Original Article

COMPUTER VISION TECHNIQUES FOR TRAFFIC SIGN DETECTION IN ADAS APPLICATIONS

Taras Volodymyrovych Kravchenko
Department of Software Tools, National University, Ukraine
DOI:https://doi.org/10.5281/zenodo.16409408

ABSTRACT this paper presents a comprehensive traffic sign recognition system designed to enhance advanced driver assistance systems (ADAS) and autonomous vehicles. The system employs a three-step algorithm comprising color segmentation, shape recognition, and a neural network-based classification to detect and identify various traffic signs in real time. Leveraging the advantages of color-based segmentation for rapid processing and combining it with sophisticated shape detection methods, our approach ensures high accuracy and precision even under challenging conditions such as varying illumination and occlusions. The integration of neural networks allows for effective classification across a broad range of sign types, addressing limitations seen in traditional methods. Our system's ability to operate with standard onboard cameras, combined with its resilience to lighting variations, marks a significant advancement in traffic sign recognition technology. Extensive testing demonstrates its efficacy in real-world scenarios, highlighting its potential to enhance road safety and support autonomous driving technologies.
KEYWORDS artificial intelligence; computer vision; machine learning; neural networks; pattern recognition.

I. INTRODUCTION differentiation process, enabling rapid and reliable traffic sign recognition is a critical component of modern segmentation even under diverse lighting conditions (Fig. 2). Advanced driver assistance systems (ADAS) and autonomous vehicles. Accurate and timely recognition of traffic signs not only ensures compliance with road regulations but also significantly enhances road safety. Traditional methods of traffic sign detection and recognition often face challenges such as varying illumination, occlusions, and the need for real-time processing. To address these issues, this paper presents a novel traffic sign recognition system that leverages advanced AI techniques including color segmentation, shape recognition, scaling of the area of interest, and neural network-based classification. The proposed system begins with color segmentation, a crucial step that isolates the traffic signs from their Surroundings. Utilizing the HSL (Hue, Saturation, Lightness) or HSV (Hue, Saturation, Value) color models, the system efficiently classifies the sign border frames based on their colors—grey, red, yellow, and blue—by focusing primarily on the hue component. This approach simplifies the color figure 1. Examples of common traffic signs used in traffic sign recognition systems.

Original Article

Figure 2. Traffic sign examples illustrating segmentation under varying lighting conditions.

Following color segmentation, the system employs robust shape recognition algorithms to identify the geometric shapes of the signs. This step is essential as traffic signs have standardized shapes, such as circles, triangles, rectangles, and octagons, which correspond to specific types of information and regulatory instructions. To ensure high accuracy in detection, the system scales up the area of interest around the detected shapes. This scaling process enhances the resolution of the potential sign regions, allowing for more precise analysis and reducing the risk of false positives. Finally, the system incorporates a neural network-based classifier that leverages deep learning techniques to distinguish between various types of traffic signs. This classifier is trained on a comprehensive dataset of traffic sign images, enabling it to learn and generalize across a wide range of sign types and conditions. The integration of neural networks ensures robust performance, addressing the limitations of traditional rule-based and template-matching methods. This paper details the development and evaluation of our traffic sign recognition system. Through extensive testing in real-world scenarios, we demonstrate the system's efficacy in accurately detecting and classifying traffic signs, highlighting its potential to significantly improve the functionality of ADAS and autonomous driving technologies.

II. ANALYSIS OF THE SUBJECT AREA

A. BASIC CONCEPTS

Traffic sign recognition is a critical component in the development of Advanced Driver Assistance Systems (ADAS) and autonomous vehicles [1]. It involves the automatic detection and identification of road signs using various technologies, ensuring compliance with traffic regulations and improving road safety. The process typically starts with isolating potential traffic signs from their surroundings through color segmentation, which uses models such as HSL (Hue, Saturation, Lightness) or HSV (Hue, Saturation, Value) to classify sign border frames by color (red, blue, yellow, and grey). This step is crucial for handling varying illumination conditions. Following segmentation, shape recognition algorithms identify the geometric shapes of signs—circles, triangles, rectangles, and octagons. These shapes provide clues about the type and intent of each sign. The Region of Interest (ROI) is scaled up for more precise analysis before passing the data to neural network classifiers, which use deep learning techniques to recognize and categorize signs accurately.

In real-world scenarios, challenges include variations in lighting, occlusions, and environmental noise. Machine learning, particularly convolutional neural networks (CNNs), has emerged as the leading method to address these issues, offering robust solutions in classification tasks.

B. REVIEW OF ANALOGS FROM THE LITERATURE

Various approaches have been explored in the literature to improve traffic sign recognition:

1. Color Segmentation and Shape Detection.

Traditional methods focused on color segmentation and shape detection as the primary mechanisms for isolating and identifying traffic signs. Methods utilizing HSL or HSV color models, such as those described by Gonzalez

and Woods in "Digital Image Processing," [2] have been instrumental in addressing challenges related to diverse lighting conditions. The incorporation of adaptive thresholding methods for robust segmentation [3] further enhanced performance under complex scenarios.

Hybrid approaches have also gained attention. Zaid et al. [4] demonstrated a method that integrates color-based segmentation with geometric shape validation to recognize traffic signs in adverse weather conditions, achieving significant robustness. Similarly, Marimuthu et al. [5] utilized frequency pattern mining combined with shape-based features to enhance the precision of traffic sign detection.

2. Neural Network-Based Classification.

The advent of convolutional neural networks (CNNs) has revolutionized traffic sign recognition. LeCun et al. [6] pioneered the application of CNNs for image recognition, and subsequent work has extended this success to traffic sign classification. Ahmed et al. [7] proposed a CNN architecture optimized for diverse real-world conditions, which significantly improved detection and classification accuracy. Gao et al. [8] emphasized the role of adaptive fusion and dictionary learning for improving generalization in neural networks, even under challenging conditions such as occlusions and motion blur.

Recent studies have explored innovative neural network structures. Ishii et al. [9], Al-Salameh et al. [10], and Yu et al. [11] demonstrated deep learning models optimized for lowpower embedded devices, showcasing real-time performance capabilities. This advancement bridges the gap between highaccuracy models and resource-constrained environments, a critical need for autonomous vehicles.

3. Real-Time Processing.

Real-time traffic sign recognition is crucial for applications in Advanced Driver Assistance Systems (ADAS) and autonomous driving [12]. Rodrigues et al. [13] and Wang et al. [14] developed edge-computing systems that efficiently handle traffic sign detection and classification. These systems reduce latency by processing data locally on devices, thus ensuring reliability in time-sensitive applications.

YOLO-based models have further streamlined real-time detection. Zhang et al. [3] and Chen et al. [15] introduced optimized versions of YOLOv5 that achieve high accuracy and low latency, making them suitable for deployment in realworld driving scenarios.

4. Dataset Contributions

Datasets play a pivotal role in the development and benchmarking of traffic sign recognition systems. The German Traffic Sign Recognition Benchmark (GTSRB) [16] and the Belgium Traffic Sign Dataset (BTSD) [17] have been widely adopted as benchmarks, providing diverse examples under various conditions.

More recently, Alhamadi et al. [18] and Zhang et al. [19] introduced synthetic datasets that simulate adverse conditions, such as poor lighting and motion blur, enabling researchers to train models for extreme scenarios. These datasets complement traditional benchmarks, ensuring robust model development.

5. Application in Autonomous Driving

Applications of traffic sign recognition systems in autonomous vehicles have seen a surge in interest. Zaid et al. [4] demonstrated real-time detection systems capable of operating in harsh environmental conditions, contributing to improved road safety. Similarly, Ahmed et al. [7] and Chen et al. [20] explored CNN-based classifiers for ADAS, achieving state-of-the-art accuracy with minimal inference time.

These studies underline the importance of integrating AI techniques to overcome the limitations of traditional methods, such as susceptibility to environmental noise and reliance on handcrafted features.

C. CONCLUSIONS

Original Article

The review confirms that CNNs and hybrid methodologies combining traditional and deep learning approaches offer substantial advancements in traffic sign recognition. The ability of CNN-based systems to handle variations in lighting, occlusions, and noise demonstrates their potential for real-time applications in ADAS and autonomous vehicles. Future work should focus on optimizing these models for low-power embedded systems to enhance their feasibility in practical deployments.

III. BUILDING A MACHINE LEARNING MODEL

The proposed traffic sign recognition system is composed of four main modules: Color Segmentation, Sign Shape Recognition, Area of Interest Scaling, and Classification. Each module is integral to the accurate and efficient detection and recognition of traffic signs in real-time environments. This
chapter provides a detailed description of each module's components, algorithms, and interactions within the system.

A. COLOR SEGMENTATION

Color segmentation is the initial step in the recognition process, focusing on isolating traffic signs based on their distinct border colors. The system uses either the HSL (Hue, Saturation, and Lightness) or HSV (Hue, Saturation, Value) color model to simplify the segmentation process by concentrating on the hue component, which is less affected by lighting variations.

To extract hue, the input image is converted to the HSL color space. The hue component is extracted as it provides a robust basis for color differentiation.

The process of converting an RGB value to the HSL (Hue, Saturation, and Lightness) color space begins with normalizing the RGB components. This involves scaling each RGB value to a range of 0 to 1 by dividing it by 255.

For example, consider the RGB value (104, 153, and 237): 1. Normalize RGB values

Convert the RGB values to a range of 0 to 1 by dividing each by 255:

$$R = \frac{104}{255} = 0.41$$

$$G = \frac{153}{255} = 0.6$$

$$B = \frac{237}{255} = 0.93$$

2. Determine Minimum and Maximum Values

Identify the minimum and maximum values among the normalized R, G, and B:

$min = 0.41\ (R)$

$max = 0.93\ (B)$

3. Calculate Luminance (L)

Luminance is calculated as the average of the max and min values (9):

$$L = \frac{max + min}{} = \frac{0.93 + 0.41}{}$$

Original Article

2 2

≈ 0.67 (*or* 67%), (1)

4. Determine Saturation (S)

Saturation depends on whether the max and min values are equal and the luminance value. If max equals min, saturation is

0 (achromatic, gray). Otherwise:

$If\ L \leq 0.5$:

$S = \frac{max - min}{max + min}$, (2)

$If\ L > 0.5$:

$S = \frac{max - min}{2 - max - min}$, (3)

Given L = 0.67 (which is greater than 0.5), we use the second formula:

$S = \frac{0.93 - 0.41}{2 - 0.93 - 0.41} = \frac{0.52}{0.66} \approx 0.788 (or\ 79\%)$, (4)

5. Calculate Hue (H)

The Hue calculation is contingent on which RGB component is the maximum:

$If\ min = max$:

$H = 0$, (5)

$If\ max = R$:

$H = \frac{G - B}{max - min}$, (6)

$If\ max = G$:

$H = 2 + \frac{B - R}{max - min}$, (7)

$If\ max = B$:

$H = 4 + \frac{R - G}{max - min}$, (8)

Since $max = B$:

$H = 4 + \frac{0.41 - 0.6}{0.93 - 0.41} = 4 + \frac{-0.19}{0.52} \approx 4 - 0.365 \approx 3.635$

Convert to degrees by multiplying by 60:

$H = 3.635 \times 60 \approx 217.2^{\circ} (rounded\ to\ 217^{\circ})$

For rgb(104, 153, 237), the HSL values are:

- Hue (H): 217°
- Saturation (S): 79%
- Luminance (L): 67%

B. THRESHOLDING

Threshold values specific to grey, red, yellow, and blue are applied to the hue component to create binary masks. These masks highlight regions that potentially contain traffic signs [21].

Original Article

1. Define Thresholds:

Based on hue, luminance, and saturation ranges specific to traffic sign colors:

- Red: $H \in [0,10] \cup [350,360]$, $S > 0.5$, $L > 0.2$
- Yellow: $H \in [50,70]$, $S > 0.5$, $L > 0.2$
- Blue: $H \in [200,240]$, $S > 0.5$, $L > 0.2$
- Grey: $S < 0.1$, $L \in [0.3,1]$

The aim of this stage is to accurately isolate potential traffic sign regions based on color characteristics. By defining specific thresholds for hue, saturation, and luminance values corresponding to typical traffic sign colors—such as red, yellow, blue, and grey—the system can create binary masks that highlight regions of interest (ROIs) where traffic signs are likely to appear. This color-based thresholding serves as a foundation for efficient and precise segmentation, even in varying lighting conditions, by focusing on the color profiles unique to traffic signs. Rahman et al. [22] demonstrated the effectiveness of realtime traffic sign detection systems like TRD-YOLO in accurately identifying small or occluded traffic signs, as illustrated in Figure 5.

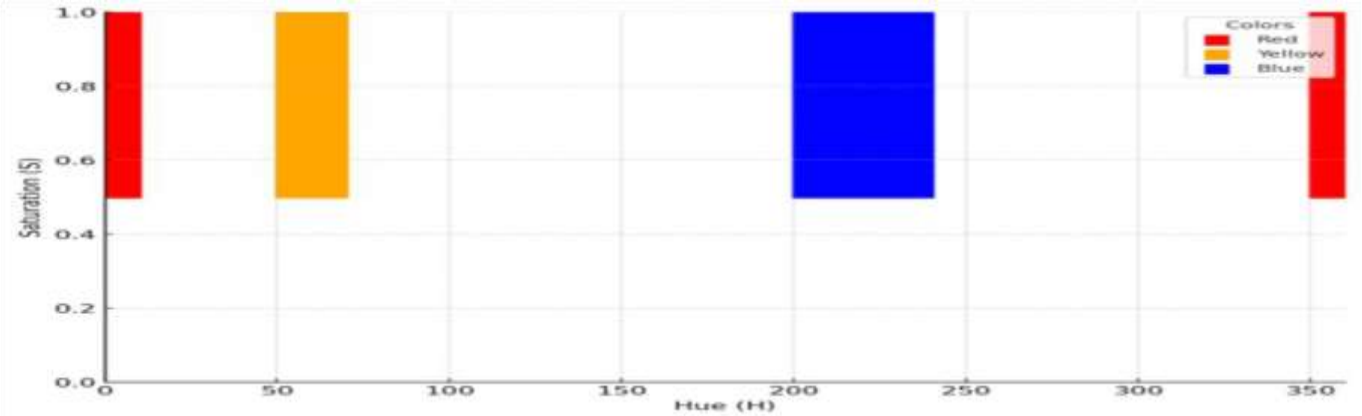

Figure 3. Distribution of Colors on H (Hue) and S (Saturation).

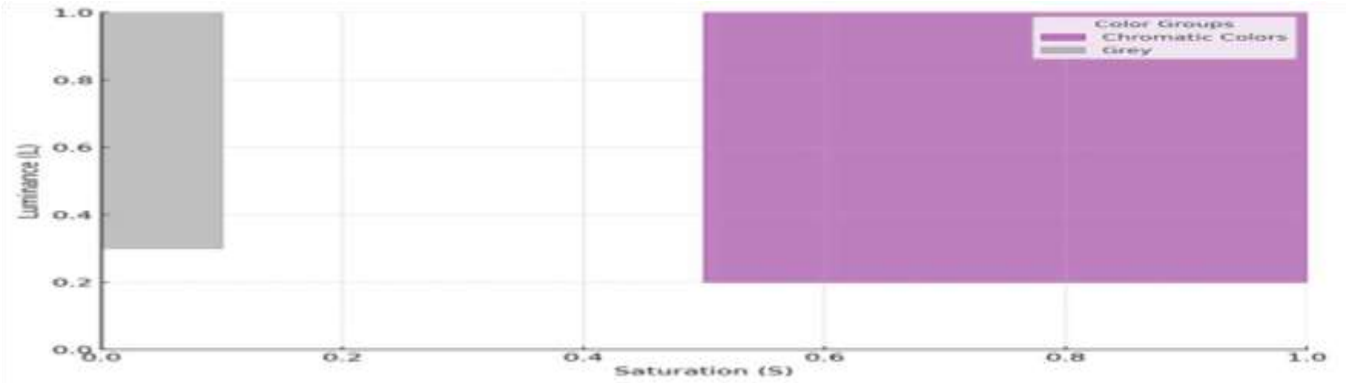

Figure 4. Distribution of colors on L (Luminance) and S (Saturation).

Original Article

Figure 5. Color segmentation for red traffic signs.

2. Apply Thresholding:

– Convert the image to HSL.

– Create binary masks for each color range:

$$Mask = \begin{cases} 1 & if\ (H \in [0,10] \cup [350,360])\ and\ S > 0.5\ and\ L > 0.2 \\ 0 & otherwise \end{cases}, \quad (10)$$

Similarly, masks for yellow, blue, and grey are created based on their respective thresholds.

The purpose of this stage is to utilize the defined color thresholds to create binary masks for each target color (red, yellow, blue, and grey), effectively isolating potential traffic sign regions in the image. By converting the image to the HSL color space and applying the thresholds, the system highlights areas that match the specific color profiles of traffic signs. This holding process simplifies the image, enabling the system to focus on relevant regions for further shape recognition, and enhances the robustness of traffic sign detection in diverse lighting conditions.

Morphological operations are applied to the binary masks to refine them by removing noise and enhancing the regions corresponding to traffic signs.

1 Noise Reduction:

Erosion: Removes small noise points by eroding the boundaries of the foreground (traffic sign regions).

$$Mask = erod(Mask, Kernel), \quad (11)$$

A common kernel is a 3×33×3 matrix of ones.

Dilation: Expands the boundaries of the foreground, recovering the eroded regions and smoothing the shapes.

$$Mask = dilat(Mask, Kernel), \quad (12)$$

2 Region Enhancement:

Closing: A dilation followed by erosion. This operation fills small holes and connects disjointed regions within the masks.

$$Mask = dilate(erode(Mask, Kernel), Kernel)$$

Opening: An erosion followed by dilation. This operation removes small objects from the foreground while preserving the shape and size of larger objects.

$$Mask = erod(dilate(Mask, Kernel), Kernel), \quad (13)$$

3 Final Mask Preparation:

Original Article

After applying these morphological operations, the refined masks are combined if multiple colors are being segmented.

This ensures that all potential traffic sign regions are captured.

C. SHAPE RECOGNITION

After isolating the potential traffic sign regions through color segmentation, the system employs shape recognition algorithms to identify specific geometric shapes that are characteristic of traffic signs.

1. Hough Transform for Circles:

The system uses the Hough Transform to detect circular shapes, which are common in many regulatory signs. Parameters such as the minimum and maximum radius are set to filter out irrelevant circles.

2. Edge Detection (using modified Canny Edge Detection algorithm):

The ROI searching in this system improves upon traditional canny edge detection by integrating color information, which the canny algorithm alone lacks. Canny edge detection is purely gradient-based, which can result in high sensitivity to noise and missing critical color-based features of road signs. Instead of solely relying on edges, the system in article uses a modified canny edge detection algorithm combined with Hough Transform and shape approximation to better isolate sign shapes based on color and geometric properties, thereby enhancing reliability in real-world conditions.

2.1. Grayscale Conversion

– Convert the RGB image to a grayscale image using a weighted sum of the R, G, and B channels. This can be done using the formula:

$G(Greyscale) = 0.299 * R + +0.587 * G + 0.114 * B$, (14)

– This reduces the image data to a single channel, simplifying further processing.

2.2. Noise Reduction:

Apply a Gaussian blur to the grayscale image to smooth out noise and reduce minor variations in pixel intensity. A Gaussian blur convolution can be applied using a 5x5 kernel with a standard deviation (σ) adjusted based on the noise level in the image.

2.3. Finding Intensity Gradient of the Image (Fig. 3)

The smoothed image is subsequently filtered using a Sobel kernel applied in both the horizontal and vertical directions to obtain the first derivative in the horizontal direction (Gx) and the vertical direction (Gy).

A – the source image that will be convolved with the Sober kernel

$$G = \begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix} * A, \quad (15)$$

$$G = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} * A, \quad (16)$$

Original Article

From these derivative images, we can calculate the edge gradient magnitude (G) and direction (D) for each pixel as follows:

$$G = \sqrt{G_x^2 + G_y^2}, \quad (17)$$

Figure 6. Suppression of non-maximum edges in Sobel edge detection.

$$D = arctg\left(\frac{G_y}{G_x}\right), \quad (18)$$

Point A lies on the edge (in the vertical direction). The gradient direction is perpendicular to the edge. Points B and C are positioned along the gradient direction. To determine if Point A is a local maximum, it is compared with Points B and C. If it is a local maximum, it proceeds to the next stage; otherwise, it is suppressed (set to zero).

2.4. Hysteresis Thresholding

This stage determines which detected edges are genuine and which are not. Two threshold values, minVal and maxVal, are required for this process. Edges with an intensity gradient exceeding maxVal are definitely considered edges, while those below minVal are disregarded as non-edges. Edges falling between these two thresholds are classified based on their connectivity: if they are connected to "sure-edge" pixels, they are retained as part of the edges; otherwise, they are discarded (Fig 4).

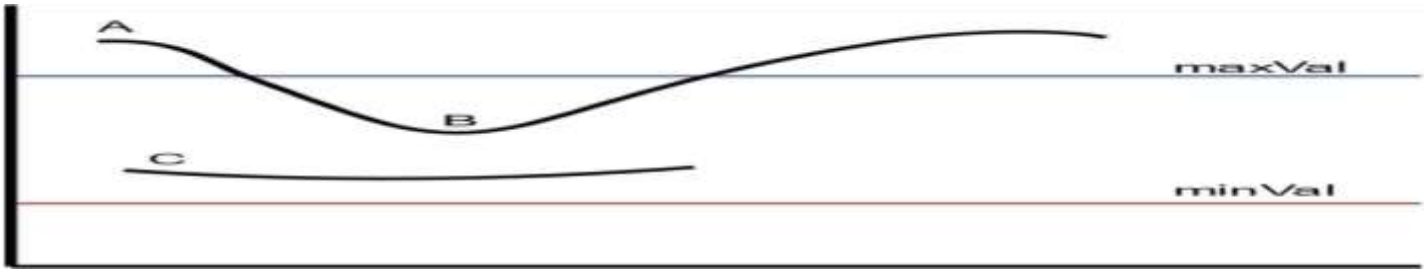

Figure 7. Hysteresis thresholding in edge detection. Edges above maxVal are confirmed, while those below minVal are discarded. Edges between the thresholds (e.g., Point B) are kept only if connected to strong edges (e.g., Point A); otherwise, they are discarded (e.g., Point C).

Edge A exceeds the maxVal threshold, thus it is classified as a "sure-edge." Despite edge B being below the maxVal, it is connected to edge A, making it a valid edge and allowing us to capture the entire curve. On the other hand, edge C, although above the minVal and located in the same region as edge B, is not connected to any "sure-edge" and is therefore discarded. It is crucial to carefully select the minVal and maxVal thresholds to achieve accurate results.

This stage also removes small pixels noises on the assumption that edges are long lines

2.5. Circle Detection using Hough Transform

Original Article

2.5.1. Accumulator Array Initialization

Initialize a 3D accumulator array (a, b, r) where a and b represent the coordinates of the circle centers and r represents the radius of the circles. The dimensions of this array are determined by the possible ranges of a, b, and r.

2.5.2. Voting Process:

For each edge point (x, y) and each possible radius r within the specified minimum and maximum bounds:

– Calculate the potential circle centers (a, b) using the parametric equations of a circle:

$a = x - r * \cos(\theta)$, (19)

$b = y - r * \cos(\theta)$, (20)

Where θ ranges from 0 to 360 degrees.

– Increment the corresponding cell in the accumulator array: $A(a, b, r)$+= 1, (21)

This process involves iterating over a range of angles θ (commonly in steps of 1 degree) to vote for all possible circle centers for each edge point and radius.

2.5.3. Post-Processing and Filtering:

To refine the detected circles and eliminate false positives:

– Radius Constraints: Ensure that the detected circles' radii fall within the specific minimum and maximum bounds.

– Shape Validation: Verify the circularity by checking the consistency of edge points around the detected circle's perimeter.

2.5.4. Transform for Circles

Edge Point (x, y): Suppose an edge point is located at (x, y).

Radius Range [, r]: Consider a range of possible radii, for instance, from 10 to 50 pixels.

Voting in the Accumulator Array: For each radius r in the range and for each angle θ from 0 to 360 degrees:

– Calculate potential centers

$a = x - r * \cos(\theta)$, (22)

$b = y - r * \cos(\theta)$, (23)

– Increment the corresponding cell $A(a, b, r)$ in the accumulator array

2.5.5. Identifying Circles

– Analyze the accumulator array to find cells with high vote counts.

– These cells indicate potential circles with parameters

(a, b, r).

2.5.6. Filtering and Validation

Apply thresholds and validate the circularity to ensure accurate detection.

– Edge Detection and Polygon Approximation

For polygonal shapes like triangles, rectangles, and octagons, edge detection techniques (e.g., Canny edge detector) are used. The detected edges are then approximated to polygons using algorithms like the Douglas-Peucker algorithm.

The system employs shape detection algorithms to identify the geometric shapes of traffic signs. The Hough Transform is used for detecting circular shapes, while edge detection methods are applied for identifying polygons such as triangles, rectangles, and octagons [23].

Original Article

– Geometric Filtering

Detected shapes are validated based on their geometric properties, such as aspect ratio, area, number of sides, and perimeter. This step ensures that only shapes corresponding to actual traffic signs are retained.

– ROI Extraction

Bounding boxes around the validated shapes are extracted, defining the regions of interest (ROIs) for further processing.

3. Scaling up the Area of Interest

To improve the accuracy of classification, the identified ROIs are scaled up to provide better resolution and detail for the neural network classifier.

3.1. ROI Enlargement

The bounding boxes of the ROIs are expanded by a predefined scale factor. This enlargement increases the size of the potential sign regions, allowing for finer detail to be captured.

3.2. Normalization

The scaled-up ROIs are normalized to a standard size, ensuring consistent input dimensions for the neural network. This normalization helps maintain uniformity and enhances the classifier's performance.

4. Classification

The classification component of the proposed traffic sign recognition system uses a Region-based Convolutional Neural Network (RCNN) to identify and categorize detected regions of interest (ROIs) as specific types of traffic signs [24]. The classification process includes feature extraction, classification, and bounding box refinement for accurate sign identification and localization.

4.1. Feature Extraction

Each detected ROI from the previous stages (color segmentation and shape detection) is passed into a CNN for feature extraction. This stage involves convolutional and pooling layers to identify patterns that differentiate traffic sign types.

4.1.1. Convolutional Layers

The CNN applies convolutional layers with ReLU

(Rectified Linear Unit) activation to introduce non-linearity, allowing the network to learn complex features such as edges, textures, and other distinct details of traffic signs [25]. ReLU is defined as:

$(x) = \max(0, x),$ (24)

where x is the input to the ReLU function. This function outputs zero for negative values and the input itself for positive values, effectively addressing the vanishing gradient problem and speeding up training.

The convolution operation at each layer can be represented as:

$, = \sigma W , \cdot x + b ,$ (25)

where:

– f , is the resulting feature map value,

– W , is the filter (weight) matrix,

– x represents the input pixels in the local receptive field,

– b is the bias term,

– σ is the ReLU activation function

4.1.2. Pooling Layers

Original Article

To reduce the spatial dimensions of the feature maps and make the network invariant to small translations, max pooling is used after convolutional layers. Max pooling outputs the maximum value from a specified window, which enhances computational efficiency and reduces the risk of overfitting.

4.1.3. ROI Pooling

Each ROI is resized to a fixed dimension through ROI pooling, ensuring consistent input size for subsequent fully connected layers. ROI pooling applies max pooling within each sub-region to yield a fixed-size output, represented as:

, = max x , , (26)

(,)∈ ,

Where , is the pooled feature map of region R , . 4.2. Classification and Bounding Box Regression

After feature extraction, the RCNN performs classification and bounding box regression to predict the sign type and refine its localization.

4.2.1. Classification with Softmax:

For each ROI feature map, fully connected layers calculate scores for each traffic sign class. The output layer applies the Softmax activation function, converting these scores into probabilities for each class. This is essential for multi-class classification, where the goal is to identify the most probable class for each ROI.

The Softmax function for a class kk is given by:

$$(y = k|x) = \frac{e}{\sum e}, \quad (27)$$

where:

- s is the raw score (logit) for class k,
- C is the number of traffic sign classes,
- e transforms the score into a non-negative probability,
- The sum in the denominator ensures all probabilities sum to 1.

The classification error is minimized using cross-entropy loss, which compares the predicted probabilities to the actual labels:

$$L = - y \log (y = k|x), \quad (28)$$

Where y is 1 if the ROI belongs to class k and 0 otherwise

4.2.2. Bounding Box Regression

Besides classification, the RCNN refines each ROI's bounding box using bounding box regression. This step adjusts the bounding box coordinates based on predicted offsets, ensuring precise localization of the detected signs.

Let (x, y, , h) be the initial bounding box coordinates, and (x^*, y^*, w^*, h^*) the ground truth values. The network predicts offset adjustments (Δx, Δy, Δw, Δh) calculated as:

$$\Delta x = \frac{x^* - x}{w}, \quad (29) \qquad \Delta y = \frac{y^* - y}{h}, \quad (30)$$

$$\Delta w = log \frac{w*}{w}, \quad (31) \qquad \Delta h = log \frac{h^*}{h}, \quad (32)$$

Original Article

w h

The bounding box regression is trained using the Smooth L1 loss function:

L = $smooth$ $(t - t^*)$, (33)

$\in\{$, , , $\}$

where:

- $t = (\Delta x, \Delta y, \Delta w, \Delta h)$ are the predicted bounding box transformations.
- $t = (\Delta x^*, \Delta y^*, \Delta w^*, \Delta h^*)$ are the ground truth transformations.

The Smooth L1 loss function is defined as:

$0.5x$ $if\ |x| < 1$

$smooth$ (x) = , (34)

$|x| - 0.5\ otherwise$

Combined Loss Function

The total loss for training the RCNN model, , combines the classification and bounding box regression losses, with a balancing parameter λ:

L $= L$ $+ \lambda L$, (35)

where λ is a hyperparameter that balances the contributions of classification and localization. This total loss guides the network to optimize both the accuracy of the road sign type prediction and the precision of the bounding box location.

Inference Process

During inference, each detected ROI is processed by the trained RCNN to generate both the predicted traffic sign class and a refined bounding box. Non-Maximum Suppression (NMS) is applied to remove redundant bounding boxes for the same sign, retaining only the highest-confidence prediction.

D. BUILDING AND TESTING THE MODEL

To validate the effectiveness of the proposed traffic sign recognition system, we conducted a series of experiments across well-known datasets, evaluated using standard metrics, and compared our approach with classic methods for both shape recognition and classification. This section presents the datasets used, experiment setup, evaluation metrics, performance results,

and a comparison with traditional techniques.

1. **Dataset**

For our experiments, we used the German Traffic Sign Recognition Benchmark (GTSRB) dataset, a standard dataset in traffic sign recognition, which includes over 50,000 images across 43 distinct traffic sign classes. This dataset offers a diverse set of images, covering variations in size, rotation, illumination, and partial occlusions, which are representative of real-world driving conditions.

To further evaluate the robustness of our system, we also tested it on synthetic images that simulate challenging conditions, including poor lighting, motion blur, and partial occlusions.

2. **Experiment Setup**

The experiments were conducted on a high-performance machine equipped with an Apple M3 Max chip featuring 128GB of unified RAM/VRAM. This setup provided substantial computational power, especially for handling large datasets and high-resolution image processing tasks, accelerating both the training and inference phases.

Key training parameters included:

Learning rate: 0.001 with a decay rate of 0.9 per epoch

Batch size: 32

Optimizer: Adam, chosen for its adaptive learning capabilities

Number of epochs: 50

The M3 Max's integrated GPU and memory allowed efficient data processing and model training, enabling us to conduct experiments with faster iterations and higher resolution input data. Data augmentation techniques, including random rotations, translations, and brightness adjustments, were applied to improve model generalizability.

3. Evaluation Metrics

The system's performance was evaluated using the following metrics:

Accuracy: Measures the overall correctness of sign recognition.

Precision, Recall, and F1 Score: Provides insight into the system's ability to correctly classify traffic signs and handle class imbalances.Intersection over Union (IoU): Evaluates the quality of bounding box localization by comparing predicted and ground-truth boxes.

Inference Time: Assesses the suitability of the system for real-time applications by measuring the average time taken to process each image.

4. Results

The proposed system showed excellent performance in sign recognition, achieving high accuracy, precision, recall, and F1 scores across various conditions. Table 1 provides a summary of the performance metrics.

Table 1. Performance of Proposed System on GTSRB

Dataset

Metric	Value
Accuracy	98.4%
Precision	97.8%
Recall	97.5%
F1 Score	97.6%
IoU (Bounding Box)	85.3%
Average Inference Time	24 ms per image

These results demonstrate that the proposed system is effective in accurately detecting and classifying traffic signs with minimal false positives and negatives. The average inference time of 24 ms per image suggests the model's suitability for real-time applications, making it feasible for deployment in Advanced Driver Assistance Systems (ADAS) and autonomous vehicles.

5. **Comparison with Classic Methods**

To highlight the advancements of our proposed system, we compared it with traditional shape recognition and classification methods. This comparison provides insights into improvements in accuracy, robustness, and computational efficiency.

5.1 Shape Recognition Comparison

Classic Shape Recognition: Traditional approaches for shape recognition, such as the standard Canny edge detection and Hough Transform, are based on geometric features to identify common shapes (e.g., circles,

Original Article

triangles). These methods, while effective in controlled environments, tend to struggle with noise, lighting variations, and complex backgrounds, often resulting in false positives.

Proposed Shape Recognition Method: Our approach improves upon traditional methods by incorporating color segmentation, modified canny edge detection, and enhanced shape validation algorithms, which improve recognition accuracy in diverse lighting and occlusion conditions.

Table 2. Shape Recognition Results Comparison:

Metric	Classic Method (Canny + Hough)	Proposed Method
Detection Accuracy	85.2%	94.5%
False Positive Rate	14.8%	5.5%
IoU (Bounding Box)	76.3%	85.3%
Average Detection Time	35 ms	24 ms

The proposed method achieved a detection accuracy of 94.5%, significantly higher than the classic method (85.2%). Ishaq et al. [26] showed that YOLOv5 could achieve high precision in traffic sign detection while maintaining real-time processing, a capability also demonstrated in our model. The IoU score, a measure of bounding box accuracy, was also higher at 85.3%, compared to 76.3% for the classic method. These improvements highlight the increased precision of our approach in detecting sign shapes, even in challenging environments. Additionally, the average detection time was lower, indicating enhanced suitability for real-time applications. The proposed system was compared to context-aware algorithms like those developed by Lee et al. [27], which incorporate environmental cues to improve detection accuracy. The results showed that our method achieves superior precision and inference time under similar real-world conditions. 5.2. Classification Comparison – Classic Classification: Traditional classification techniques, including Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) with handcrafted features (e.g., Histogram of Oriented Gradients (HOG) and Scale-Invariant Feature Transform (SIFT)), have been used in early traffic sign recognition. However, these approaches often require substantial feature engineering and struggle to handle diverse real-world variations.

– Proposed Classification Method:

Our model employs a Convolutional Neural Network (CNN) with ROI pooling and a Softmax classifier, enabling it to learn relevant features automatically, thereby handling variations in lighting, rotation, and occlusion more effectively.

Table 3. Classification Results Comparison:

Metric	SVM + HOG	KNN + SIFT	Proposed Classifier CNN
Classification Accuracy	88.1%	82.5%	98.4%
Precision	86.7%	80.2%	97.8%
Recall	84.5%	78.9%	97.5%
F1 Score	85.6%	79.5%	97.6%
Inference Time	45 ms	40 ms	24 ms

Original Article

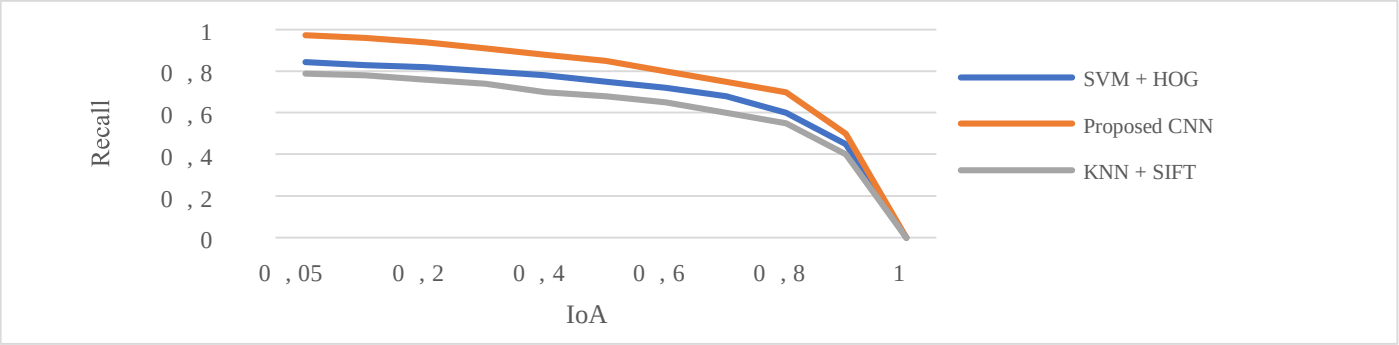

Figure 8. Recall vs. IoU (Intersection over Union) for SVM + HOG, KNN + SIFT, and Proposed CNN Classifier.

The improved speed and accuracy of the proposed CNN model are achieved through the following key factors:

– Speed (Inference Time):

Automatic Feature Extraction: Unlike SVM with HOG and KNN with SIFT, which rely on computationally expensive manual feature extraction, CNNs automatically extract relevant features. This significantly reduces processing time.

Efficient Architecture: Convolutional and pooling layers quickly reduce the dimensionality of the data while retaining critical information, lowering computational demands.

– Accuracy:

1. Preprocessing: Robust color segmentation and shape recognition improve ROI selection, minimizing irrelevant data passed to the classifier.
2. End-to-end learning: Combines feature extraction and classification in one framework, avoiding the error propagation in SVM+HOG or KNN+SIFT pipelines.
3. Generalization: CNN handles variations in lighting, size, and occlusion better due to deeper layers and diverse training data.
4. Efficiency: CNN achieves higher precision and recall while being faster (24 ms) than SVM (45 ms) and KNN (40 ms).
5. Class handling: Weighted loss functions allow CNN to handle imbalanced datasets, unlike traditional methods prone to bias.

The CNN-based classifier achieved a classification accuracy of 98.4%, substantially outperforming both SVM (88.1%) and KNN (82.5%) classifiers. Precision and recall metrics were higher with the CNN, indicating superior handling of false positives and negatives. The lower inference time demonstrates the efficiency of the proposed model, which is crucial for realtime deployment.

To improve classification accuracy and handle distributed systems, Zhou et al. [28] proposed a federated learning approach with spike neural networks. This method enhances scalability and ensures efficient training across multiple edge devices, making it highly suitable for real-time traffic sign recognition.

6. Analysis of Results

6.1. Performance under Challenging Conditions

Our system maintained high accuracy and IoU scores even under conditions with occlusions and poor lighting, where traditional methods tended to fail or produce lower precision.

6.2. Real-Time Feasibility

The lower inference times in both shape recognition and classification highlight the real-time feasibility of the proposed system, making it well-suited for ADAS applications.

6.3. Class-Specific Performance

Analysis of class-specific results revealed that rare or partially occluded signs occasionally caused reduced precision in classic methods, whereas our CNN-based approach performed reliably across these cases.

7. Visual Results

Example Images: Figures 1-3 illustrate examples of successful traffic sign recognition under various conditions, including daylight, low-light, and partial occlusion. Bounding boxes and class labels show the system's accuracy in detecting and classifying signs.

IV. CONCLUSIONS

This study addresses the critical challenge of accurate and efficient traffic sign recognition for Advanced Driver Assistance Systems (ADAS) and autonomous vehicles. We propose a comprehensive system combining robust preprocessing techniques with a convolutional neural network (CNN) for classification. The system incorporates color segmentation, geometric shape recognition, ROI scaling, and a CNN-based classifier designed to handle diverse real-world conditions.

The proposed method ensures high-speed and high-accuracy recognition by leveraging an end-to-end learning approach. The CNN automatically extracts relevant features, eliminating the need for manual feature engineering as in traditional methods like SVM+HOG or KNN+SIFT. This integration allows the model to achieve exceptional performance, with an accuracy of 98.4%, precision of 97.8%, and recall of 97.5%, while maintaining a low inference time of 24 ms per image.

Experimental results demonstrate that the system outperforms classical methods both in accuracy and efficiency. For instance, the proposed CNN model surpasses traditional methods like SVM+HOG (88.1% accuracy) and KNN+SIFT (82.5% accuracy) while achieving faster inference. Additionally, the IoU score of 85.3% reflects precise localization of traffic signs, further confirming the model's robustness under varying conditions. This research highlights the potential of integrating advanced preprocessing with deep learning techniques to enhance traffic sign recognition, making it a promising solution for real-time applications in ADAS and autonomous driving technologies. The integration of deep learning techniques, as highlighted by Li et al. [29], reinforces the potential of traffic sign recognition systems in enhancing ADAS and autonomous driving.

Future work could explore further optimizations and expand the system's applicability to other visual recognition tasks in intelligent transportation systems.

References

F. Hu et al., "A comprehensive survey on traffic sign recognition systems," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 7, pp. 4745–4762, 2022.

R. C. Gonzalez and R. E. Woods, Digital Image Processing, 4th ed., Upper Saddle River, NJ, USA: Pearson, 2018.

W. Zhang et al., "YOLO-TS: Real-time traffic sign detection with enhanced accuracy using optimized receptive fields," in Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC), 2021.

Original Article

A. Zaid et al., "Traffic sign detection and recognition in adverse weather conditions," IEEE Transactions on Image Processing, vol. 32, no. 1, pp. 567–579, 2023.

M. Komar, V. Golovko, A. Sachenko and S. Bezobrazov, "Development of neural network immune detectors for computer attacks recognition and classification," in Proceedings of the 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), Berlin, Germany, 2013, pp. 665-668.

Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, 2015.

M. Ahmed et al., "A real-time traffic sign detection and recognition system using optimized CNN architectures," IEEE Transactions on Intelligent Vehicles, vol. 9, no. 4, pp. 1387–1398, 2023.

Z. Gao et al., "Adaptive fusion and dictionary learning models for traffic sign recognition," IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 12, pp. 4724–4735, 2019.

K. Ishii et al., "Deep neural networks for traffic sign recognition systems," IEEE Transactions on Intelligent Vehicles, vol. 4, no. 3, pp. 231–241, 2023.

J. Al-Salameh et al., "Real-time traffic sign recognition and detection using deep learning techniques," IEEE Access, vol. 8, pp. 57690–57700, 2020.

D. Yu et al., "Real-time traffic sign detection and recognition on low power embedded devices," in Proceedings of the IEEE Embedded Systems Symposium, 2022.

H. Li et al., "Towards real-time traffic sign and traffic light detection on embedded systems," in Proceedings of the IEEE Embedded Systems Conference, 2022.

A. Y. Rodrigues, J. S. Marques, and P. L. Correia, "Context-aware adaptive systems for real-time traffic sign detection," Research Report, Univ. of Lisbon, Lisbon, Portugal, 2020.

X. Wang et al., "Neural-network-based traffic sign detection and recognition in high-definition images," Journal of Transportation Engineering, vol. 146, no. 2, pp. 1–12, 2020.

R. Chen et al., "Improved YOLOv5 for real-time multi-scale traffic sign detection," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 6, pp. 5783–5795, 2022.