VIETNAM GENERAL CONFEDERATION OF LABOR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

NGUYEN QUANG HUY - 523H0140

NGUYEN DANG NAM KHANH - 523H0148

FINAL REPORT

DIGITAL IMAGE PROCESSING

HO CHI MINH CITY, 2025

VIETNAM GENERAL CONFEDERATION OF LABOR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

NGUYEN QUANG HUY - 523H0140

NGUYEN DANG NAM KHANH - 523H0148

FINAL REPORT

DIGITAL IMAGE PROCESSING

Advised by

Dr. Nguyen Chi Thien

HO CHI MINH CITY, 2025

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Dr. Nguyen Chi Thien, our instructor and mentor, for his valuable guidance and support throughout the report. He has been very helpful and patient in providing us with constructive feedback and suggestions to improve our work. He has also encouraged us to explore new technologies and techniques to enhance our system's functionality and performance. We have learned a lot from his expertise and experience in web development and software engineering. We are honored and privileged to have him as our teacher and supervisor.

Ho Chi Minh city, 27th November 2025.

Author

(Signature and full name)

Huy

Nguyen Quang Huy

Khanh

Nguyen Dang Nam Khanh

DECLARATION OF AUTHORSHIP

We hereby declare that this is our own project and is guided by Dr. Nguyen Chi Thien. The content research and results contained herein are central and have not been published in any form before. The data in the tables for analysis, comments and evaluation are collected by the main author from different sources, which are clearly stated in the reference section.

In addition, the project also uses some comments, assessments as well as data of other authors, other organizations with citations and annotated sources.

If something wrong happens, we'll take full responsibility for the content of my project. Ton Duc Thang University is not related to the infringing rights, the copyrights that We give during the implementation process (if any).

Ho Chi Minh city, 27th November 2025.

Author

(Signature and full name)

Huy

Nguyen Quang Huy

Khanh

Nguyen Dang Nam Khanh

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1. METHODOLOGY OF SOLVING TASKS

1.1 Introduction and Approach

The objective of this project is to automatically detect and recognize traffic signs from input videos. The system focuses on common traffic sign categories such as:

- Circular signs (prohibitory and mandatory, often in red or blue),
- Triangular warning signs (typically yellow with a red border),
- Rectangular informational or directional signs

Based on the requirements, we do not use Deep Learning models. Instead, we use traditional image processing techniques using the OpenCV library. The overall process consists of three main stages:

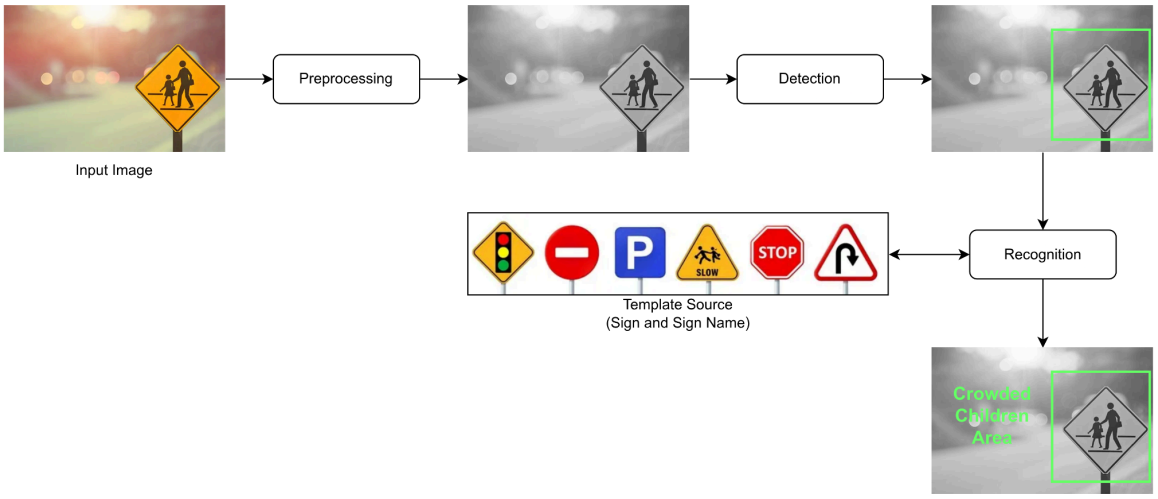

Figure 1.1.1. Solving Method

1. **Preprocessing:** Enhancing the image quality and converting color spaces.
2. **Detection:** Finding the location of traffic signs using color masking and shape analysis.
3. **Recognition:** Identifying the specific meaning of the sign using Template Matching.

1.2 Background Knowledge and Methods

To solve the problem, the following computer vision concepts and methods were applied:

1. **Gamma Correction:** This is a non-linear operation used to adjust the brightness of the image. It helps the system work better in different lighting conditions (too dark or too bright).
2. **Gaussian Blur:** This technique allows us to smooth the image and reduce noise. It removes high-frequency details that might interfere with edge detection.
3. **HSV Color Space:** Unlike the RGB model, HSV (Hue, Saturation, Value) separates color information (Hue) from brightness (Value). This makes it much easier to detect specific colors like Red or Blue regardless of shadows or light intensity.
4. **Color Thresholding and Mask Combination:** The system defines HSV ranges for red, blue, and yellow. Individual masks are created by cv2.inRange, and then combined using cv2.bitwise_or into a single binary mask that highlights all potential sign regions.
5. **Morphological Operations:** A morphological opening operation (cv2.morphologyEx with cv2.MORPH_OPEN) is applied using a small kernel. This step removes small noisy blobs from the combined mask and helps keep only meaningful connected regions.
6. **Contour Detection:** This method finds the boundaries of objects in the binary mask.
7. **Shape Approximation:** Using cv2.approxPolyDP to determine if a shape is a triangle (3 vertices), a rectangle (4 vertices) or using circularity math to find circles.
8. **Template Matching:** A method for searching and finding the location of a template image inside a larger image. We use cv2.matchTemplate to compare the detected region with reference images.

1.3 Detailed Implementation Steps

The source code is organized into a FrameProcessor class that handles individual frames and a VideoProcessor class that handles video input/output. Below is the detailed explanation of the programming steps:

Step 1: Image Preprocessing

- Function: preprocess(self, frame, ...)
- Explanation:
  - First, we apply Gamma Correction (adjust_gamma). The code calculates a lookup table to adjust pixel values. This makes dark traffic signs clearer.
  - Next, Gaussian Blur (cv2.GaussianBlur) is applied with a kernel size of (5,5) to reduce noise.
  - Finally, the image is converted from BGR to HSV (cv2.cvtColor).

Step 2: Color Segmentation (Masking)

- Function: masks_color(self, preprocessed_frame)
- Explanation:
  - We define specific ranges for 3 main colors: red, blue, yellow in HSV space.
    - For Red: Since red is at both the beginning and end of the Hue scale (0-180), we use two ranges: lower_red1 to upper_red1 and lower_red2 to upper_red2. We combine them using cv2.bitwise_or.
    - For Blue: We use a single range (lower_blue to upper_blue) to capture blue signs.
    - For Yellow: Another interval is used to extract yellow warning signs.
  - Combine all masks (red, blue, and yellow) into one unified binary mask using cv2.bitwise_or. In this mask, pixels belonging to any of the specified color ranges are set to white (255), and the rest are black (0).
  - Apply a morphological opening operation with a small (3, 3) kernel using cv2.morphologyEx. This reduces small isolated noise spots and preserves coherent object regions.

- Result: The function returns the combined cleaned mask that highlights all potential sign regions.

Step 3: Finding Regions of Interest (ROI) and Shape Classification

- Function:
  - bounding_boxes(self, mask, frame)
  - valid_bounding_box(self, box)
  - roi_frame(self, masks, frame, frame_idx)
- Explanation:
  - We find contours in the mask using cv2.findContours.
  - We filter out small noise by ignoring contours with an area less than 900.
  - Shape Classification: For the remaining contours, the system determines the geometric shape:
    - Rectangle (quadrilateral): If the contour area is relatively large (e.g., > 7500) and the approximated polygon (cv2.approxPolyDP) has 4 vertices, it is classified as a rectangle.
    - Triangle: If the approximated polygon has 3 vertices, the three side lengths are computed. If all three sides are similar (within 30% deviation from the average length), the contour is classified as a triangle.
    - Circle: For other contours, the minimum enclosing circle is computed (cv2.minEnclosingCircle). The circularity is defined as the ratio between the contour area and the area of the enclosing circle. If this ratio exceeds a certain threshold (e.g., 0.67), the region is classified as a circle.
  - The function roi_frame uses the validated bounding boxes to crop the original frame and produce a list of regions: (roi image, bounding box, shape type) where the shape type is "circle", "triangle", or "rectangle".

Step 4: Sign Recognition (Template Matching)

- Function: detected_label(self, roi, type) and template_matching(...)

- Explanation:
    - We crop the image frame to the bounding box (ROI).
    - The system loads reference images (templates) from the train_image folder based on the type ("circle", "triangle", or "rectangle").
    - Multi-scale Matching: Since the traffic sign in the video might be smaller or larger than the template, the code resizes the template to different ratios (1.0, 0.9, 0.8) using resize_template.
    - We use cv2.matchTemplate with the method cv2.TM_CCOEFF_NORMED. This returns a score indicating how well the template matches the ROI.
    - If the highest score exceeds the threshold (0.3), the label is accepted.

Step 5: Visualization and Output

- Function: draw_bounding_boxes(...) and video_process(...)
- Explanation:
    - If a label is found, the code draws a green rectangle around the sign (cv2.rectangle) and writes the text name of the sign (cv2.putText).
    - The VideoProcessor reads the input video frame by frame, calls the FrameProcessor, and saves the result to a new .mp4 file.

CHAPTER 2. Task Results

2.1 Description of Results

The program successfully processes video1.mp4 and video2.mp4. The output videos show that the system can:

1. Detecting traffic signs in various positions.
2. Distinguish between Red (Warning/Prohibition), Blue (Instruction) signs and Yellow (triangular warning).
3. Identify specific meanings such as "No entry", "No parking", "Go slow", etc.
4. Draw bounding boxes and display text labels in real-time.

2.2 Visual Results

Below are 10 sample frames extracted from the output videos, showing the detection results at different times (with a minimum gap of 4 seconds between frames).

Output Video 1 Results:

1. Frame at 00:02: Successfully detected “Way street tutorial”.

Figure 2.2.1. Frame at 00:02 detected Way street tutorial

2. Frame at 00:07: Successfully detected "No entry", “Keep right” and “No turn left”.

Figure 2.2.2. Frame at 00:07 detected No entry, Keep right, No turn left

3. Frame at 00:18: Successfully detected "No entry" and “Keep right”.

Figure 2.2.3. Frame at 00:18 detected No entry, Keep right

4. Frame at 00:26: Successfully detected "No entry" and “Keep right”.

Figure 2.2.4. Frame at 00:26 detected No entry, Keep right

5. Frame at 00:34: Successfully detected "No entry" and “Keep right”.

Figure 2.2.5. Frame at 00:34 detected No entry, Keep right

6. Frame at 00:41: Successfully detected "No entry", “Keep right” and “No turn left”.

Figure 2.2.6. Frame at 00:41 detected No entry, Keep right, No turn left

7. Frame at 00:46: Successfully detected “No turn left”.

Figure 2.2.7. Frame at 00:46 detected No turn left

8. Frame at 01:03: Successfully detected "No entry", “Keep right” and “No stopping and parking”.

Figure 2.2.8. Frame at 01:03 detected No entry, Keep right, No stopping and Parking

9. Frame at 01:22: Fail detected "Go slow".

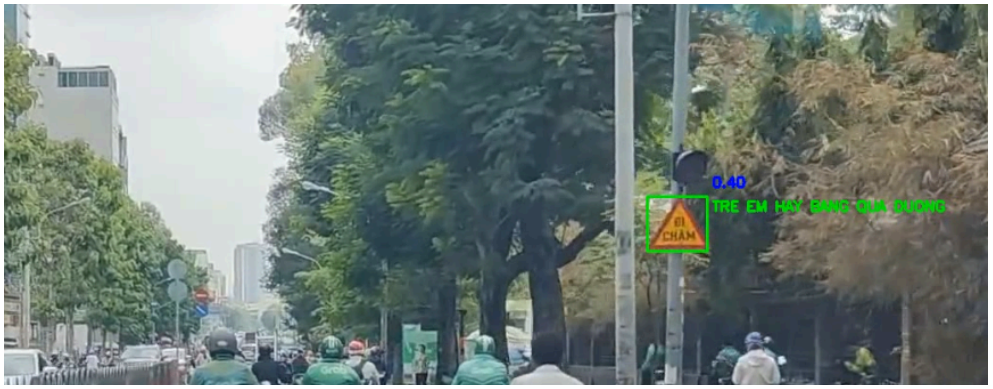

Figure 2.2.9. Frame at 01:22 Fail detected Go slow

10. Frame at 01:30: Successfully detected "No entry", "Keep right" and "No Parking".

Figure 2.2.10. Frame at 01:30 detected No entry, Keep right, No parking

Output Video 2 Results:

1. Frame at 00:03: Successfully detected "No parking".

Figure 2.2.11. Frame at 00:03 detected No parking

2. Frame at 00:17: Successfully detected "No parking".

Figure 2.2.12. Frame at 00:17 detected No Parking

3. Frame at 00:22: Successfully detected "Go slow", “Crowd children area”.

Figure 2.2.13. Frame at 00:22 detected Go slow, Crowd Children Area

4. Frame at 00:55: Successfully detected "No parking”.

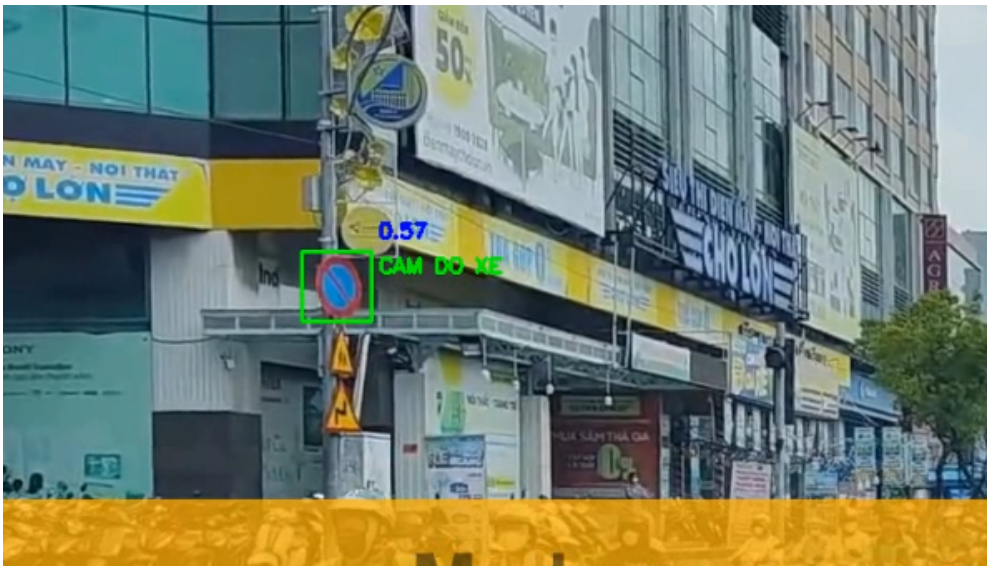

Figure 2.2.4. Frame at 00:55 detected No Parking

5. Frame at 00:59: Successfully detected "No parking”.

Figure 2.2.15. Frame at 00:59 detected No parking

6. Frame at 01:03: Successfully detected "No parking".

Figure 2.2.16. Frame at 01:03 detected No parking

2.3 Output Video Links

The full output videos have been uploaded to Google Drive as required:

- Video 1: Google Drive
- Video 2: Google Drive