EXTRACTING DESIRED FEATURES FROM AN IMAGE USING BASIC IMAGE PROCESSING TECHNIQUES

PRATIK KUMAR MAITY

1 Introduction

We are provided with an image of a highway, and the goal is to create a white lane and yellow lane mask and apply it to the image so that we can extract the white and yellow lanes.

2 Solution and Main Ideas

The first step is to convert the image from BGR to HSV format. In BGR, colors can be affected by brightness variations, making it hard to extract specific colors. On the other hand, HSV is better suited for color-based segmentation. For example, a yellow lane in sunlight and shadow has different BGR values but similar HSV values. Thus, HSV helps us extract yellow and white lanes independently of brightness.

3 Ideas to Generate a White Lane Mask

The main idea is to create a binary (black and white) image without affecting the white-colored pixels and setting all other pixels to black. However, a problem with this approach is that clouds in the image will also appear in the mask, which we do not want. To address this, we first create a mask that includes the white clouds and then find a way to remove them.

We define two NumPy arrays: `lower_white` and `upper_white`, which are $[0, 0, 200]$ and $[180, 30, 255]$, respectively. Here, the Hue (H) value ranges from 0 to 180 because white does not have a specific hue, so we set the entire color range. The Saturation (S) value is set from 0 to 30 since white is any hue with very low saturation, and we want to capture pixels close to white but with a slight tint of color. Finally, we set the Value (V) from 200 to 255, as white pixels in the image are mostly bright.

Next, we use the function `cv2.inRange()` with input parameters as the HSV image, `lower_white`, and `upper_white`. This ensures that any pixel whose HSV value falls within this range is set to 255 (pure white), while others are set to black (0). This process creates a white lane mask, including clouds.

To remove the clouds from the mask, we use the method of contours. First, we increase the resolution of the image for better accuracy. Then, we apply binary thresholding, which requires the image to be in grayscale since binary thresholding operates on a single channel. We create a binary image `sky_thresh` using `cv2.threshold()`, where all pixel values greater than 200 are converted to pure white pixels, and the rest remain black. This creates an image that mostly contains white clouds (with some noise). Then, we apply contour detection to extract large white areas and define a new image `sky_mask`. The idea is that white lane markings have a small contour area and thus will not appear in `sky_mask`. Finally, we invert the pixels of `sky_mask` and apply the resulting mask to the main image to extract only the white lanes.

4 Ideas to Generate a Yellow Lane Mask

To create a yellow lane mask, the approach is similar to generating a white lane mask, except that we define `lower_yellow` and `upper_yellow` as $[15, 100, 100]$ and $[180, 30, 255]$, respectively. Additionally, we apply morphological transformations to the mask to reduce noise using a 5×5 kernel of ones.

5 Final Processing

Finally, we combine both masks using `cv2.bitwise_or` and `cv2.bitwise_and`, and then apply this combined mask to the original image to extract the white and yellow lanes.