Lane Marking Extraction and Segmentation

I used OpenCV to detect the different colours form the Image and by differentiating between colours I can mark boundaries markings.

Approach

1. Image Processing Workflow

The lane detection process follows these key steps:

1. **Load the Image**: Read the input image using OpenCV.
2. **Convert to HSV Color Space**: HSV (Hue, Saturation, Value) is more robust for color-based segmentation than RGB.
3. **Define Color Thresholds**: Define lower and upper HSV bounds to detect white and yellow lane markings.
4. **Create Binary Masks**: Use cv2.inRange() to extract pixels within the defined color ranges.
5. **Highlight Detected Markings**: Apply the masks to the original image to highlight lane markings.
6. **Save the Output Images**: The binary masks and highlighted image are saved for further analysis.

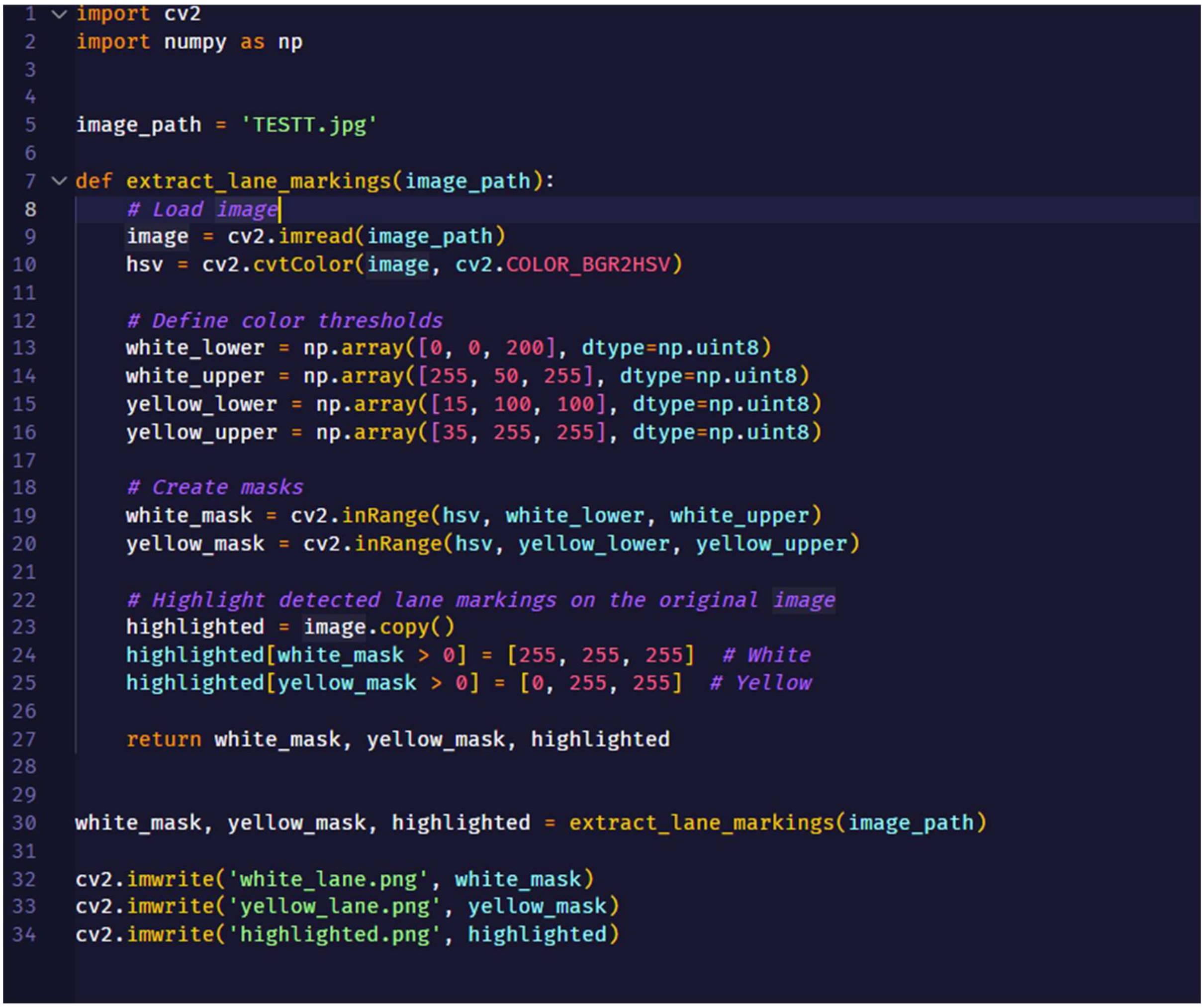
2. Why Use HSV Color Space?

- The Hue channel helps isolate specific colors (yellow and white) more effectively.
- Saturation and Value adjustments make detection more robust under varying lighting conditions.

3. Image Segmentation Using Thresholding

- White lanes typically have high brightness and low saturation.
- Yellow lanes have a distinct hue range that can be isolated.
- Applying cv2.inRange() generates binary masks for easier processing.

Code Implementation



```python
import cv2
import numpy as np


image_path = 'TESTT.jpg'

def extract_lane_markings(image_path):
    # Load image
    image = cv2.imread(image_path)
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Define color thresholds
    white_lower = np.array([0, 0, 200], dtype=np.uint8)
    white_upper = np.array([255, 50, 255], dtype=np.uint8)
    yellow_lower = np.array([15, 100, 100], dtype=np.uint8)
    yellow_upper = np.array([35, 255, 255], dtype=np.uint8)

    # Create masks
    white_mask = cv2.inRange(hsv, white_lower, white_upper)
    yellow_mask = cv2.inRange(hsv, yellow_lower, yellow_upper)

    # Highlight detected lane markings on the original image
    highlighted = image.copy()
    highlighted[white_mask > 0] = [255, 255, 255]  # White
    highlighted[yellow_mask > 0] = [0, 255, 255]  # Yellow

    return white_mask, yellow_mask, highlighted


white_mask, yellow_mask, highlighted = extract_lane_markings(image_path)

cv2.imwrite('white_lane.png', white_mask)
cv2.imwrite('yellow_lane.png', yellow_mask)
cv2.imwrite('highlighted.png', highlighted)
```

Output Explanation

1. white_lane.png - Binary mask of detected white lane markings.
2. yellow_lane.png - Binary mask of detected yellow lane markings.
3. highlighted.png - Original image with detected lanes highlighted in white and yellow.