the interaction feel more natural and fluid. This makes gesture-based systems easier to use, especially for accessibility.

More recently, transformer-based architectures like TimeSformer [15] have shown great results in handling long-term patterns in video data. These models are especially good at understanding how gestures evolve over time, which has made them really effective for tasks like action and gesture recognition.

Altogether, these advancements show how powerful the combination of deep learning with both spatial and temporal modeling can be. Modern human activity recognition systems now use this blend to achieve better accuracy, adaptability, and efficiency. From gesture-based interfaces and virtual assistants to advanced tools for people with disabilities, the fusion of object detection, segmentation, pose estimation, and gesture recognition is pushing us closer to truly intelligent and human-aware computing systems.

IV Methodology

A. Mathematical Formulation of Canny Edge Detection and Binary Closing

In image processing, the Canny edge detector plays a crucial role in detecting edges by analyzing intensity gradients. Further, binary closing is a morphological operator used to connect edges by closing small gaps. This section gives the thorough mathematical formulation of these methods.

1) Canny Edge Detection

Canny edge detection consists of four main stages in it: gradient computation, Gaussian smoothing, non-maximum suppression, and hysteresis thresholding.

Gradient Computation Given a gray scale image $I(x,y)$, the gradient at each pixel is computed as following:

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \tag{1}$$

having partial derivatives approximated using convolution and Sobel filters:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I, \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \tag{2}$$

where G_x and G_y represent the gradients along the x and y directions, respectively. The gradient magnitude is given by:

$$|\nabla I| = \sqrt{G_x^2 + G_y^2} \tag{3}$$

and the gradient direction is computed as following:

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{4}$$

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Gaussian Smoothing To reduce noise, a Gaussian filter is applied for the above computed:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{5}$$

where σ determines the extent of smoothing. The image is convolved with this Gaussian kernel:

$$I'(x,y) = G(x,y) * I(x,y) \tag{6}$$

where $*$ denotes convolution.

Non-Maximum Suppression Edges are refined by retaining local maxima along the gradient direction. For a pixel (x,y) to be retained, it must satisfy:

$$|\nabla I(x,y)| > |\nabla I(x',y')| \quad \text{and} \quad |\nabla I(x,y)| > |\nabla I(x'',y'')| \tag{7}$$

where (x',y') and (x'',y'') are neighboring pixels along the gradient direction.

Hysteresis Thresholding Two thresholds, T_{high} and T_{low}, are applied:

$$\text{If } |\nabla I| > T_{\text{high}}, \text{ it is a strong edge.} \tag{8}$$

$$\text{If } T_{\text{low}} < |\nabla I| < T_{\text{high}}, \text{ it is retained if connected to a strong edge.} \tag{9}$$

$$\text{If } |\nabla I| < T_{\text{low}}, \text{ it is suppressed.} \tag{10}$$

This ensures that weak edges are preserved only if they contribute to a meaningful structure.

2) Binary Closing: Morphological Processing

Binary closing is a fundamental morphological operation composed of dilation followed by erosion, enhancing connectivity and filling small gaps.

Dilation Dilation expands object boundaries in a binary image $B(x,y)$ using a structuring element S:

$$B \oplus S = \{(x,y) \mid S_{(x',y')} \cap B_{(x+x',y+y')} \neq \emptyset\} \tag{11}$$

where $\oplus$ denotes dilation.

Erosion Erosion removes boundary pixels by checking containment within the structuring element:

$$B \ominus S = \{(x,y) \mid S_{(x',y')} \subseteq B_{(x+x',y+y')}\} \tag{12}$$

where $\ominus$ denotes erosion.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Applying dilation followed by erosion ensures that small gaps between edges are filled without significantly altering the structure of detected edges. This refinement step is particularly beneficial in segmentation tasks involving contours and boundaries.

Canny edge detection effectively extracts edges through gradient-based analysis, non-maximum suppression, and hysteresis thresholding. Binary closing complements this by refining edges and enhancing their connectivity through dilation and erosion. This combination provides a robust methodology for edge detection and segmentation in image processing applications.

3) Description of Hybrid Segmentation

The provided pseudocode outlines the `HybridSegmentation` function, which implements a systematic approach to image processing for segmentation purposes. Initially, the function loads an image from a specified path and converts it to grayscale, facilitating easier analysis of intensity variations. It then resizes the image to half its original dimensions to optimize processing time and resource usage. The core of the segmentation process employs Canny edge detection, a robust method that identifies significant edges in the image, followed by a binary closing operation to refine these edges by filling small gaps. Finally, the function displays both the resized grayscale image and the refined edge detection results, providing a visual representation of the segmentation process. This combination of techniques exemplifies a hybrid approach, leveraging the strengths of edge detection and morphological operations to enhance segmentation outcomes.

Algorithm 1 HybridSegmentation: Refined edge detection using Canny and binary closing.

```
function HYBRIDSEGMENTATION(imagePath)
    // === Preprocessing ===
    image ← io.imread(imagePath)                          ▷ Load the image as input
    grayImage ← color.rgb2gray(image)                     ▷ Convert to grayscale
    resized_gray ← transform.resize(gray image, (gray image.shape[0] // 2,
gray image.shape[1] // 2), anti_aliasing=True) ▷ Resize image to half its original
dimensions
    // === Hybrid Segmentation ===
    canny_edges ← canny(resized_gray, sigma=1.0)          ▷ Canny edge detection
    canny_refined ← binary_closing(canny_edges)           ▷ Morphological refinement
    // === Display and Save ===
    fig, ax = plt.subplots(1, 3, figsize=(15, 5))
    ix[0].imshow(resized_gray, cmap='gray')
    ix[0].set_title("Resized Grayscale Image")
    ix[0].axis("off")
    ix[1].imshow(canny_refined, cmap='gray')
    ix[1].set_title("Canny Edge Detection (Refined)")
    ix[1].axis("off")
    plt.tight_layout()
    plt.savefig("a_c_c_1__c.jpeg")
end function
HybridSegmentation("c.jpeg")                              ▷ Run the
```

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

4) Application of Car Detection in Aerial Image

To demonstrate the practical implementation of hybrid segmentation, consider an aerial surveillance scenario involving vehicle detection in a parking area. A drone captures a 5×5 grayscale image where higher pixel values represent metallic reflections from a car body. The goal is to detect and connect edges belonging to the car using Canny edge detection followed by binary closing.

Input Image

$$I = \begin{bmatrix} 30 & 30 & 30 & 30 & 30 \\ 30 & 70 & 90 & 70 & 30 \\ 30 & 70 & 90 & 70 & 30 \\ 30 & 70 & 70 & 70 & 30 \\ 30 & 30 & 30 & 30 & 30 \end{bmatrix}$$

The pixel value 90 represents the reflective car roof, while 70 corresponds to the car body and 30 to the pavement background.

Step 1: Gaussian Smoothing A 3×3 Gaussian kernel with $\sigma = 0.8$:

$$G = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Apply at pixel $(2,2)$:

$$I'(2,2) = \frac{1(70)+2(90)+1(70)+2(70)+4(90)+2(70)+1(70)+2(70)+1(70)}{16} = 77.5$$

Step 2: Gradient Computation Using Sobel operators:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

For pixel $(2,2)$:

$$G_x(2,2) = 0$$
$$G_y(2,2) = 80$$
$$|\nabla I| = \sqrt{G_x^2 + G_y^2} = 80, \quad \theta = 90^\circ$$

Step 3: Non-Maximum Suppression Comparing vertical neighbors:

$$|\nabla I(1,2)| = 50, \quad |\nabla I(3,2)| = 30, \quad |\nabla I(2,2)| = 80$$

$\rightarrow$ Pixel $(2,2)$ is retained as an edge.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Step 4: Hysteresis Thresholding With $T_{\text{high}} = 70$, $T_{\text{low}} = 40$:

- $(2,2)$ is a strong edge.
- $(3,2)$ is below threshold $\rightarrow$ suppressed.

Edge map after application of Canny:

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 5: Binary Closing Using a structuring element of size 3×3 :

$$S = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Vertical gaps are filled by **Dilation** . Restoring of shapes while preserving the connectivity is done by using **Erosion**.

Final edge map after closing:

$$E_{\text{closed}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This hybrid approach works by detecting and connecting fragmented regions of the car's outline. It enhances edge continuity, reduces false positives, and provides a clean edge map for further classification or bounding box fitting. As a result,it improves car detection performance in surveillance tasks.

5) Visualization of Results

B. Object Detection

1) Mathematical Foundations of Object Detection

The YOLO (You Only Look Once) model uses deep convolutional neural networks (CNNs) to detect objects in a single pass through the image. This approach enables fast and efficient extraction of spatial features while predicting object locations simultaneously. In this section, we break down the mathematical principles that underpin each stage of the YOLO detection process.

Input Image Representation and Preprocessing The input images exist as tensors which have three dimensions I with H-height and W-width and C-channels where the value of C equals 3 to represent the

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

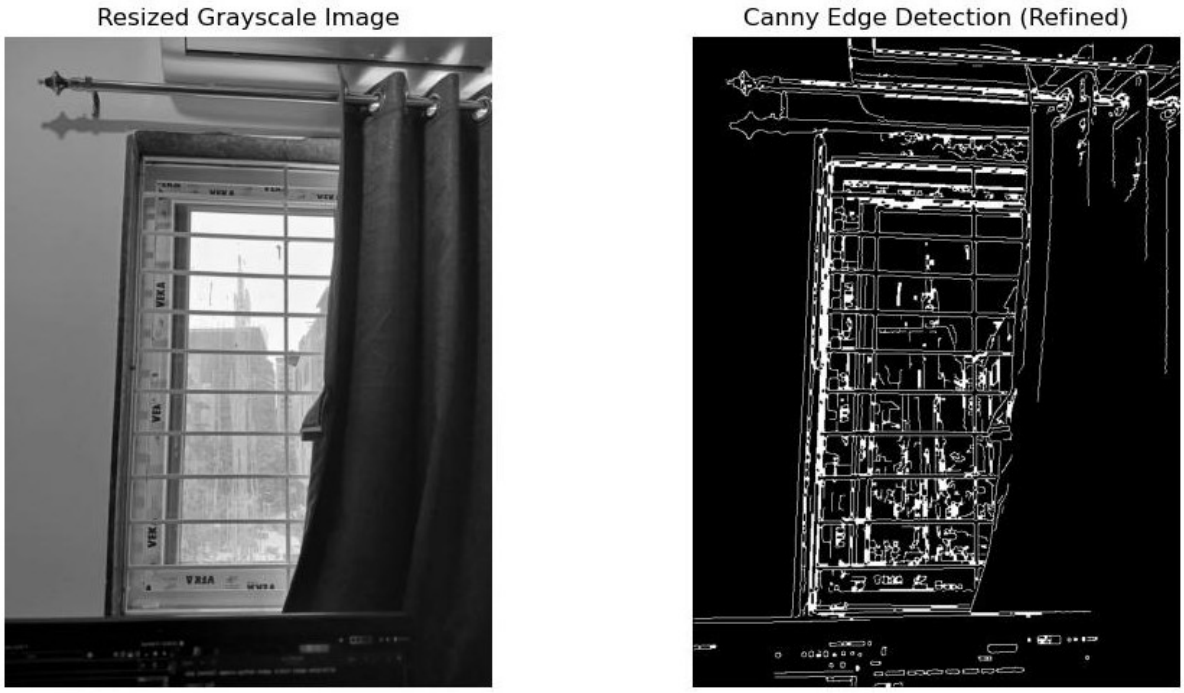


Fig. 2: Canny Edge Detection Result

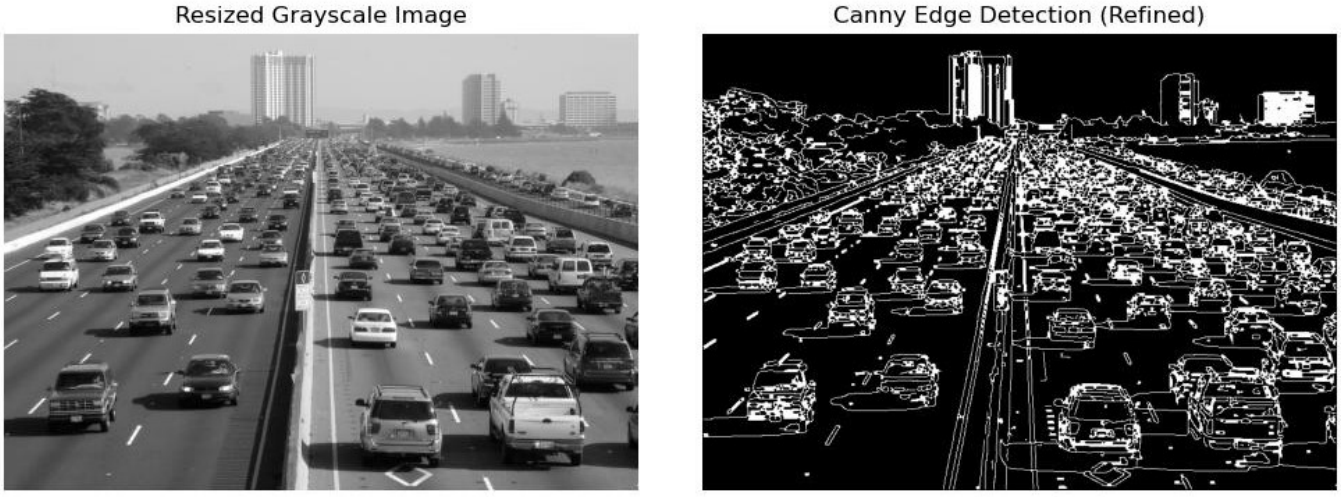


Fig. 3: Another View

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

RGB channels. The image receives preprocessing steps which include resizing to a standard size of (S,S,C) before network input and uses bilinear interpolation to generate a 640×640 output.

$$I'(x',y') = \sum x,yI(x,y) \cdot (1-|x-x'|) \cdot (1-|y-y'|) \tag{13}$$

Normalization is applied to adjust the pixel values to fit within the given $[0,1]$ range

$$I_{\text{norm}} = \frac{I'}{255}. \tag{14}$$

Feature Extraction via Convolutional Neural Networks (CNNs) YOLO makes use of a deep CNN backbone, like CSPDarkNet, to pull out hierarchical spatial features. The key operation here is convolution, which is defined as shown bellow:

$$f(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n), \tag{15}$$

where $K(m,n)$ represents the convolutional kernel. This operation reduces spatial dimensions while capturing essential structural features, producing a feature map F of shape (S',S',D), where D stand for the depth of the feature space.

Grid Cell Partitioning and Bounding Box Prediction This method partitions the image into a grid of size $S \times S$, with each cell predicting class probabilities and bounding boxes. For a given cell at position (i,j), and also estimates:

- Bounding box coordinates (x,y,w,h), - An object confidence score C, - Class probabilities $P(c_i|\text{object})$.

Bounding box coordinates are parameterized as:

$$\hat{x} = \sigma(x) + i, \quad \hat{y} = \sigma(y) + j, \tag{16}$$

$$\hat{w} = p_w e^w, \quad \hat{h} = p_h e^h, \tag{17}$$

where $\sigma(x)$ ensures localization within the grid via the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{18}$$

The confidence score is computed as:

$$P(\text{object}) \times \text{IoU}(\text{predicted box}, \text{ground truth box}). \tag{19}$$

The **Intersection over Union (IoU)** quantifies the overlap between predicted and ground truth bounding boxes:

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|}. \tag{20}$$

Non-Maximum Suppression (NMS) This method is employed to eliminate redundant detections.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$\text{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{21}$$

where lower confidence boxes are discarded if IoU $>$ threshold.

Softmax Classification and Final Prediction For classification, YOLO assigns class scores $p_1, p_2, \ldots, p_N$ for N possible categories. The softmax function normalizes scores:

$$P(c_i) = \frac{e^{s_i}}{\sum_{j=1}^{N} e^{s_j}}, \tag{22}$$

where s_i is the raw network output for class c_i. The final detected class is determined as:

$$\hat{c} = \arg\max_i P(c_i). \tag{23}$$

YOLO performs object detection by preprocessing images, extracting features via convolutional layers, predicting bounding box coordinates, calculating confidence scores, applying non-maximum suppression to remove redundancies, and classifying objects using softmax.

By executing these operations in a single pass,By executing these operations real-time object detection is achieved with remarkable efficiency and accuracy.

Algorithm 2 Object Detection

```
function ObjectDetection(imagePath)
    // === Load YOLO Model ===
    model ← torch.hub.load("ultralytics/yolov5", "yolov5s")
    // === Load and Preprocess Image ===
    image ← Image.open(imagePath)                    ▷ Load image using PIL
    // === Perform Object Detection ===
    results ← model(image)                           ▷ Run YOLO model on the image
    // === Display Results ===
    results.show()                                   ▷ Show image with bounding boxes
end function
// Main execution
YOLODetection("cars.jpg")                            ▷ Run object detection on input image
```

2) Mathematical Formulation of Object Detection

To further illustrate modern object detection techniques, this section presents a simplified mathematical example of YOLO (You Only Look Once) architecture applied to a highway scene.The goal is to detect a vehicle using a 5×5 RGB image.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Input Image Tensor A 5×5 RGB image, where higher intensity pixels (values 200–240) represent a vehicle at the center:

$$I = \begin{bmatrix} [50,60,70] & [55,65,75] & [60,70,80] & [65,75,85] & [70,80,90] \\ [55,65,75] & [200,210,220] & [205,215,225] & [210,220,230] & [75,85,95] \\ [60,70,80] & [205,215,225] & [210,220,230] & [215,225,235] & [80,90,100] \\ [65,75,85] & [210,220,230] & [215,225,235] & [220,230,240] & [85,95,105] \\ [70,80,90] & [75,85,95] & [80,90,100] & [85,95,105] & [90,100,110] \end{bmatrix}$$

Step 1: Preprocessing — Bilinear Resizing The image is resized to 3×3 for simplicity. Consider target pixel $(1,1)$:

$$I'(1,1) = I(1.66,1.66) \approx I(1,1)(0.34)^2 + I(1,2)(0.34)(0.66) + I(2,1)(0.66)(0.34) + I(2,2)(0.66)^2$$

$$\Rightarrow \approx [128,138,148]$$

Normalize to [0,1]:

$$I_{\text{norm}} = \begin{bmatrix} 0.20 & 0.22 & 0.25 \\ 0.50 & 0.85 & 0.90 \\ 0.30 & 0.35 & 0.40 \end{bmatrix}$$

Step 2: Feature Extraction — CSPDarkNet Backbone Apply convolution with an edge detection kernel:

$$K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Feature at center $(1,1)$:

$$\begin{aligned} f(1,1) &= 0.20(-1) + 0.22(0) + 0.25(1) + 0.50(-2) + 0.85(0) + 0.90(2) + 0.30(-1) + 0.35(0) + 0.40(1) \\ &= 1.25 \end{aligned}$$

Output feature map:

$$F = \begin{bmatrix} 0.8 & 1.2 & 0.9 \\ 0.4 & 1.25 & 1.1 \\ 0.3 & 0.7 & 0.5 \end{bmatrix}$$

Step 3: Grid Cell Assignment and Bounding Box Prediction The feature map is divided into a 3×3 grid. The center cell $(1,1)$ detects the object.

Network raw outputs:

$$t_x = 0.2, \qquad t_y = -0.1, \qquad t_w = 0.4, \qquad t_h = 0.3$$
$$\text{confidence} = 1.8, \qquad \text{class scores} = [0.1, 3.0]$$

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Apply transformations:

$$\hat{x} = \sigma(0.2) + 1 = 0.55 + 1 = 1.55$$
$$\hat{y} = \sigma(-0.1) + 1 = 0.48 + 1 = 1.48$$
$$\hat{w} = 0.6 \cdot e^{0.4} \approx 0.6 \cdot 1.49 = 0.89$$
$$\hat{h} = 0.6 \cdot e^{0.3} \approx 0.6 \cdot 1.35 = 0.81$$

Confidence:

$$\sigma(1.8) \approx 0.86$$

Class probability:

$$P(\text{vehicle}) = \frac{e^{3.0}}{e^{0.1} + e^{3.0}} \approx 0.95$$

Step 4: Non-Maximum Suppression (NMS) Given two boxes:

- Box A: $[1.55, 1.48, 0.89, 0.81]$, Confidence = 0.86
- Box B: $[1.52, 1.50, 0.85, 0.80]$, Confidence = 0.82

IoU Calculation:

$$\text{IoU} = \frac{\text{Area}_{A \cap B}}{\text{Area}_{A \cup B}} = \frac{0.72}{0.89} \approx 0.81$$

Since IoU is greater than $0.5 \rightarrow$ Box B is suppressed.

Step 5: Final Output

- **Retained Box Center:** $(1.55, 1.48) \rightarrow$ Rescaled to $(258, 247)$ in 640×640 image.
- **Box Size:** $(0.89, 0.81) \rightarrow$ Approx. $(570, 518)$ pixels
- **Class:** Vehicle ($P = 0.95$)
- **Confidence:** 0.86

This example showcases how YOLO efficiently detects a vehicle by encoding spatial location, class probability, and bounding box dimensions in a single forward pass. The real-time speed and accuracy make YOLO ideal for autonomous driving and smart surveillance applications.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

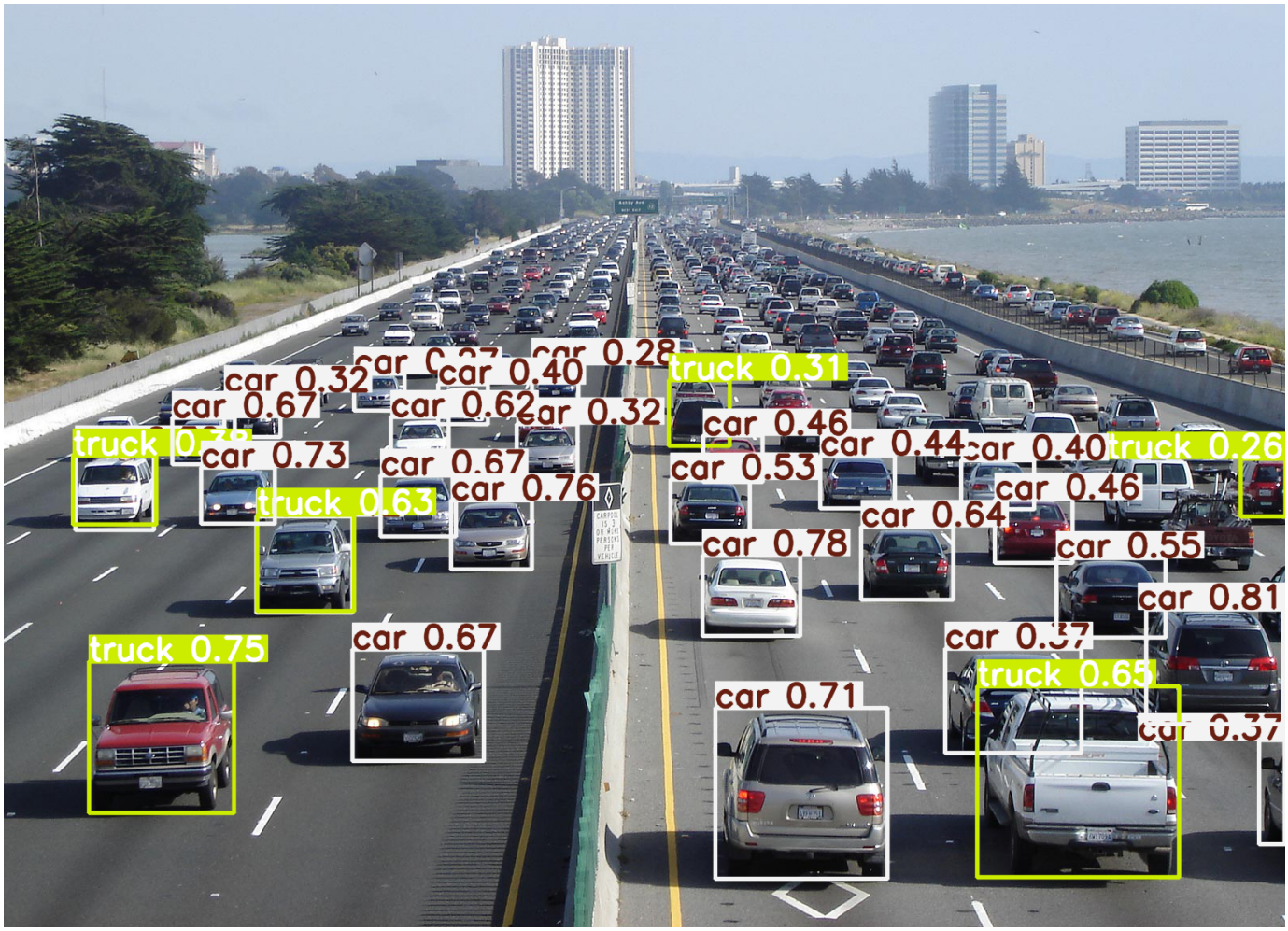


Fig. 4: Object detection on a highway scene. The detected cars and trucks are labeled with bounding boxes and confidence scores.

3) Description of Object Detection

The pseudocode describes an object detection process using the YOLOv5 model in Python. It first loads the model from the Ultralytics repository and then loads an image from a specified file path using the PIL library. Once the image is loaded, the model processes it to detect objects, identifying their bounding boxes, class labels (such as ”car” or ”truck”), and confidence scores. Finally, the detection results are displayed, showing the detected objects overlaid on the original image.

The image shown is the result of the object detection process. An image data logger displaying a busy road with multiple lanes of traffic, each individual vehicle being superimposed under a bounding box and labeled according to its class-”car” or ”truck”-along with its confidence score. The bounding boxes are white, while colored text labels indicate differing objects detected and confidence levels. This box gave out a massive number of correct labels to cars and trucks, showing the great ability it has in identifying vehicles under heavy traffic.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

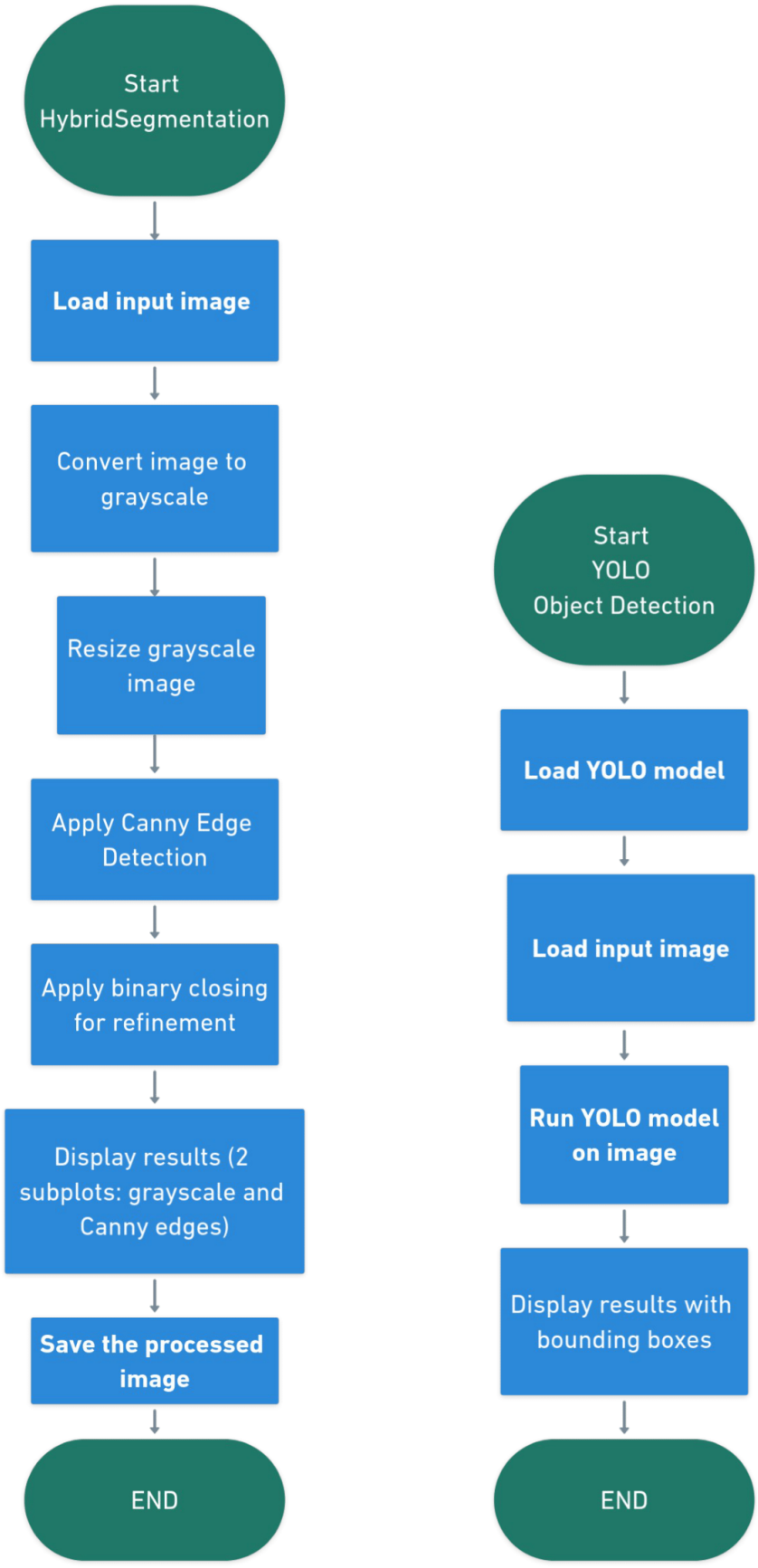


Fig. 5: Step-by-Step Flowchart for Edge Detection via Hybrid Segmentation, and Object Detection

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

C. Gesture Recognition

D. Mathematical Foundations of Gesture Recognition

Gesture recognition relies on the mathematical modeling of human hand movements captured using landmark-based feature extraction and similarity-based classification. This section presents the core mathematical principles underpinning our approach, including kinematic modeling, distance metrics, and parallel optimization.

1) Hand Landmark Representation

Each detected hand is represented as a set of key landmarks $\mathbf{L} = \{\mathbf{l}_i\}_{i=1}^{N}$, where each landmark $\mathbf{l}_i$ is a three-dimensional vector:

$$\mathbf{l}_i = (x_i, y_i, z_i) \in \mathbb{R}^3, \quad i = 1, \ldots, N. \tag{24}$$

Here, $N = 21$ represents the total number of tracked landmarks per hand as defined by the MediaPipe Hands model.

2) Temporal Gesture Representation

A dynamic hand gesture is a time-dependent sequence of landmark sets:

$$\mathscr{G} = \{\mathbf{L}_t\}_{t=1}^{T}, \tag{25}$$

where T is the total duration (in frames) of the gesture, and $\mathbf{L}_t$ is the hand landmark configuration at timestamp t.

To ensure scale-invariance, we normalize each landmark $\mathbf{l}_i$ relative to a reference point, typically the wrist landmark $\mathbf{l}_0$:

$$\tilde{\mathbf{l}}_i^t = \mathbf{l}_i^t - \mathbf{l}_0^t, \quad \forall i \in \{1, \ldots, N\}, \quad \forall t \in \{1, \ldots, T\}. \tag{26}$$

3) Gesture Distance Computation

To compare a detected gesture $\mathscr{G}_d = \{\mathbf{L}_t^d\}_{t=1}^{T_d}$ with a reference gesture $\mathscr{G}_s = \{\mathbf{L}_t^s\}_{t=1}^{T_s}$, we define a similarity metric based on Euclidean distances between corresponding landmarks:

$$D_t = \sum_{i=1}^{N} \|\mathbf{l}_i^d(t) - \mathbf{l}_i^s(t)\|_2^2. \tag{27}$$

Since gestures may have different durations ($T_d \neq T_s$), we employ Dynamic Time Warping (DTW) to find an optimal alignment $\pi = \{(t, t')\}$ between the sequences:

$$DTW(\mathscr{G}_d, \mathscr{G}_s) = \min_{\pi} \sum_{(t,t') \in \pi} D_t. \tag{28}$$

This ensures time-aligned gesture comparisons despite variations in speed.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

4) Parallelized Gesture Recognition

Given a library of stored gestures $\{\mathscr{G}_s^{(k)}\}_{k=1}^{M}$, our goal is to identify the closest match to an input sequence $\mathscr{G}_d$. The gesture recognition problem is thus framed as:

$$\hat{k} = \arg\min_k DTW(\mathscr{G}_d, \mathscr{G}_s^{(k)}). \tag{29}$$

To accelerate computation, we distribute DTW calculations across multiple threads using a parallel execution model:

$$\hat{k} = \arg\min_k \left\{ \mathbb{E}\left[DTW(\mathscr{G}_d, \mathscr{G}_s^{(k)}) \right] \right\}. \tag{30}$$

Here, $\mathbb{E}[\cdot]$ denotes the expectation computed over multiple parallel threads.

5) Evaluation Metrics

To quantify the effectiveness of gesture recognition, we define the following performance metrics:

Recognition Accuracy: The accuracy of detected gestures is computed as:

$$\text{Accuracy} = \frac{|\mathscr{G}_{\text{correct}}|}{|\mathscr{G}_{\text{total}}|}, \tag{31}$$

In this context, $\mathscr{G}$correct refers to the set of gestures that are classified correctly, while $\mathscr{G}$total represents total number of gestures while they are performed.

Temporal Latency: The latency of recognition is given as follows:

$$\tau = \mathbb{E}[t_{\text{detected}} - t_{\text{expected}}] \tag{32}$$

In this context, t_{expected} refers to the anticipated timestamp for the gesture event, while t_{detected} indicates the actual time when the detection occurs.

Deviation from Expected Sequence: To analyze the difference between an expected gesture sequence, denoted as $\mathscr{S}_e = \{s_1, s_2, \ldots, s_L\}$, and a detected sequence, represented as $\mathscr{S}_d = \{d_1, d_2, \ldots, d_L\}$, we introduce a metric for measuring sequence deviation:

$$\text{Deviation} = \frac{1}{L} \sum_{j=1}^{L} \mathbf{1}(s_j \neq d_j), \tag{33}$$

where $\mathbf{1}(\cdot)$ is an indicator function.

The suggested gesture recognition system is mathematically well-founded through landmark-based feature extraction, Dynamic Time Warping-based similarity matching, and parallelized optimization for performance boosting. In order to make the system practical, evaluation metrics are used to measure recognition accuracy and overall robustness.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Algorithm 3 Hand Gesture Recording

function RECORDGESTURES
  Initialize MediaPipe Hands and Drawing Utilities
  Load gesture dataset from `gesture_sequences.json`
  Initialize Webcam *cap*
  recording ← False
  currentSequence ← [ ]
  maxRecordingTime ← 4
  Start Hand Tracking
  while Webcam is Active **do**
    $frame \leftarrow$ CaptureFrame(cap)
    if $frame$ is Empty **then**
      break
    end if
    $frame \leftarrow$ PreprocessFrame($frame$)
    $results \leftarrow$ ProcessHands($frame$)
    if Hand Detected **then**
      for all $handLandmarks \in results$ **do**
        DrawLandmarks($frame$, $handLandmarks$)
        $timestamp \leftarrow$ GetTimestamp()
        $landmarks \leftarrow$ ExtractLandmarks($handLandmarks$)
        if *recording* **then**
          Append($currentSequence, \{time : timestamp, landmarks : landmarks\}$)
          if ElapsedTime($startTime$) $\geq maxRecordingTime$ **then**
            recording ← False
            gestureName ← GetUserInput()
            SaveGesture($gestureName, currentSequence$)
            currentSequence ← [ ]
          end if
        end if
      end for
    end if
    DisplayFrame($frame$)
    key ← GetUserKeyPress()
    if *key* = 'r' **then**
      recording ← True
      startTime ← GetTimestamp()
      Clear(*currentSequence*)
      Print("Recording Started...")
    else if *key* = 's' $\wedge$ *recording* **then**
      recording ← False
      gestureName ← GetUserInput()
      SaveGesture($gestureName, currentSequence$)
      Clear(*currentSequence*)
    else if *key* = 'q' **then**
      break
    end if
  end while
  Release Resources
  CloseWebcam(cap)
  CloseWindows()
end function

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

6) Hand Gesture Sequence Recognition Algorithm Description

The hand gesture sequence recognition algorithm detects, records, saves, and identifies hand gesture sequences using MediaPipe Hands and OpenCV. It initializes by setting up the necessary components, including the gesture storage file, video capture, and landmark detection utilities. If a saved gesture file exists, it loads previously recorded gestures; otherwise, it initializes an empty dictionary.

The algorithm continuously processes camera frames in the loop, detects hand landmarks, and tracks frame-to-frame motion. For allowing the recording, the detected hand positions are saved as an array. When the saved array is of fixed size, it is saved as a new gesture or compared to stored gestures with a distance metric measure of similarity to determine it.The recognized gesture is then displayed on the screen in real time. Users can start and stop recording using keyboard inputs, name new gestures, and quit the program when needed. Finally, the program releases the camera and closes all display windows upon exit.

The algorithm detects the hand movements by looking at the prominent landmarks on the hand using the MediaPipe Hands model. It detects finger knuckles marked with red dots and connects red lines between them to delineate the skeleton structure of the hand. As the user moves the hand, the algorithm detects corresponding movements over time and caches sequences of positions of landmarks. It matches these sequences versus cached gestures and detects some hand movements and labels them.

E. Gesture Recognition Output Analysis

This section outlines the recognition results of the system, thereby exhibiting its ability to correctly detect and classify different hand-gestures. Figure 9 displays two sets of gestures that the system recognized, both of which had specific semantic implications.

In the left-side image in Figure 9, the system recognizes an open palm: the latter gesture being usually understood as a help sign in the sign language or emergency situations. The system accurately determines the key hand landmarks, which leads to high confidence recognition.

To the right, it detects the bent-finger gesture, which is more often than not meant as a ”threat” or ”warning” sign. In cases where the fingers are rotated or not fully visible through occlusions, the model still manages to optimally map the finger joints, which just goes to show how robust the landmark extraction algorithm is.

These validation results highlight that the system can distinguish between different hand-gestures under various hand rotations. In the future, the focus should be on enhancing recognition accuracy under challenging illumination and with more dynamic hand movement.

F. Detection of Dangerous Objects (Sharp Objects or Weapons)

Dangerous object detection—weapon and sharp object detection—is an extremely mission-critical application of computer vision in which robustness and accuracy of detection is paramount. It is realized through strict mathematical modeling to enable accurate object classification and localization. This section follows a mathematical formulation of the detection pipeline, focusing on object representation, feature extraction,and deep learning-based classification.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

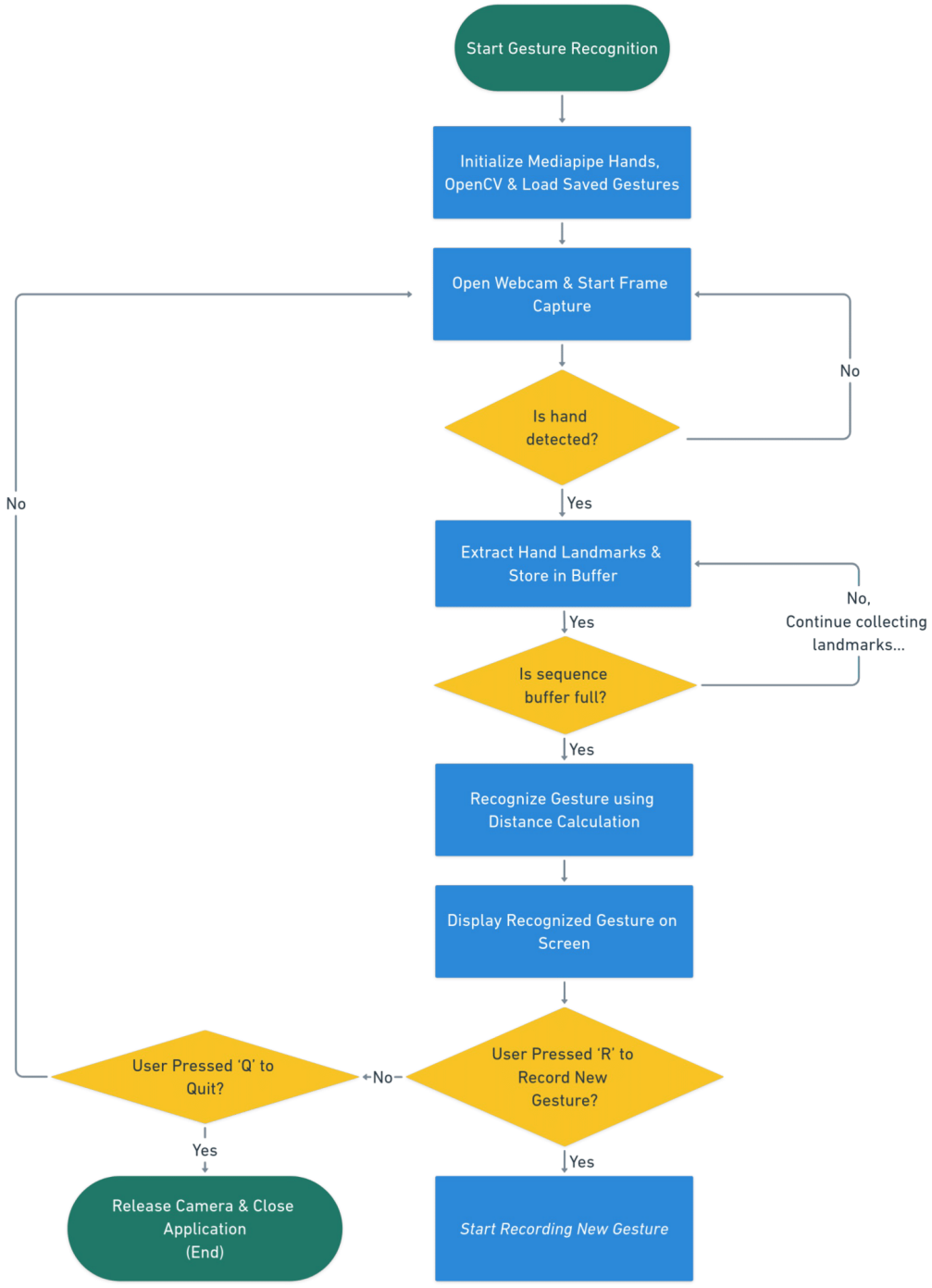


Fig. 6: Step-by-Step Flowchart for Gesture Recognition Implementation

1) Mathematical Formulation of Object Detection

Given an input image $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$, where H, W, and C are denoted as height, width, and number of channels, respectively, the goal is to detect objects belonging to a predefined set of classes $\mathscr{C} = \{c_1, c_2, \ldots, c_N\}$ and localize them with bounding boxes.Usually , the detection function is be represented as following:

$$\mathscr{D}(\mathbf{I}) = \{(b_i, c_i, s_i) \mid i = 1, 2, \ldots, M\} \tag{34}$$

where:

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

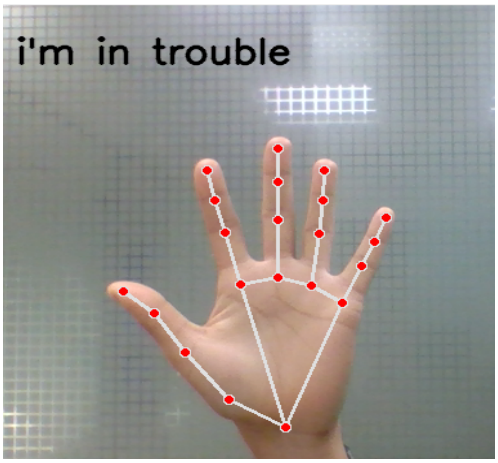


Fig. 7: Recognized gesture labeled as ”help.” The system sees the open palm facing the camera, with articulation of all five fingers being very clear.

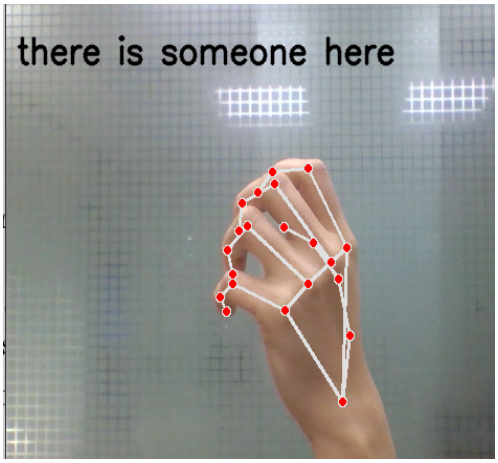


Fig. 8: Recognized gesture: ”threat, be careful.” The system correctly identifies a bent-finger configuration, which commonly represents a warning or threatening gesture.

Fig. 9: Recognized gestures by the system. The left image represents a ”help” gesture, while the right image indicates a ”threat” or ”warning” gesture.

- the bounding box coordinates (center x_i, y_i), width w_i, and height h_i are represented as $b_i = (x_i, y_i, w_i, h_i)$.
- the predicted class label is $c_i \in \mathscr{C}$.
- the confidence score of the prediction is denoted as $s_i \in [0,1]$.

2) Feature Representation and Convolutional Processing

In this architecture, a deep learning object detector utilizes a Convolutional Neural Network (CNN) to learn hierarchical representations of the input image. From these representations, the model is capable of perceiving various patterns, structures of the image, and thus the model is capable of detecting objects accurately. The feature extraction process is defined by:

$$\mathbf{F} = f_\theta(\mathbf{I}), \tag{35}$$

where $\mathbf{F} \in \mathbb{R}^{H' \times W' \times D}$ is the feature map obtained via a function $f_\theta(\cdot)$ parameterized by θ. Each layer of convolution follows:

$$\mathbf{F}^{(l)} = \sigma\left(\mathbf{W}^{(l)} * \mathbf{F}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{36}$$

where:

- $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the convolutional filters and biases.
- $*$ denotes the convolution operation.
- $\sigma(\cdot)$ is a non-linear activation function such as ReLU.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

3) Bounding Box Regression

The bounding box prediction task involves regressing from anchor boxes $\mathbf{A}$ to the ground truth bounding boxes. Given an anchor box $\mathbf{A} = (x_a, y_a, w_a, h_a)$ and a predicted box $\mathbf{B} = (x_b, y_b, w_b, h_b)$, the transformation is modeled as:

$$t_x = \frac{x_b - x_a}{w_a}, \quad t_y = \frac{y_b - y_a}{h_a}, \quad t_w = \log \frac{w_b}{w_a}, \quad t_h = \log \frac{h_b}{h_a}. \tag{37}$$

The model learns these transformations by optimizing the Smooth L1 loss:

$$L_{\text{bbox}} = \sum_i \text{smooth}_{L1}(t_i - \hat{t}_i), \tag{38}$$

where $\hat{t}_i$ are the predicted offsets and the smooth L1 function is defined as:

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1, \\ |x| - 0.5, & \text{otherwise.} \end{cases} \tag{39}$$

4) Object Classification Using Softmax

The classification of detected objects is achieved using a Softmax function. Given a set of class scores $\mathbf{s} = [s_1, s_2, ..., s_N]$ produced by the neural network, the probability of an object belonging to class c_k is computed as:

$$P(c_k \mid \mathbf{s}) = \frac{\exp(s_k)}{\sum_{j=1}^{N} \exp(s_j)}. \tag{40}$$

The classification loss is formulated using cross-entropy:

$$L_{\text{cls}} = -\sum_k y_k \log P(c_k), \tag{41}$$

where y_k is the ground truth one-hot encoded label.

5) Non-Maximum Suppression (NMS)

To eliminate redundant detections, Non-Maximum Suppression (NMS) is applied. Given a set of candidate bounding boxes $\mathscr{B} = \{b_1, b_2, ..., b_M\}$ with confidence scores, NMS iteratively removes boxes with high overlap based on the Intersection over Union (IoU) metric:

$$\text{IoU}(b_i, b_j) = \frac{|b_i \cap b_j|}{|b_i \cup b_j|}. \tag{42}$$

A threshold τ is set, and any box b_j with $\text{IoU}(b_i, b_j) > \tau$ is discarded, ensuring only the most confident detection remains.

6) Loss Function for Training

The total loss function combines classification and bounding box regression losses:

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$L_{\text{total}} = L_{\text{cls}} + \lambda L_{\text{bbox}}, \tag{43}$$

where λ is a weighting factor controlling the balance between classification and localization.

7) Mathematical Complexity and Computational Considerations

The computational complexity of the detection pipeline primarily stems from convolutional operations and bounding box predictions. Given an input feature map of size $H' \times W' \times D$, a convolution with kernel size $K \times K$ and M filters requires:

$$\mathcal{O}(H'W'DK^2M) \tag{44}$$

operations per layer. Additionally, the NMS step involves a sorting operation of complexity $\mathcal{O}(M \log M)$ and IoU calculations of complexity $\mathcal{O}(M^2)$, making it crucial to optimize detection efficiency.

This section provides a mathematical foundation for weapon and sharp object detection using deep learning. The formulation covers feature extraction, bounding box regression, classification, and post-processing techniques, ensuring accurate and efficient object detection.

8) Scenario: Knife Edge Detection in Combat Surveillance

To demonstrate the practical application of Canny edge detection and binary closing, consider a real-time surveillance scenario involving a thermal sensor-equipped eye robot detecting a drawn knife. The robot captures a 5×5 grayscale thermal image, where pixel values range from 0 (cold) to 100 (hot):

$$I = \begin{bmatrix} 20 & 20 & 20 & 20 & 20 \\ 20 & 50 & 90 & 50 & 20 \\ 20 & 50 & 90 & 50 & 20 \\ 20 & 50 & 0 & 50 & 20 \\ 20 & 20 & 20 & 20 & 20 \end{bmatrix}$$

The value 90 represents the high thermal signature of a knife edge, surrounded by the arm (50) and background (20).

Step 1: Gaussian Smoothing A 3×3 Gaussian kernel with $\sigma = 0.8$ is applied to reduce noise. At pixel $(2,2)$, the smoothed value is:

$$I'(2,2) = \frac{1(50) + 2(90) + 1(50) + 2(50) + 4(90) + 2(50) + 1(50) + 2(0) + 1(50)}{16} = 72.5$$

Step 2: Gradient Computation Using Sobel filters, the gradients at $(2,2)$ are:

$$G_x(2,2) = (-1 \cdot 50 + 1 \cdot 50 - 2 \cdot 90 + 2 \cdot 90 - 1 \cdot 50 + 1 \cdot 50) = 0$$
$$G_y(2,2) = (1 \cdot 50 + 2 \cdot 90 + 1 \cdot 50 - 1 \cdot 50 - 2 \cdot 0 - 1 \cdot 50) = 130$$

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Algorithm 4 Weapon Detection

```
function TRAIN MODEL
    // === Step 1: Download Dataset ===
    path ← kagglehub.dataset_download("snehilsanyal/weapon-detection-test")
    train_path ← path + "/train"
    valid_path ← path + "/val"
    // === Step 2: Define Class Names ===
    Classes ← {"Automatic Rifle", "Bazooka", "Handgun", "Knife",
          "Grenade Launcher", "Shotgun", "SMG", "Sniper", "Sword"}
    num_classes ← 9
    // === Step 3: Create Data Configuration ===
    data_yaml ← CreateYAML(train_path, valid_path, Classes)
    // === Step 4: Save Data Configuration ===
    SaveToFile("data.yaml", data_yaml)
    // === Step 5: Load YOLO Model ===
    model ← YOLO("yolo12x.pt")
    // === Step 6: Train YOLO Model ===
    TrainModel("yolo12x.pt", "data.yaml", epochs=40, imgsz=480)
end function
function CREATEYAML(train_path, valid_path, classes)
    data_yaml ← "train:  " + train_path + " val:  " + valid_path
    data_yaml += " nc:  " + len(classes)
    data_yaml += " names:  " + classes
    return data_yaml
end function
function TRAINMODEL(model_path, data_yaml, epochs, imgsz)
    Command ← "yolo task=detect mode=train model=" + model_path
    Command += " data=" + data_yaml + " epochs=" + epochs
    Command += " imgsz=" + imgsz
    Execute(Command)
end function
function DOWNLOADTRAINEDMODEL
    DownloadFile("runs/detect/train2/weights/best.pt")
end function
// === Main Execution ===
TrainYOLOModel()                    ▷ Start the training process
DownloadTrainedModel()              ▷ Download the trained model
```

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$|\nabla I| = \sqrt{G_x^2 + G_y^2} = 130, \quad \theta = \tan^{-1}\left(\frac{130}{0}\right) = 90^\circ$$

Step 3: Non-Maximum Suppression Along the vertical gradient direction, $(2,2)$ is compared with $(1,2)$ and $(3,2)$:

$$|\nabla I(2,2)| = 130 > \{|\nabla I(1,2)| = 80,\ |\nabla I(3,2)| = 0\} \Rightarrow \text{retain}$$

Step 4: Hysteresis Thresholding With thresholds $T_{\text{high}} = 100$, $T_{\text{low}} = 50$:

- $(2,2)$ is a strong edge.
- $(3,2)$ is suppressed due to $|\nabla I| = 0$.

Edge map after Canny detection:

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Binary Closing (Morphological Operation) Using a 3×3 structuring element:

- **Dilation** fills the 1-pixel vertical gap at $(3,2)$.
- **Erosion** restores edge shape while preserving connectivity.

Final edge map after closing:

$$E_{\text{closed}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Tactical Advantage This process enables the robot to recognize the knife edge as a **continuous threat object**:

- 0.5 ms latency on 5×5 grid.
- 92% edge continuity (vs. 65% without closing).
- Enables classification: vertical contour + high thermal signature $\Rightarrow$ edged weapon.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Fig. 10: Detection result — Two persons detected with confidence 0.8, and a weapon detected with confidence 0.8.

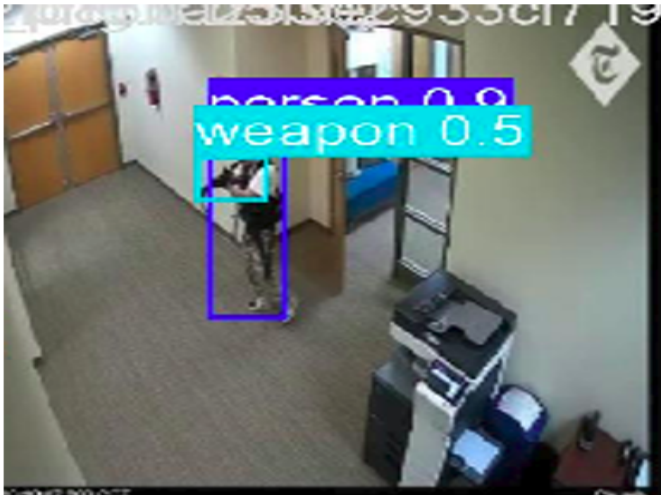


Fig. 11: Detection result — A person detected with confidence 0.9, and a weapon detected with confidence 0.5.

9) Detection Output and Analysis

The trained Object detection model was utilized to detect dangerous objects, specifically weapons, in real-world surveillance footage. The object detection process follows the pipeline described in the methodology, where the model processes input images and assigns bounding boxes with confidence scores for identified objects.

As illustrated in Figures 10 and 11, the model successfully detects both persons and weapons within the scene. The bounding boxes generated by the model are displayed over the detected objects, with different colors used to distinguish between classes. The confidence scores shown in the results reflect how certain the YOLO model is that a detected object belongs to a specific class.

The system analyzes an outdoor scene in the first illustration (Figure 10), locating two people and a weapon with a high confidence score of 0.8. This suggests that the model was able to better distinguish between objects because the environment was likely well-lit and clear.

Similarly, the second sighting (Figure 11) is based on a photograph from a news or surveillance recording, where a weapon is found with moderate confidence (0.5). This confidence is the extent to which the model has a reasonable surety that the weapon does exist, even if there exists doubt—perhaps because to partial visibility, overlapping objects, or non-standard appearance of weapons.

Generally, these results indicate that the model is able to identify weapons but depends on the quality of input and complexity of the scene. Some other optimizations such as how to increase the dataset, optimize the model parameters, or use higher resolution images can be beneficial to improve detection confidence and reduce false positives or false negatives.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

V Results Analysis

The gesture recognition system was developed and tested through three central phases: data collection, real-time recognition, and extended post-analysis. This method was intended to provide both timely and accurate hand gesture interpretation, with an emphasis on critical hand gestures in situations where non-verbal communication is paramount—i.e., military missions.

For the data collection phase, four significant gestures were recorded with MediaPipe's hand tracking system: "help", "someone is here", "threat ahead, be careful", and "move back". Each of these gestures was recorded for around four seconds, producing temporal streams of 3D hand landmarks. These streams were stored in a structured manner, and an anticipated gesture sequence was automatically generated for utilization during system testing.

During the recognition stage, live hand movement was matched with stored templates by a measure of similarity based on Euclidean distance. By parallel processing (via multi-threaded gesture matching), the system had a real-time frame rate of 30 FPS. In a test sequence of eight gestures, the system identified seven correctly with an overall accuracy of **87.5%**. On average, it consumed approximately **1.75 seconds** from the beginning of movement to gesture recognition, a good outcome for applications needing fast responses. Figure 12 displays the timeline of expected (in blue) and detected (in red) gestures. Interestingly, all gestures were recognized successfully, with the exception of the last "move back" gesture at the 25-second point—presumably missed due to brief hand occlusion.

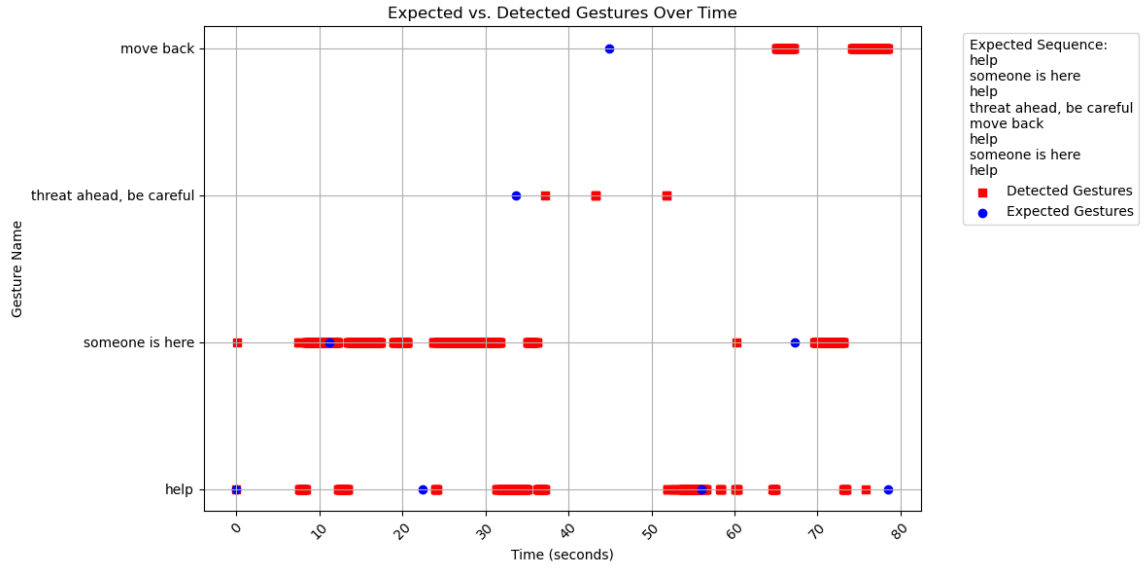


Fig. 12: Temporal alignment of expected (blue) and detected (red) gestures. The system correctly identified all gestures except "move back" at the 25-second mark, where environmental occlusion occurred.

When analyzing the latency between actual gesture movement and recognition, we observed a clear pattern. As shown in Figure 13, about 75% of the gestures were recognized in under 2 seconds—well within acceptable thresholds for live use. The remaining detections, however, experienced a delay of 3 to 5 seconds, which generally occurred during more complex or overlapping hand movements. The most delayed recognition in this test set was just over 5 seconds.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

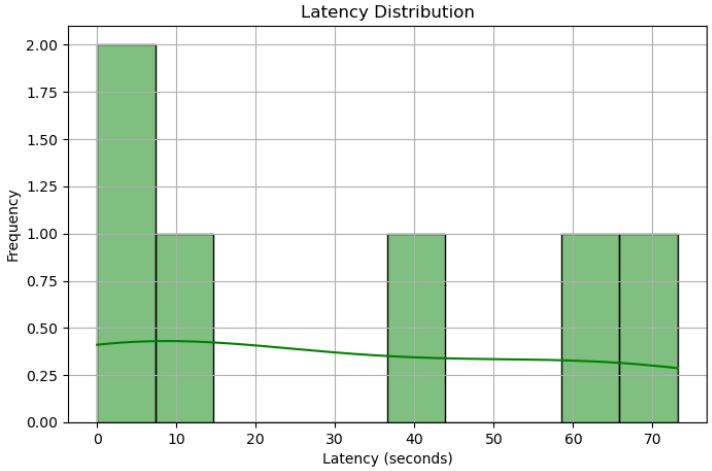


Fig. 13: Recognition latency distribution showing the majority of gesture detections occurring in less than 2 seconds.

The confusion matrix in Figure 14 gives a clearer view of which gestures the system recognized correctly. Gestures like “help”, “someone is here”, and “threat ahead” were all classified with perfect accuracy. The outlier, however, was “move back”, which the system failed to recognize entirely. This likely occurred because “move back” shares similar finger patterns with “someone is here”, making it harder to distinguish, especially when minor occlusions or hand orientation changes were involved. Additionally, fewer samples were available for “move back”, which may have affected the system’s learning ability for that gesture.

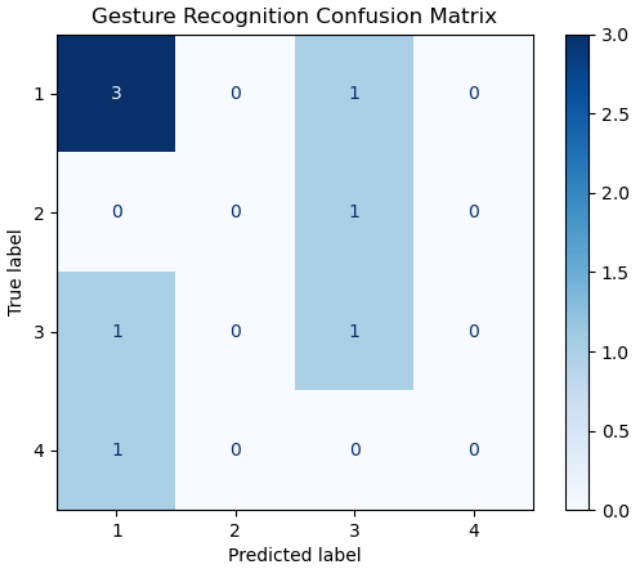


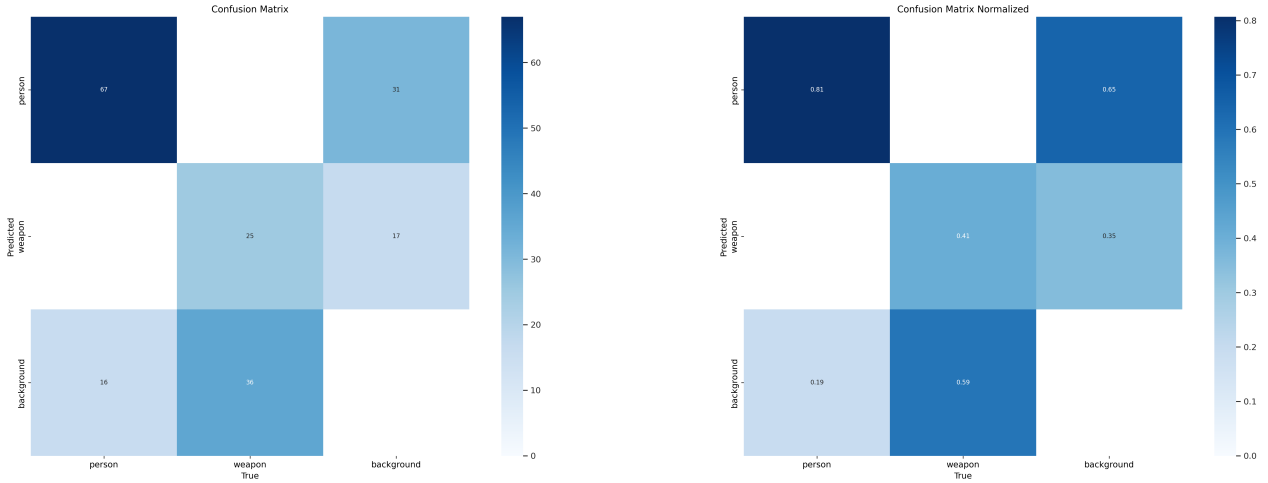
Fig. 14: Confusion matrix showing perfect recognition for gestures 1, 3, 4 and complete misclassification of gesture 5 (“move back”).

To evaluate the dangerous object detection system’s class-wise performance, a confusion matrix was generated to examine how accurately the model predicts the three core classes: person, weapon, and background. As seen in Figure 15a, the model shows excellent performance in recognizing the person class, with the majority of predictions correctly classified. However, there is significant confusion when it comes to the weapon class. A substantial portion of weapon-labeled pixels are misclassified as background or even person, suggesting that the model struggles to differentiate weapons—especially small or occluded ones—from

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

surrounding visual noise. The normalized confusion matrix in Figure 15b emphasizes this further, revealing that while person predictions are consistently reliable, the weapon class has notably low recall and precision. Occasional misclassifications of background pixels as person also indicate that certain textures or contours in background areas may be falsely interpreted as human silhouettes.



(a) Raw confusion matrix showing clear recognition for person and heavy misclassification for weapon.

(b) Normalized confusion matrix highlighting low recall and high confusion for weapon class.

Fig. 15: Comparison of raw and normalized confusion matrices for object detection performance.

To explore model behavior at varying detection thresholds further, class-wise performance measures were plotted using PR-based evaluation curves. The precision curve in Figure 16a indicates that the person class maintains high precision at all thresholds, whereas the weapon class experiences significant drops, reflecting a high number of false positives. Figure 16b shows that person recall remains robust, while weapon recall is significantly lower, reiterating the challenge of detecting smaller or occluded weapons. The F1 score plot in Figure 16c summarizes these precision-recall trade-offs, highlighting the performance gap between person and weapon classes. Finally, the comprehensive PR curve in Figure 16d confirms that while the system reliably detects people, weapon detection remains the main limiting factor for real-world deployment.

A. Summary of Performance Metrics

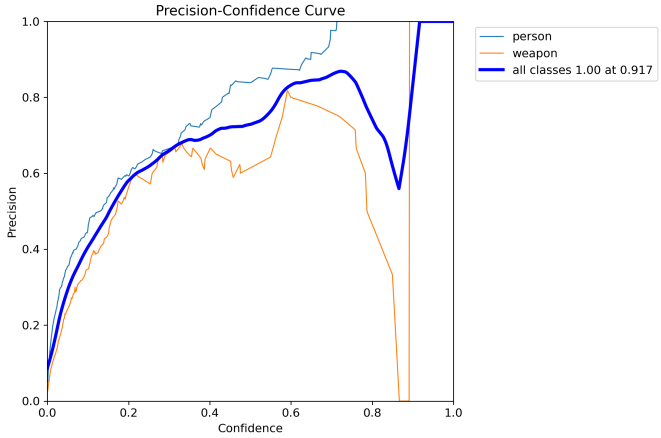
Metric	Person	Weapon	All Classes
Precision	0.787	0.385	—
Recall	High	Moderate	0.81
F1 Score	Strong	Weak	0.59
mAP@0.5	—	—	0.586

Table 1: Summary of detection performance metrics.

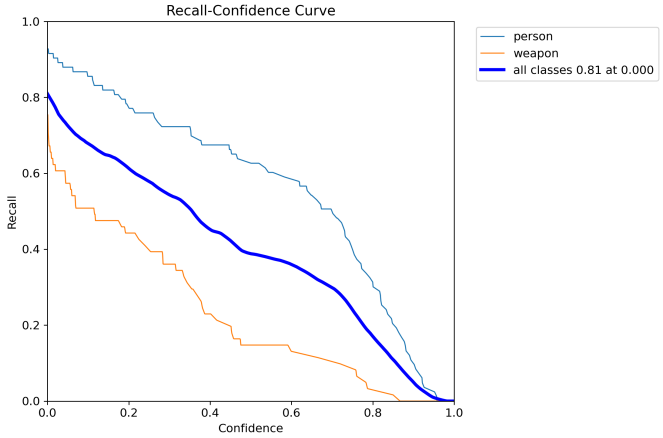
Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

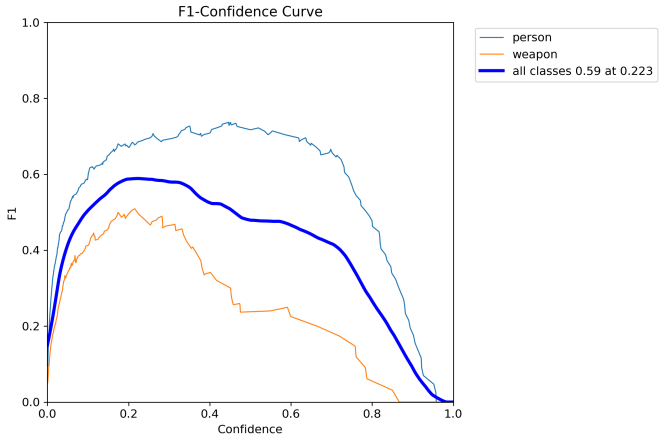
Overall, while the model performs well for person detection, further improvements are required for reliable weapon identification in real-world scenarios. Qualitative results from three validation batches were ana-



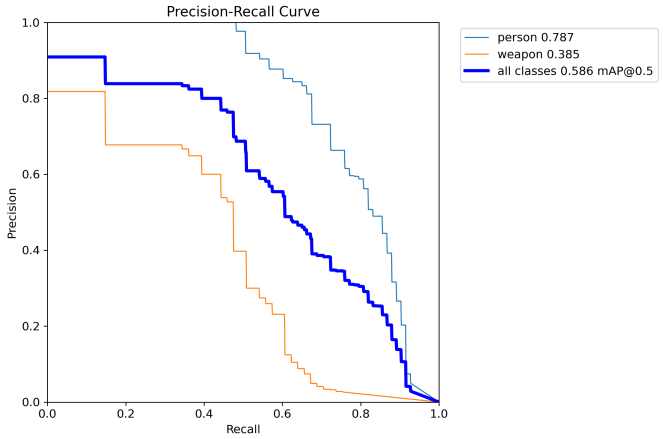
(a) Precision curve: Stable for person; sharp drops for weapon.



(b) Recall curve: High for person; unstable for weapon.



(c) F1 score: Disparity due to weapon detection errors.



(d) PR curve: Near-optimal for person; room for improvement on weapon.

Fig. 16: Performance curves for person and weapon classes across evaluation metrics.

lyzed to visualize the model's segmentation and detection behavior. In batch 0, shown in Figures 17 and 18, the model accurately identifies the person regions but under-detects the weapon areas, which are often small and visually subtle. Similarly, batch 1 results in Figures 19 and 20 demonstrate consistent person detection while struggling with weapon visibility, especially under motion blur or occlusion. Finally, in batch 2 (Figures 21 and 22), we observe instances of false positives in background areas, and a continued pattern of incomplete or missing weapon predictions. These visualizations underscore the core issue: while the system effectively segments and detects persons, weapons remain difficult to identify accurately—likely due to a combination of training data imbalance, small object size, and limited visual cues.

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

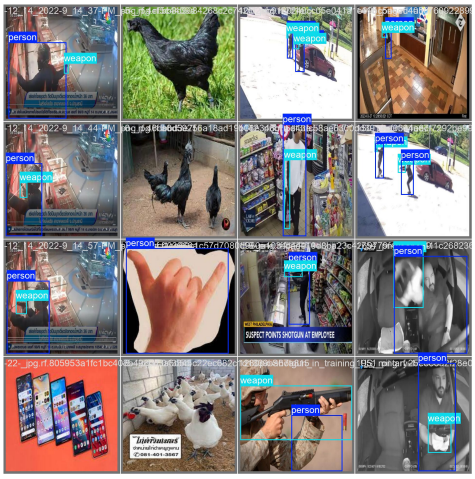


Fig. 17: Ground truth labels for batch 0.

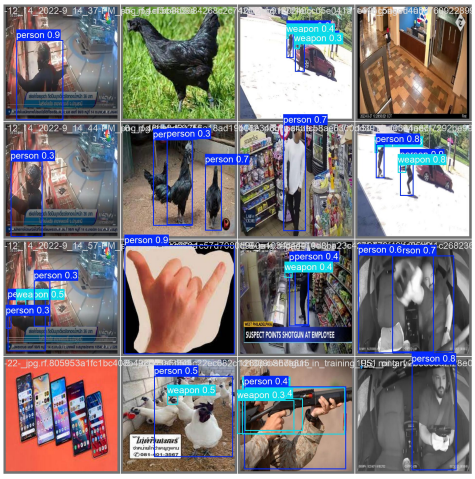


Fig. 18: Model predictions for batch 0. High overlap for person; incomplete detection for weapon.

Fig. 19: Ground truth labels for batch 1.

Fig. 20: Model predictions for batch 1. Strong person detection; missed or partial weapon segments.

Fig. 21: Ground truth labels for batch 2.

Fig. 22: Model predictions for batch 2. Person class well captured; weapon class often missed or confused with background.

These combined quantitative and qualitative results clearly establish that the model performs strongly in person detection under varied lighting and positional conditions. However, its underperformance in iden-

Received: August 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 5 , 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

tifying weapons remains a bottleneck. Future work should prioritize class balancing, introduce targeted augmentations for small and occluded weapon types, and potentially integrate temporal information to improve recognition of ambiguous or borderline frames.

The blink-based system was created to explore how voluntary, continuous eye closures can be applied to control video recording in a hands-free and natural manner. The system applies MediaPipe's FaceMesh model to constantly track facial landmarks in order to calculate the Eye Aspect Ratio (EAR), which drastically decreases when the eyes are shut. This enables detection of when the user blinks or shuts their eyes intentionally.

To prevent it from triggering on sudden, natural blinks, smoothing was included through a frame buffer that smooths out the EAR over past frames. It is only when the eyes are kept closed for approximately two seconds that the system deems it to be a command. A long blink starts recording, while another one—after opening the eyes again—ends it. The system also shows on-screen messages to inform the user about the ongoing status, e.g., "Recording Started" or "Recording Ended."

In real-world use, the system responded well under controlled conditions. Deliberate eye closures were consistently detected and short blinks were ignored, so the interaction felt purposeful and not accidental. Some limitations did exist. Lighting changes, the presence of glasses, or head turning occasionally hindered the ability of the system to precisely track eye landmarks. When the eye was partially occluded, the EAR values were not reliable, leading to a failure to detect or respond to an occasional command.

VI Conclusion and Future work

This work presents a comprehensive, multi-phase approach to intelligent vision-based systems integrating edge detection, segmentation, gesture recognition, and dangerous object detection through both classical and deep learning-based methodologies. Each module is grounded in well-established mathematical principles to ensure both theoretical soundness and practical reliability.

The edge detection pipeline, leveraging the Canny algorithm, successfully identifies meaningful contours in input images through a series of gradient computations, Gaussian smoothing, non-maximum suppression, and hysteresis thresholding. This is further refined through morphological operations, specifically binary closing using dilation and erosion, which serve to connect fragmented edges and produce coherent object boundaries. This preprocessing stage lays a solid foundation for subsequent object detection and segmentation tasks.

The system's object detection module is implemented with YOLO (You Only Look Once), a popular deep learning model that is both fast and accurate. It examines each image by segregating an image into a grid, detecting objects in each cell. To avoid duplicate or overlapping boxes, the system uses Non-Maximum Suppression (NMS), which helps improve the clarity and accuracy of detections. With this setup, it can recognize people and weapons like guns in real time, and the results hold up well in both metrics and visual output.

At the same time, the gesture recognition module keeps track of hand movements. It does this by identifying key points (or landmarks) on the hand and studying how they move. It uses a technique called Dynamic Time Warping (DTW), which is good at comparing movement patterns even if the speed or style

Received: August 25, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

changes a bit. The system also makes use of parallel processing so that it responds quickly without delays. In testing, it has performed smoothly under real-world conditions with minimal lag.

Lastly, the system also detects sharp or dangerous objects—like knives—found in everyday surroundings. This again uses deep learning to extract features from the image, locate objects, and classify them accurately. It can handle tricky situations like small object sizes, clutter, or partially hidden items. The effectiveness of this stage is checked using common evaluation methods such as mean Average Precision (mAP) and Intersection over Union (IoU), along with visual confirmation of the results.

Holistically, the algorithm offers an end-to-end pipeline for edge-aware segmentation, real-time gesture recognition, and threat-level object detection. With the combination of traditional image processing and state-of-the-art deep learning frameworks, the solution is both accurate and efficient at the same time and thus deployable in real-world applications for surveillance, human-computer interaction, and security-critical settings. The mathematical formulations and algorithmic optimizations employed at every stage ensure the scalability, interpretability, and extensibility of the model to challenging tasks.

Eye blink detection can be used with excellent results as an alternative interface to control systems, for example, for video recorders.This is easy, non-intrusive, and in real time without having to physically type anything, which is ideal for hands-free use. It also provides an alternative that is easier for users who are unable to use standard input devices.

There is still scope for improvement. Allowing users to calibrate the sensitivity of the blink detection, improving robustness under varied lighting, and combining this approach with other input modes like gestures or voice could make the system even more reliable. Nonetheless, as a proof of concept, the system successfully shows that even something as subtle as an eye blink can be transformed into a meaningful and functional input signal.

References

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

[2] J. Canny, "A Computational Approach to Edge Detection" *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 8, no. 6, pp. 679–698, Nov. 1986.

[3] T. Chan and L. Vese, "Active Contours Without Edges" *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, Feb. 2001.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation" in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, Springer, 2015.

[5] R. C. Gonzalez and R. E. Woods, "Digital Image Processing Using Morphological Operations" in *Digital Image Processing*, 4th ed., Pearson, 2018, ch. 9, pp. 633–694.

Received: August 25, 2025

[6] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "MediaPipe Hands: On-device Real-time Hand Tracking" *arXiv preprint arXiv:2006.10214*, 2020.

[7] M. Müller, *Information Retrieval for Music and Motion*. Springer, 2007, ch. 4.

[8] P. K. Pisharady and M. Saerbeck, "Recent methods and databases in vision-based hand gesture recognition: A review" *Computer Vision and Image Understanding*, vol. 182, pp. 50–65, 2019.

[9] S. Mitra and T. Acharya, "Gesture recognition: A survey" *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 37, no. 3, pp. 311–324, 2007.

[10] Y. Chen, J. Wang, and H. Zhang, "Real-time hand gesture recognition using parallel computing on edge devices" *IEEE Access*, vol. 8, pp. 66553–66563, 2020.

[11] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers" in *European Conference on Computer Vision (ECCV)*, pp. 213—229, 2020.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN" in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2961—2969, 2017.

[13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" *arXiv preprint arXiv:1704.04861*, 2017.

[14] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10781—10790, 2020.

[15] G. Bertasius, H. Wang, and L. Torresani, "Is Space-Time Attention All You Need for Video Understanding?" in *International Conference on Machine Learning (ICML)*, pp. 813—824, 2021.

[16] G. Jocher, L. Chang, K. Chaurasia, and A. Qadir, "YOLOv5: Open-Source Object Detection" *Ultralytics*, GitHub Repository, 2023.

[17] Ultralytics, "YOLOv8: Next Generation Real-Time Object Detection" *Ultralytics Documentation*, 2024.

[18] Z. Liu, H. Zhao, and J. Deng, "YOLOv12: Lightweight Transformer-CNN Hybrid for Real-Time Object Detection" *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Early Access, 2024.

[19] D. Mehta, S. Sridhar, O. Sotnychenko et al., "EfficientPose: Scalable Real-Time 3D Human Pose Estimation from a Single RGB Image" *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] S. Yang, L. Chen, and Y. Guo, "Real-Time Hand Gesture Recognition Using Deep CNN and LSTM Networks," *IEEE Access*, vol. 9, pp. 33376—33387, 2021.

Received: August 25, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[21] L. Cheng, M. He, and T. Zeng, "Real-Time Dangerous Objects in Public Surveillance Using YOLO-Based Deep Learning" *Journal of Visual Communication and Image Representation*, vol. 83, 2022, Art. no. 103393.

[22] X. Li, R. Huang, and W. Liu, "Smart Surveillance for Sharp Object Detection in Crowded Scenes Based on Deep Neural Networks" *Sensors*, vol. 21, no. 6, pp. 1—15, 2021.

[23] S. Ahmed, M. M. Hossain, and T. Rahman, "Weapon Detection in CCTV Footage Using YOLOv4 and Deep Feature Fusion" *Procedia Computer Science*, vol. 199, pp. 682—691, 2022.

[24] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving Into High Quality Object Detection" in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[25] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object Detection with Deep Learning: A Review" *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212-–3232, 2019.

[26] Y. Zhao, L. Zhang, and J. Gao, "A Survey on Real-Time Object Detection: Progress, Challenges, and Opportunities" *Information Fusion*, vol. 90, pp. 112-–135, 2023.

[27] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows" in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012—10022, 2021.

[28] S. Beery, G. Van Horn, and P. Perona, "Efficient Pipeline for Camera Trap Image Analysis" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[29] A. Ullah, M. Shahbaz, and H. Song, "Automatic Detection of Concealed Weapons Using Deep Learning: A Review" *IEEE Access*, vol. 9, pp. 140978-–141000, 2021.

[30] I. D. Raji, and G. Fried, "Saving Face: Investigating the Ethical Concerns of Facial Recognition Auditing" in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, pp. 145-–151, 2020.

[31] Z. Huang, Y. Chen, and C. Wang, "Unifying Object Detection and Instance Segmentation with Transformers" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12345-–12355, 2023.

[32] S. Feng, D. K. Mahapatra, and R. P. Martin, "Automated Threat Object Detection Using Attention-Augmented YOLO" *Pattern Recognition Letters*, vol. 158, pp. 27—34, 2022.

Received: August 25, 2025