Traffic Sign Detection using Image Processing Methods

Opruța George, Sabău Oana-Maria
Group 30232
Faculty of Automation and Computer Science
Opruta.Gl.George@student.utcluj.ro, Sabau.Io.Oana@student.utcluj.ro

***Abstract*— This paper explores the concept of traffic sign detection using traditional image processing techniques and details the proposed image processing pipeline. Additionally, post-processing techniques based on size and aspect ratio filtering are employed. The paper then presents the experimental results obtained by applying the proposed method to a set of test images. It analyzes the success rate in detecting traffic signs and discusses factors affecting performance. Finally, it concludes by summarizing the achievements, limitations, and potential for future development of this image processing-based traffic sign detection system.**

I. Introduction

Traffic sign detection plays a critical role in intelligent transportation systems. These systems rely on automatic interpretation of visual information from the environment to enhance road safety and traffic flow.

While advancements in Artificial Intelligence (AI) have yielded highly effective methods for traffic sign detection, this paper explores an alternative approach: utilizing traditional image processing techniques.

AI-based methods, particularly those employing deep learning architectures, offer exceptional accuracy in traffic sign detection. However, they often come with significant drawbacks. This can be a barrier for real-world deployments with limited processing power or data availability.

A. Purpose of the Study and Research Objectives

This study aims to explore the effectiveness of utilizing traditional image processing techniques for traffic sign detection. The primary objectives of this project are:

- To design and implement an image processing pipeline for traffic sign detection that is computationally efficient.
- To evaluate the performance of the proposed system using various test datasets encompassing diverse lighting conditions, backgrounds, and traffic sign types.
- To analyze the strengths and limitations of the image processing-based approach compared to AI methods for traffic sign detection.

B. Outline of the Paper

This paper is organized as follows:

1) **Introduction**: This section provides an overview of traffic sign detection and its importance in various applications. It also motivates the exploration of traditional image processing techniques.
2) **Literature Review**: This section reviews existing research on traffic sign detection using image processing methods. It discusses various color segmentation, shape detection, and post-processing approaches implemented in prior works.
3) **Design and Implementation**: This section details the proposed image processing pipeline for traffic sign detection. It describes the pre-processing steps like color space conversion and noise reduction, followed by color segmentation, shape analysis, and post-processing techniques used for filtering potential sign regions.
4) **Experimental Results**: This section presents the results obtained by applying the proposed method to a set of test images. It discusses the success rate in detecting traffic signs and analyzes factors affecting performance.
5) **Discussion**: This section analyzes the effectiveness of the proposed approach, highlighting its strengths and limitations. It compares the performance with existing techniques and discusses potential improvements.
6) **Conclusion**: The implications of the study for traffic sign segmentation applications are discussed, and suggestions for future research directions are provided.
7) **References**: Lists the references cited throughout the paper.

II. Literature Review

In this chapter, we review the existing body of research on traffic sign detection and discuss the methods relevant. We also summarize the methodologies and findings of previous studies, providing a foundation for our research.

A. Existing Studies on Traffic Sign Detection

Traffic sign recognition (TSR) is a crucial technology within Advanced Driver-Assistance Systems (ADAS), enabling vehicles to interpret traffic signs and react accordingly. Various automotive suppliers are actively developing TSR systems, primarily relying on image processing techniques for sign detection. [1]

These detection methods can be broadly categorized into three groups:

1) **Color-based methods**: Traffic signs often exhibit distinct colors compared to their surroundings. Techniques exploit this property by defining color ranges in color spaces like HSV (Hue, Saturation, Value) to

segment potential sign regions. For instance, a red hexagonal shape in an image might be flagged for further analysis as a potential stop sign.

2) **Shape-based methods**: Traffic signs typically adhere to specific geometric shapes like circles, rectangles, or triangles. Techniques like Hough Transform for circles or contour analysis can be employed to identify these shapes within segmented regions. For example, the Hough Transform can be used to detect the circular shape of a speed limit sign.
3) **Learning-based methods**: This category encompasses approaches that leverage machine learning algorithms to learn the characteristics of traffic signs from labeled datasets. Early methods might utilize features like Haar-like features or Freeman Chain codes for character recognition within signs. These features capture specific patterns within the sign image, allowing the algorithm to learn and recognize characters like numbers or letters on the sign.

Modern TSR systems are increasingly adopting deep learning, particularly Convolutional Neural Networks (CNNs). This shift is driven by the growing need for autonomous vehicles to identify a vast variety of traffic signs, not just speed limits.

Here's how deep learning contributes to TSR advancements:

1) **Vienna Convention as a Guide**: The Vienna Convention on Road Signs and Signals provides a standardized set of traffic signs. Deep learning models can be trained on these predefined signs to "learn" their characteristics using techniques like convolutional layers and backpropagation. These layers automatically extract features from image data, allowing the model to learn the intricacies of different traffic signs.
2) **Image Processing and Computer Vision**: Deep learning models leverage image processing and computer vision techniques to extract meaningful features from images. These features might include edges, shapes, or color patterns. The model then uses these features to learn and distinguish between different traffic signs.
3) **Real-Time Applications**: Once trained, the neural network can be deployed on embedded systems within vehicles to detect traffic signs in real-time. This enables autonomous vehicles to react appropriately to changing road conditions, such as slowing down for a stop sign or adjusting speed based on a detected speed limit sign.
4) **Data Acquisition and Collaboration**: self-driving car companies like Waymo and Uber, along with map and navigation companies like Tom Tom, are actively generating and outsourcing large traffic sign datasets. This collaborative effort provides the necessary training data for deep learning models. The more data a model is trained on, the more accurate and efficient it becomes in recognizing traffic signs in real-time traffic scenarios. [1]
   Our main inspiration and study was the article *Traffic sign segmentation with classical image processing methods (Canny edge detection + color based segmentation)* [2].
   The blog post advocates for a simple image processing pipeline for traffic sign segmentation, leveraging techniques well-suited for real-time applications with limited computational resources. The core steps involve:
   a) *Image Preprocessing*: This stage prepares the image for subsequent processing. It may include denoising, contrast enhancement to improve feature visibility, and resizing to a standard size for efficient processing.
   b) *Canny Edge Detection*: This technique identifies edges within the image, potentially corresponding to the boundaries of traffic signs. The Canny edge detector is known for its ability to balance edge preservation and noise suppression.
   c) *Color-Based Segmentation*: Traffic signs often exhibit distinct colors compared to their surroundings. This approach defines color ranges in color spaces like HSV (Hue, Saturation, Value) to segment potential sign regions. The blog post highlights the importance of carefully selecting these color ranges for optimal performance.
   d) *Shape Detection*: Since traffic signs typically adhere to specific geometric shapes (circles, rectangles, triangles), techniques like Hough Transform for circles or contour analysis are employed to identify these shapes within the segmented regions.
   e) *Contour Detection and Filtering*: This stage involves extracting contours from both the edge detection and color segmentation outputs. Typically, only the largest contour (by area) is retained as the potential traffic sign region.
   f) *Bounding Box and Intersection over Union (IoU)*: A bounding box is drawn around the identified traffic sign region. IoU, a metric to quantify the segmentation quality, is calculated to measure the overlap between the predicted bounding box and the ground truth (actual) location of the sign.

   The blog post highlights the advantages of this classical image processing approach like interpretability, efficiency, sensitivity to illumination and background complexity, limited shape handling. [2]
   Another article, *Automatic Traffic Sign Detection and Recognition Using Colour Segmentation and Shape Identification* [3], breaks down the same approach as described above and serves as a main inspiration for our implementation.

III. Design and Implementation

The dataset is from Chinese Traffic Sign Recognition Database [4]. The workflow was inspired from the main article of interest [2].

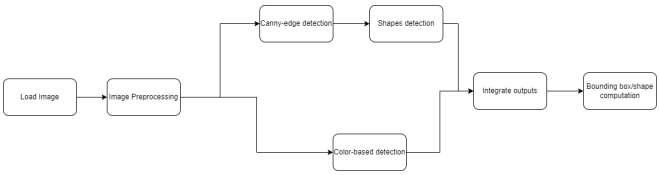

Fig. 1. Block diagram of the application

A. Color-based Segmentation

Steps for Color-Based Segmentation

Define HSV Color Space Tuples: define the lower and upper HSV color space tuples for each color. These tuples will be the input arguments for the `cv.inRange()` function. Geometrically, these tuples define boxes in the HSV color space. Generate Color Masks: the four outputs from `cv.inRange()`, which correspond to the four color masks.

Apply Morphological Operations: apply closing to the output of the previous step to join the breaks.

Function Descriptions

isInsideB

Checks if the coordinates (x, y) are within the bounds of the image matrix `src`.

axisOfElongation

Calculates the angle of elongation of a specific color region in the image. It iterates through all pixels, and for pixels matching the specified color (`pixel`), it updates the numerator and denominator based on their relative positions to a reference point (r_i, c_i). The final angle is computed using the `atan2` function.

thinnessRatio

Computes the thinness ratio, a shape descriptor, using the area and perimeter of a region. The formula used is:

$$\text{Thinness Ratio} = 4\pi \left(\frac{\text{area}}{\text{perimeter}^2} \right)$$

areaPlusCentereOfMass

Calculates the area and the center of mass (centroid) of a specified color region in the image. It sums up the row and column indices for the matching pixels and divides by the area to find the centroid.

perimeterPixel2

Determines if a pixel at (x, y) is on the image boundary of the specified color region. It checks the neighboring pixels to see if any of them are not the specified color.

perimeter2

Computes the perimeter of the specified color region in the image. It iterates through all pixels, and if a pixel belongs to the region and is a boundary pixel (checked using `perimeterPixel2`), it increments the perimeter counter and marks it on the destination matrix `dst`.

normalizeColor

Normalizes the color of non-black pixels to a specified color (`color`).

erosion

Performs morphological erosion on a binary image. It thresholds the image to binary, then processes each pixel to determine if it should remain based on its neighbors.

signDetection

This is the main function which handles the detection of different traffic sign colors (red, blue, yellow). It performs the following steps:

- Loads the image and converts it to HSV color space.
- Applies color thresholding to create masks for red, blue, and yellow regions.
- For each color region:
  - Normalizes the detected regions to a single color.
  - Computes perimeter, area, and centroid.
  - Calculates the thinness ratio and axis of elongation.
  - Based on the predominant color, classifies the sign and draws bounding boxes.
- Displays the results including the original image, processed regions, and detected signs.

B. Shapes Detection

It is based on Canny-edge detection implemented as a laboratory work and different algorithms to recognize the shapes. Those were implemented using OpenCV functions.

Function Descriptions

The **cnt_rect** function is designed to detect rectangles from a list of contours. It uses the Douglas-Peucker [5] algorithm to approximate each contour, and if the approximation results in a shape with four points, it is identified as a rectangle. Among these approximations, the function selects the largest rectangle based on its area and stores its points in the result vector. If no rectangles are detected, the result is cleared.

The **cnt_circle** function focuses on detecting circles within an image. It applies the Hough Circle Transform to identify potential circles and selects the largest one based on its radius. Once the largest circle is identified, it creates a mask and finds the contours within this mask. The function then chooses the largest contour, assuming it represents the circle, and stores its points in the result vector. If no circles are detected, the result is cleared.

The **integrate_circle_rect** function integrates the results of rectangle and circle detections. It first checks if both inputs are empty, returning an empty matrix if so. If both shapes are detected, it prioritizes the one with the larger area. If only one shape is detected, it returns that shape.

The **integrate_edge_color** function compares two detected shapes by their contour areas and returns the larger one. If one of the shapes is not detected, it returns the detected shape. If neither shape is detected, it returns an empty matrix.

The **test_shape_detection** function is a comprehensive test routine for shape detection. It opens an image file, converts

it to grayscale, and applies the Canny edge detector to find edges. It then detects rectangles and circles using the cnt_rect and cnt_circle functions. The results are integrated using the integrate_circle_rect function. If a shape is detected, it draws the contour on the original image and displays the result. This process repeats for each image file selected.

IV. Experimental Results

After the implementation, we tested both of the approaches and obtained satisfying results due the fact the photos were quite bad quality.

The color segmentation method finds the element based on the predominant color in the image and has good results and tells what traffic sign might be.

Fig. 2. Color-based segmentation result

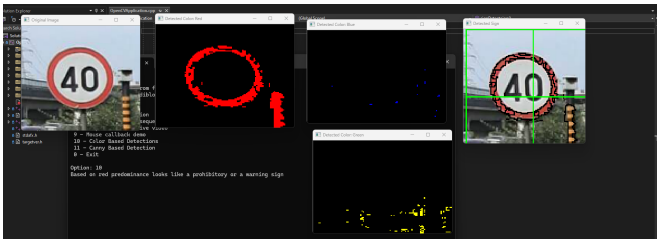

Fig. 3. Color-based segmentation with the color mask

Its weak point is that it doesn't always identify the right colors.

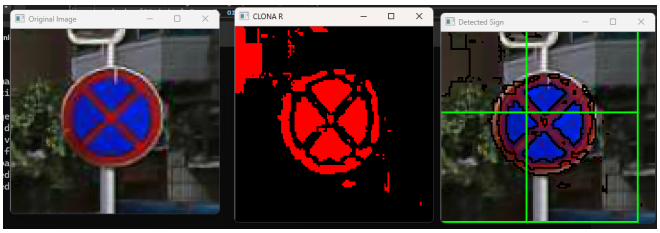

Fig. 4. Color-based segmentation with the color mask - Bad result

The shapes detection approach seemed to work better, finding the circle-shaped traffic signs, but has some limitations: doesn't always find rectangles and squares and doesn't work on triangle-shaped trafiic signs.

This approach doesn't always find the traffic sign correctly because the image is low quality and canny-edge doesn't extract the edges the right way, as shown below. The circle shape isn't extracted correctly from the source image.

Fig. 5. Image resulted after applying Canny-Edge Detection

Fig. 6. Shaped detected traffic sign - Good result

Fig. 7. Shaped detected traffic sign - Bad result

V. Discussion

This section wraps up the topics discussed earlier and interprets and compares the results.

A. Strengths and Advantages

The primary strength of the proposed image processing approach lies in its **computational efficiency**. The reliance on well-established image processing techniques minimizes the processing power required compared to AI methods, making it suitable for deployment on resource-constrained platforms.

Another key strength is the **interpretability** of the image processing pipeline. By breaking down the process into distinct stages like color segmentation, shape analysis, and post-processing, each step contributes in a clear and understandable way to the final detection outcome. This allows for targeted modifications and optimizations for specific applications or sign types.

B. Limitations and Trade-offs

While the proposed approach offers advantages, it's important to acknowledge its limitations. Compared to deep learning, the accuracy of image processing techniques might

be lower, especially under challenging conditions. Complex backgrounds with clutter or variations in lighting can pose difficulties for the system. Additionally, the ability to handle a wider variety of traffic sign types might be limited compared to AI models trained on extensive datasets encompassing diverse signs.

C. Comparison with AI-based Methods

The choice between image processing and AI methods for traffic sign detection depends on the specific application requirements. Here's a breakdown of the key considerations:

Computational Resources: If processing power is limited, image processing offers a clear advantage due to its efficiency.

Data Availability: When large labeled datasets are readily available, AI methods can achieve superior accuracy. However, if data is scarce, image processing becomes a viable alternative.

Real-time Applications: For real-time scenarios where processing speed is crucial, the efficiency of image processing can be advantageous.

Interpretability and Customization: If understanding and potentially modifying the detection process is important, the interpretable nature of image processing provides a benefit.

D. Future Considerations

This research opens doors for further exploration in the domain of image processing-based traffic sign detection. Here are some potential areas for future development:

Enhancing Robustness: Investigating techniques like incorporating background subtraction or illumination normalization to improve performance under challenging conditions.

Leveraging Domain Knowledge: Exploring the integration of prior knowledge about specific traffic sign shapes and colors to guide the detection process and enhance accuracy.

Real-time Implementation: Optimizing the image processing pipeline for real-time applications on embedded systems with limited resources.

VI. CONCLUSION

This paper investigated the effectiveness of traditional image processing techniques for traffic sign detection. While acknowledging the exceptional accuracy achieved by AI methods, this work explored an alternative approach that offers potential benefits for real-world deployments with limitations.

The proposed image processing pipeline, designed to be computationally efficient and require minimal training data, achieved suitable results in detecting traffic signs within various test scenarios. Our contribution was to implement some of the functions (e.g. Canny-edge detection) instead of using OpenCV functions to see the flow of the program.

By addressing the identified challenges and exploring the suggested research directions, future work can significantly advance the field, leading to more accurate and efficient traffic sign detection systems. This progress will be crucial for developing robust and reliable tools that can effectively handle the diverse and dynamic nature of traffic signs.

REFERENCES

[1] "Traffic-sign recognition", Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Traffic-sign_recognition

[2] Lim, J., "Traffic sign segmentation with classical image processing methods (Canny edge detection + color based segmentation)." Retrieved from *https://jq0112358.medium.com/traffic-sign-segmentation-with-classical-image-processing- methods-canny-edge-detection-color-8ff1096535db*

[3] Horak, K., Cip, P., Davidek, D., "Automatic Traffic Sign Detection and Recognition Using Colour Segmentation and Shape Identification ", MATEC Web of Conferences 6, ICIEA 2016, 8 17002 (2016). Retrieved from https://www.matec-conferences.org/articles/matecconf/pdf/2016/31/matecconf_iciea2016_17002.pdf

[4] Chinese Traffic Sign Database. Retrieved from https://nlpr.ia.ac.cn/pal/trafficdata/recognition.html

[5] "Ramer–Douglas–Peucker algorithm", Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm