Catalog

Overview

Lane detection has strong practical significance, mainly reflected in driverless road planning, autonomous driving planning needs to abide by traffic rules. In addition to traffic signals, the reference standard is road lane lines. By detecting lane lines, ground indication signs can be further detected and pre-collision warning strategies can be designed.

Lane Detection has two lane key points, one is elimination of irrelevant information including selection of region of interest and noise filtering and the other is edge detection including canny edge detector and Hough Transformation for straight lines.

Lane detection is a three-step process. The process of lane recognition is shown in Figure 1.

The first step is image preprocessing. The purpose of image pretreatment is to reduce the fuzzy part of the image to enhance the image, remove the noise interference to make the image clear, improve the contrast and highlight the acquisition of the edge layer. For this purpose, we will use interpolation algorithm, histogram equalization, noise filtering, sharpening and other learned knowledge, and new image processing methods will try to supplement in the subsequent study.

The second step is feature extraction. Edge detection methods such as Sobel operator and Canny operator were used to extract edge features as lane markers. We will try different edge enhancement methods and compare their extraction effects for different images, so as to analyze their advantages and disadvantages as well as their

application scope.

The third step is lane marking. We use the road features extracted in the previous step for lane marking.

Figure 1 The process of lane detection

The project also used Gaussian filter to optimize the result.

Image preprocessing

The goal of this step is to make image become what we want to use. First, we convert the origin image to a grayscale image, because it can be processed easily.

Figure 2 Grayscale transformation

As we can see from the two images, the left image is origin image from capturing. The right image is grayscale image after processing.

Then we need to select the region of interest. The lane is what we are interested in. We will tag the lines of the lane in subsequent steps so we select the region previously for preparation.

Figure 3 Selection of ROI

As we can see from the two images, the lane is extracted entirely. We can label the lines in subsequent steps.

Edge Detection

Now came to the edge detection part. There exist many methods of edge detection including Laplacian-based edge detection, Log Filter and Canny edge detection. Due to the good performance on anti-noise of Canny edge detectors, we choose Canny edge detector as our edge detection method.

Principles of Canny Edge Detector

The picture below is the workflow of canny edge detector. The first step is to filter the image with derivative of Gaussian, so that a single pixel noise becomes almost irrelevant on a Gaussian smoothed image.

The next step is to find magnitude and orientation of gradient, which contains most information of edges in an image.

Then suppress all non-maximum values to ensure the edges are single pixel wide

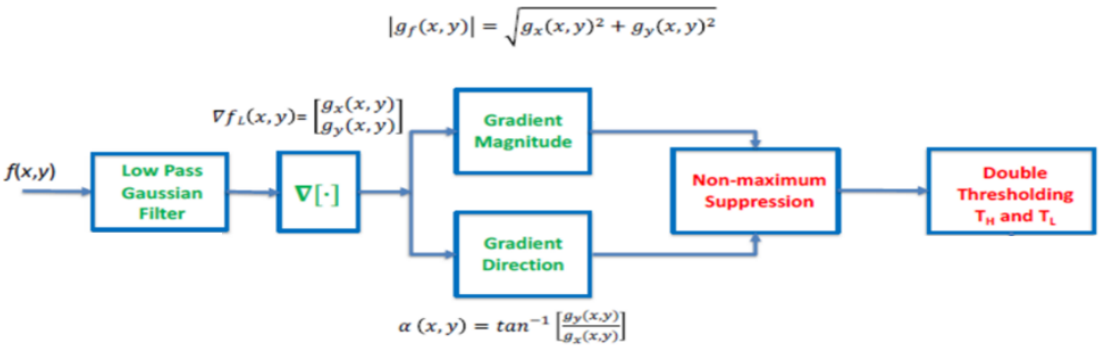

Figure 4 Workflow of Canny detector

by making the blurred boundary sharp, which, in other words, preserves the maximum value of the gradient intensity on each pixel and deletes the rest.

Next is linking and thresholding. After non-maximum suppressing, there are still

many noise points in the image after non-great suppression. Canny algorithm applies a technique called double threshold, that is, setting an upper limit and lower limit of the threshold. If the pixel point in the image is larger than the upper limit of the threshold, it is considered to be bound (called strong edge). If it is smaller than the lower limit of the threshold, it is considered to be not bound and the one between the two is considered to be a candidate (called weak edge), which needs further processing.

To better determine edges, Canny adapted Hysteresis Thresholding. Hysteresis Thresholding requires two thresholds, namely, high threshold and low threshold. By assuming that the important edges in the image are continuous curves, we can track the blurred parts of a given curve and avoid mistaking noisy pixels that do not make up the curve as edges. So, we start with a large threshold that identifies real edges that we are fairly sure of. Using the orientation information we exported earlier, we start with these real edges and track the entire edge in the image. When tracing, we use a small threshold so that we can follow the fuzzy part of the curve until we get back to the starting point.

Once this is done, we have a binary image where each point indicates whether it is an edge point or not.

Results after Canny Edge Detection

Here is our result after canny edge detection. We can see that all the edges of road lanes are clearly shown in the picture on the right with only a little interference from other objects. Basically, we have already got what we want, the road lanes edges. In next part we will further process this image to accurately mark out these lane lines.

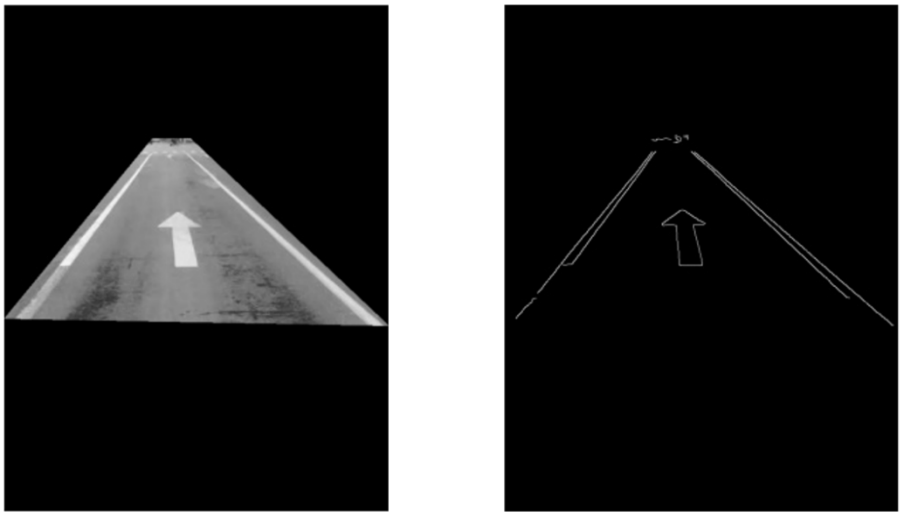

Figure 5 Result after canny edge detection

Lane marking

After the process of image preprocessing and the selection of ROI, we decided to use 'Hough Transform' to complete the 'Lane marking'.

Introduction

The Hough transform is a feature extraction technique which is mostly used in image analysis, computer vision, and digital image processing. Firstly, the Hough transform as it is universally used today was invented by Richard Duda and Peter Hart in 1972. The purpose of this technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform. Basically, the classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses.

Theory

In automated analysis of digital images, a subproblem often arises of detecting simple shapes, such as straight lines, circles or ellipses. Therefore, in many cases an edge detector can be used as a pre-processing stage to obtain image points or image pixels that are on the desired curve in the image space. Due to imperfections in either

the image data or the edge detector, however, there may be missing points or pixels on the desired curves as well as spatial deviations between the ideal line/circle/ellipse and the noisy edge points as they are obtained from the edge detector. For these reasons, it is often non-trivial to group the extracted edge features to an appropriate set of lines, circles or ellipses. The purpose of the Hough transform is to address this problem by making it possible to perform groupings of edge points into object candidates by performing an explicit voting procedure over a set of parameterized image objects.

Detecting Lines

For the case of Hough detecting straight lines, in general, the straight line $y = mx + b$ can be represented as a point (b, m) in the parameter space. However, vertical lines do not satisfy this condition. They give rise to unbounded values of the slope parameter m. Thus, for computational reasons, we need to use another equation

$$r = x\cos\theta + y\sin\theta$$

where r is the distance from the origin to the closest point on the straight line, and θ is the angle between the x axis and the line connecting the origin with that closest point.

The intuition for this form, similarly to the plane equation, is that every vector on the line must be perpendicular to the straight line of length r that comes from the origin. It is easy to see that the intersection points of the function line and the perpendicular line. Therefore, for any point P on the line, since $P = (x,\ y)$ and $r = x\cos\theta + y\sin\theta$, we get the final form of $P = (r\cos\theta,\ r\sin\theta)$.

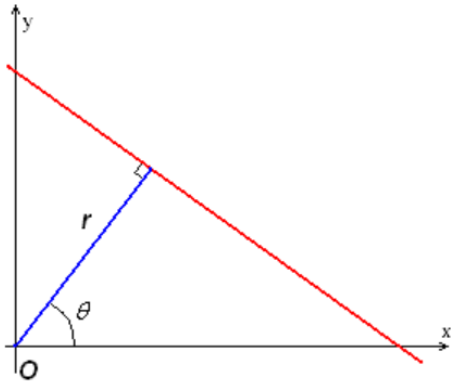

Figure 6 Cartesian coordinates

It is therefore possible to associate with each line of the image a pair (r, θ). The (r, θ) plane is sometimes referred to as Hough space for the set of straight lines in two dimensions. This representation makes the Hough transform conceptually very close to the two-dimensional Radon transform. In fact, the Hough transform is mathematically equivalent to the Radon transform, but the two transformations have different computational interpretations traditionally associated with them.

Given a single point in the plane, then the set of all straight lines going through that point corresponds to a sinusoidal curve in the (r, θ) plane, which is unique to that point. A set of two or more points that form a straight line will produce sinusoids which cross at the (r, θ) for that line. Thus, the problem of detecting collinear points can be converted to the problem of finding concurrent curves.

Implementation

The linear Hough transform algorithm estimates the two parameters that define a straight line. The transform space has two dimensions, and every point in the transform space is used as an accumulator to detect or identify a line described by $r = x \cos\theta + y \sin\theta$. Every point in the detected edges in the image contributes to the accumulators.

The dimension of the accumulator equals the number of unknown parameters, i.e., two, considering quantized values of r and θ in the pair (r, θ). For each pixel at (x, y) and its neighborhood, the Hough transform algorithm determines if there is enough evidence of a straight line at that pixel. If so, it will calculate the parameters (r, θ) of that line, and then look for the accumulator's bin that the parameters fall into and increment the value of that bin. By finding the bins with the highest values, typically by looking for local maxima in the accumulator space, the most likely lines can be extracted, and their geometric definitions read off. The simplest way of finding these peaks is by applying some form of threshold, but other techniques may yield better results in different circumstances – determining which lines are found as well as how many.

The final result of the linear Hough transform is a two-dimensional array (matrix) similar to the accumulator—one dimension of this matrix is the quantized angle θ and the other dimension is the quantized distance r. Each element of the matrix has a value equal to the sum of the points or pixels that are positioned on the line represented by quantized parameters (r, θ). So, the element with the highest value indicates the straight line that is most represented in the input image.

Results

Figure 7 Lane detected after Hough Transform

We can clearly see that after 'Hough Transform', the lane is marked by green lines. Since the lane has width, we detect the two borderlines of each lane.

Besides, obviously, in the left lane, there is a part of the lane darker than the rest part, as a result, that part is missed instead of marking.

This result can be used to perform more applications. For example, we can use it to detect whether a vehicle crosses inhibit line.

Optimization

Selection of smoothing filter

Boxcar filter

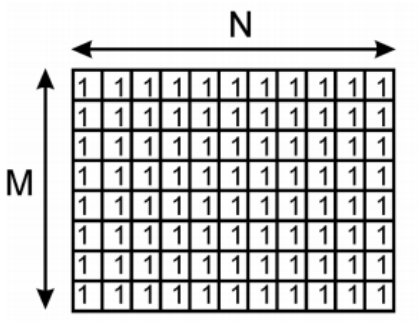

Figure 8 M✕N Boxcar filter

We want to achieve a better smoothing effect by adjusting different parameters.

Figure 9 Kernel size 3 × 3

Figure 10 Kernel size 5× 5

As a result, when the kernel size is 3 × 3, the Boxcar filter has a better effect.

Gaussian filter

Selection of Kernel size:

Different kernel size determines different smoothing effects, so we need to select

suitable kernel size.

$$h(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Figure 11 Kernel size 3 × 3

Figure 12 Kernel size 5 × 5

Figure 13 Kernel size 7 × 7

Figure 14 Kernel size 9 × 9

As a result, we have a better smoothing effect in size 5 × 5.

Selection of σ

Different σ(standard deviation) determines different smoothing effects, so we need to select suitable standard deviation for Gaussian filter.

Figure 15 $\sigma_x = 0.2\ \sigma_y = 0.2$

Figure 16 $\sigma_x = 0.4\ \sigma_y = 0.4$

Figure 17 $\sigma_x = 0.8\ \sigma_y = 0.8$

Figure 18 $\sigma_x = 1.1\ \sigma_y = 1.1$

As we can see from the above pictures, $\sigma_x = 1.1\ \sigma_y = 1.1$ is a better choice.

Finally, we choose Gaussian filter which has a better smoothing effect, the parameters are shown in the table.

Table 1 Final parameters of Gaussian filter

Gaussian filter				
Kernel size	3 × 3	5 × 5	7 × 7	9 × 9
Choice		✓		
Standard deviation	$\sigma_x = \sigma_y = 0.2$	$\sigma_x = \sigma_y = 0.4$	$\sigma_x = \sigma_y = 0.8$	$\sigma_x = \sigma_y = 1.1$
Choice				✓

Summary

In this project, we use some methods such as Gaussian Blur, Canny Edge Detection and Hough Transformation. These applications help us to achieve a better target. Briefly, Gaussian Smoothing Filter decreases noise in the image, ROI Selection obtains the region we are interested in, Canny detects edge of lane lines and Hough transformed marks edges that are detected before. The project also used smoothing filter including Boxcar filter and Gaussian filter to optimize the result.

Appendix

The structure of the code

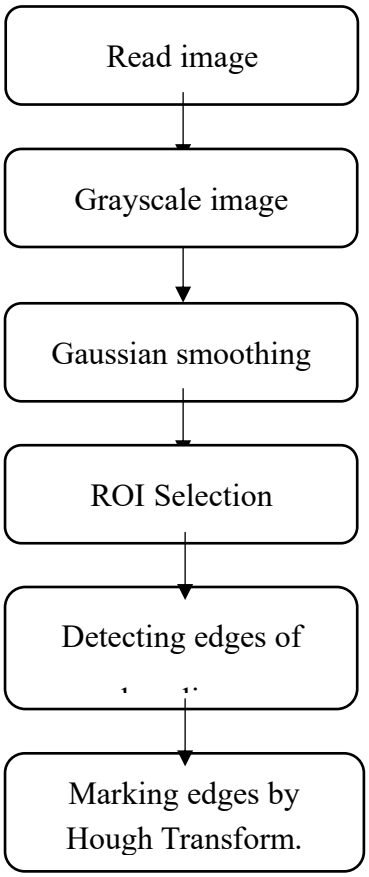

Figure 1 flow chart

Table 2 Function

Function name	Input parameters	Function
region_of_interest()	img: The original image vertices: The coordinates of region of Interest	Extract ROI of the image.
drow_the_lines()	img: Original image lines: The coordinates created by Hough Transform	Mark the edge of the lane.

show_vedio()	None	Mark the edge of the lane in video.