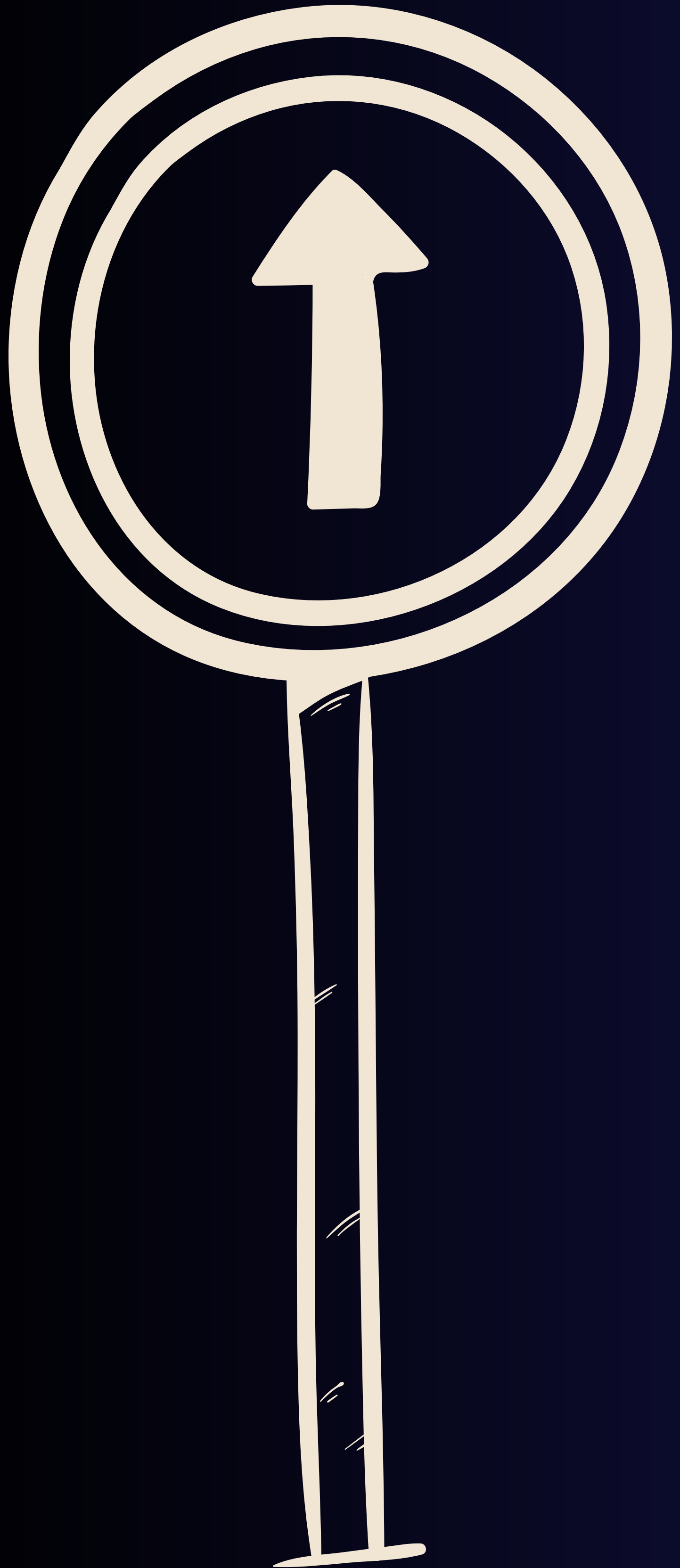
GÖRÜNTÜ İŞLEYİCİLERİ

Traffic Sign Recognition

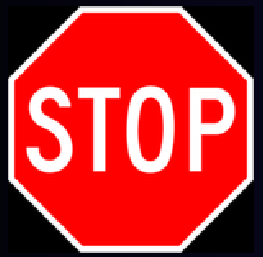
CSE 464 Term Project



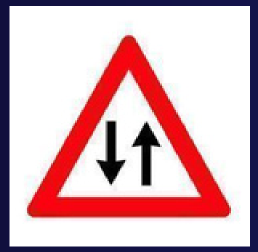
Traffic sign recognition is a critical feature for autonomous systems, enabling vehicles to follow road regulations and ensure safety.

This project aims to recognize traffic signs using manual image processing techniques instead of prebuilt libraries like OpenCV; by focusing on **edge detection**, **contour extraction** and **shape matching**.

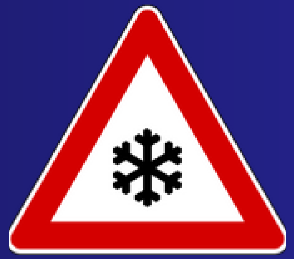
The system currently supports 16 traffic sign templates:



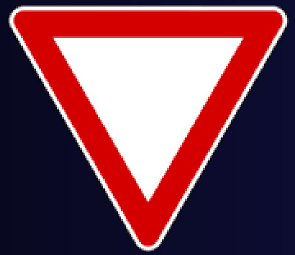
Stop



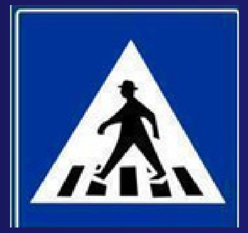
Two Way



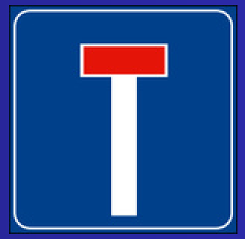
Snow Warning



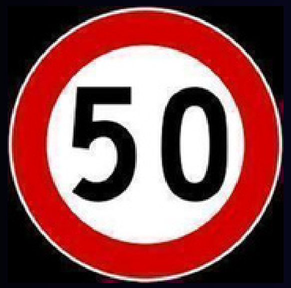
Yield



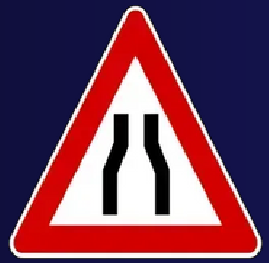
Pedestrian Crossing



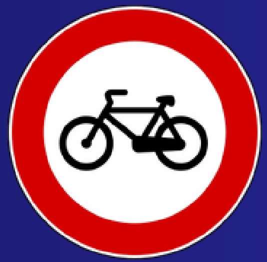
Dead End



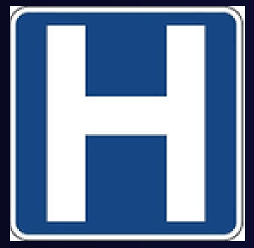
Speed Limit



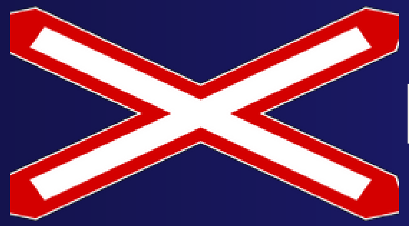
Road Narrows Ahead



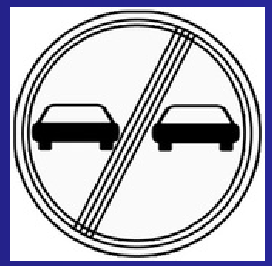
Cyclists Not Permitted



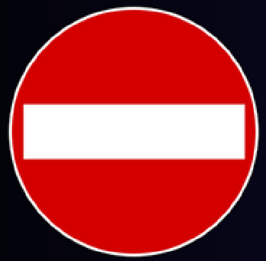
Hospital



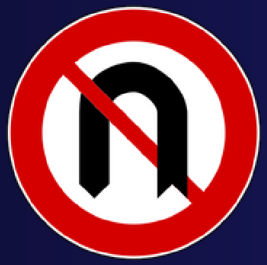
Rail Crossing



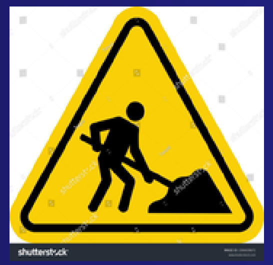
End Of The Overtaking Prohibition



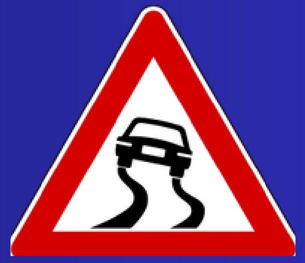
No Entry



No U Turn Allowed



Road Work



Slippery Road Ahead

Image Preprocessing

The first step in the system is preprocessing the input images.

01

Grayscale Conversion

The image is converted to grayscale to simplify data, keeping only intensity information. The formula used ensures accurate brightness representation based on red, green, and blue channels.

02

Gaussian Blurring

Noise is reduced using a manually implemented Gaussian filter. This step ensures that sharp variations are smoothed, allowing better edge detection later.

Edge Detection

Edge detection is performed using a custom Canny algorithm.

01

Gradient Computation

Sobel operators calculate intensity changes in horizontal and vertical directions, providing the gradient magnitude and direction for each pixel.

02

Non-Maximum Suppression

Only the strongest edges aligned with the gradient direction are kept, removing redundant pixels.

03

Double Thresholding and Edge Tracking

Edges are classified as strong or weak based on intensity. Weak edges connected to strong ones are retained, while others are discarded.

Contour Extraction

Contours are extracted from the edge-detected image to identify the sign's shape.

01

Contour Tracing

A depth-first search algorithm traces connected edge pixels, forming contours. Small contours with less than 10 pixels are ignored to avoid noise.

02

Normalization

The extracted contours are scaled and centered, ensuring size and position differences do not affect shape matching.

Shape Matching

The system matches input contours with template contours using weighted similarity metrics.

01

Weighted Metrics

The top **5** largest contours are compared, with larger contours receiving higher weights. This approach ensures that the most important shapes are prioritized during matching.

02

Similarity Calculation

Contours are normalized, and a mean squared distance metric is used to measure the difference between the input and template shapes.

Results

The system mostly achieves success in recognizing traffic signs.

A.

Challenges

- Weak edges sometimes affect accuracy.
- Large image boundaries are sometimes misidentified as shapes.
- Occasional mismatches occurred due to noise or overlapping shapes.

B.

Opportunities

- Prominent **edges** are successfully identified.
- **Contours** are traced effectively.
- Example: The system correctly **matched** "Slippery Road Ahead".

Conclusion

This project recognizes traffic signs using **custom, library-free** image processing.

The process begins with **grayscale conversion and Gaussian blurring** to prepare the image, followed by **edge detection** using a handcrafted Canny algorithm.

Contours are then extracted and compared to predefined templates using **weighted similarity metrics**.

By **matching the shapes** of input signs to templates, the system identifies traffic signs.

The project highlights the effectiveness of manually implemented techniques in traffic sign recognition without relying on external libraries like OpenCV.

Thank you!

Buse Deniz Hanaylı
Serdar Genç