Team 4 : Lane Detection Mobile App using Canny Edge Detection based on the Video-based Source

Chang Chen
Department of Electrical and Computer Engineering
University of Illinois at Chicago
Chicago, U.S.A
cchen269@uic.edu

Yingyi Luo
Department of Electrical and Computer Engineering
University of Illinois at Chicago
Chicago, U.S.A
yluo52@uic.edu

***Abstract*—This report is to use Canny Edge Detection to implement Lane Detection for a mobile app. The detected lanes are overlaid to the original image by Hough Transform. Finally, we analyze the effectiveness of the method and the possibilities for further improvement.**

***Index Terms*—Canny edge detection, Hough transform, Lane detection, Mobile App Development**

I. Introduction

Lane detection and identification is an important part of intelligent vehicle visual navigation research. Its accuracy, real-time, and robustness are directly related to the safety and reliability of intelligent vehicle driving.

Edge information is an important image feature information. Therefore, the study of edge detection methods is a topic of great interest in the field of image analysis and recognition. The traditional edge detection operators such as Sobel, Prewitt, Roberts and Kirch are mostly local window gradient operators. Since they are sensitive to noise, they are not very practical for processing real images. [3] Canny operator is a class of edge detection operator with excellent performance, which is widely used in many fields of image processing. Hence, we decide to perform Canny operator in our lane detection project for a video-based source.

The Hough transform is a reliable method for straight line detection and is widely used in lane line detection. [4] However, the conventional Hough transform-based lane line detection in complex roadway environment suffers from the interference of non-lane line segments in the image, which has the disadvantages of large computation and poor real-time performance. We also use Hough Transform for the efficiency of presenting straight lines in images after performing the canny detection operator. Then we want to realize the real-time lane detection in a mobile app.

II. Method Description

A. Canny Edge Detection

The Canny operator is a multilevel edge detection algorithm developed by Australian computer scientist John F. Canny in 1986 and the goal is to find an optimal edge. The conventional Canny edge detection algorithm is divided into four steps:

- smoothing the image with a two-dimensional Gaussian filter;
- calculating the gradient amplitude and direction by differentiation using a 2×2 template;
- non-maximal suppression of the smoothed image to obtain a single-edge image with accurate localization;
- detecting and connecting the edges by double thresholding.

The conventional Canny algorithm uses three judgment criteria for image detection, namely:

- Signal-to-noise ratio criterion. That is, the true edges must be detected as much as possible.
- Localization accuracy criterion. That is, the distance between the detected edge point and the true edge center must be minimized as much as possible.
- Single-edge response criterion. That is, the algorithm must return only one edge point whenever possible.

B. Hough Transform

Hough transform is a kind of feature detection, which is widely used in image analysis, computer vision and digital image processing. The classical Hough transform detects straight lines in images, but later, the Hough transform can recognize not only straight lines, but also any shapes, such as circles, ellipses, etc. In this way, we can realize the mapping between polar coordinate system and parameter space, and we can see that the points A, B and M in the Cartesian coordinate system are actually lines in the parameter space under the polar coordinate system, while the line ABM in the Cartesian coordinate system is actually the point F in the parameter space under the polar coordinate system, i.e., it has the same duality. And the line parallel to the X or Y axis can be mapped to the parameter space well to ensure that it has intersection.

III. Key Ideas and Simulation Description

In our example, we use Canny edge detection to detect edges and use Hough transform to make the smart phone know the specific location of the lane line, which provides support for subsequent decisions making and other operations. The general processes consist the following steps:

A. Convert Color Image to Gray Scale Image

In this step, we convert the color image to gray scale image, that is, from a three-channel RGB image to a single-

channel image. You can consider this step as a feature extraction operation or a simplification operation. We use the `inputFrame.gray()` function in this step. The method's role is to convert the RGB image into gray scale.

B. Perform Gaussian Blurring

We use a Gaussian blur with a 7×7 kernel to smooth the image. with `Imgproc.GaussianBlur()` method provided by OpenCV.

C. Extract Vertical Edge Features

We use matrix $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ to extract the vertical edge features from the gray scale image. It is because the lanes are mainly made up of vertical features in each image. We want to eliminate the horizontal features and only preserve the useful vertical features. The specific code is as follows.

```
Mat kernel = Mat.zeros(1,3,CvType.CV_32F);
kernel.put(0,0,-1);
kernel.put(0,2,1);
Imgproc.filter2D(edges,edges,-1,kernel);
```

D. Set Up Region of Interest

Region of interest is the area that is crucial to lane detection. In the real world, the lane information normally appears at the lower part of the frame. We cut a half frame width times half frame height rectangle area as the region of interest at the lower center of the frame. We didn't figure out the usage of triangle in OpenCV for Android development. However, we thinks the triangle area is more suitable for lane detection. In simulation, instead of setting up region of interest. We use filters to reduce the unwanted information. For example, Wiener filter algorithm is a common way to filter the gray scale image using a pixel-wise adaptive low-pass Wiener filter. Its method in Matlab is `wiener2`. Its parameters are roughly input image, and neighborhood size specified as a 2-element vector [m n] where m is the number of rows and n is the number of columns. We also use the mask $\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$ and the Matlab method `bwpropfilt` to further reduce the cost of calculation. That is because only certain features of the image are interesting to us to perform further algorithm - Hough transform. Hence, we adopt the mask to extract edges and the method `bwpropfilt` to extract objects from binary image using properties. The method has three parameters, the input image, name of attribute on which to filter, and minimum and maximum property values. These methods can give us a region of interest equivalent effect, which is shown in Figure 1. The simulation code is as follows.

```
% denoising
edged_image = wiener2( ...
thickened_image, [3, 3]);
% Edge detection with mask
edged_image = imfilter(edged_image,...
[-1 0 1]);
edged_image = imfilter(edged_image,...
[-1 0 1]');
% take only largest areas
simplified_image = bwpropfilt(...
edged_image, 'filledarea', 3);
```

E. Perform Canny Edge Detection

The Canny edge detection algorithm is specifically an information extraction algorithm that converts the gray scale image to a binary image with maximum information retention for further operations. We use `Imgproc.Canny()` to extract edges from the processed image. There are 4 parameters in this method, the first is the input matrix, the second is the output matrix, the third and the fourth are the first and second threshold for the hysteresis procedure. The method ignores all edges that are not stronger than the first threshold and accepts the edge that either stronger than the second threshold or stronger than the first threshold and neighbors at least an edge stronger than the second threshold. The simulation result is shown in Figure 1.

F. Perform Hough Transform

Here we use the example from the Matlab document to identify the line features in the prepossessed image by Hough Transform, and finally overlay them to the input image. The simulation is shown in Figure 2. [1] In our app, We use straight line model to fit the lines. The implementation code is as follows. From Figure 3, we can tell that the app runs good when the lane is not curving. We should adopt curving line model to solve this issue. Also, from Figure 4, the detection process can not recognize the lane and other things such as the front cars. Hence, we can further adopt some machine learning algorithm that can recognize things to solve this issue. The specific code to perform Hough Transform is as follows.

```
// store lines in mat format
Mat lines = new Mat();
// starting and ending point of lines
Point p1 = new Point();
Point p2 = new Point();
double a, b;
double x0, y0;
Imgproc.HoughLinesP(edges, lines,
    2.0, Math.PI/180.0, 40, 100, 50);
// then loop through each line
for (int i = 0; i < lines.rows(); i++) {
    // for each line
    double[] vec = lines.get(i, 0);
    double rho = vec[0];
    double theta = vec[1];
    a = Math.cos(theta);
    b = Math.sin(theta);
    x0 = a * rho;
    y0 = b * rho;
```

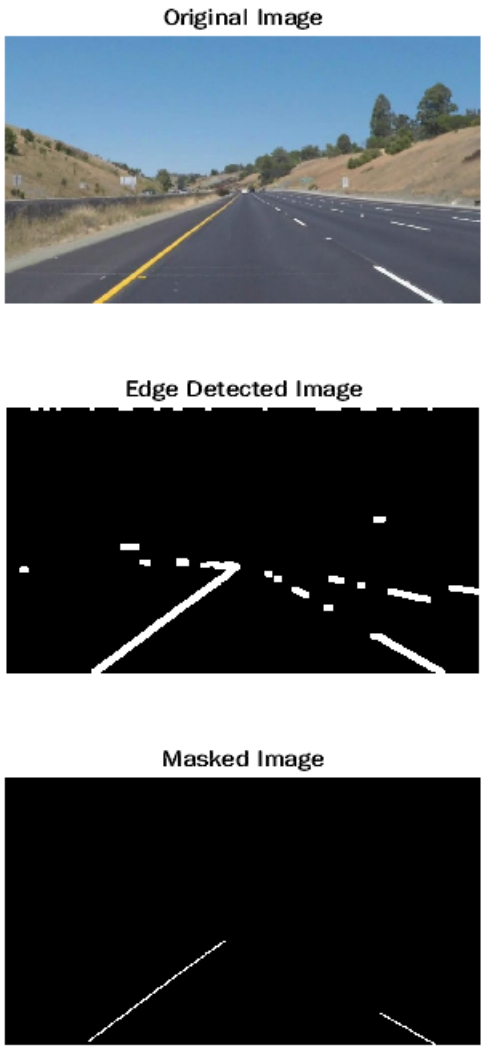

Fig. 1. Original Image, Edge Detected Image, Masked Image.

```
        // starting point and end point
        p1.x = vec[0] + mGrey.width() / 4;
        p1.y = vec[1] + mGrey.height() / 2;
        p2.x = vec[2] + mGrey.width() / 4;
        p2.y = vec[3] + mGrey.height() / 2;
        Imgproc.line(mRgba, p1, p2,
            new Scalar(0, 255.0, 0), 20,
            Imgproc.LINE_AA, 0);
}
```

IV. New knowledge and Challenging parts

Our biggest takeaway from this project was the implementation of the Canny edge detection and Hough transform that we learned in this course. Canny operator is one of the most representative local extramural edge detection operators in edge detection, which has better noise immunity and higher accuracy of edge localization. Experimental results show that the algorithm can preserve more useful edge information and be more robust to noise.

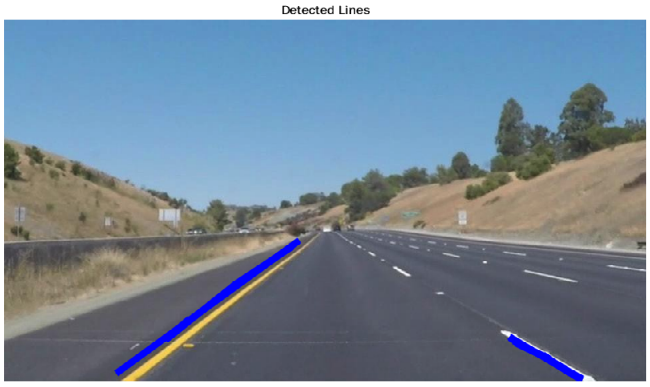

Fig. 2. Detected Lines Simulation on MATLAB.

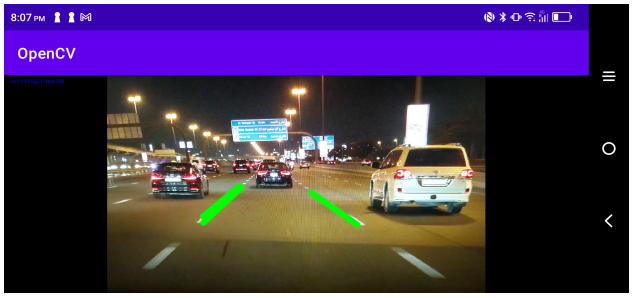

Fig. 3. Detected Lines in Real-time result.

Another new knowledge we gained is that we used new framework in our solution. We tried to deploy our code on a mobile device. This is the most challenging part of the project because there are few tutorials for OpenCV on the Android platform. Hence, we spent tons of time trying to translate the Python and Matlab codes into Java language and find equivalent libraries. However, we intuitively proved the computation-consuming property of Hough Transform which we can barely notice when we simulate on the computer. The application also tells the limitation of this method in real-time lane detection. This gives us hints to seek better solutions for the lane detection task.

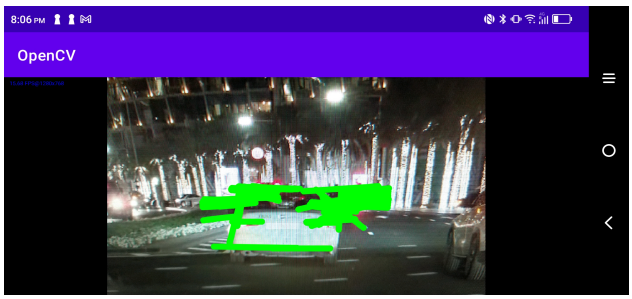

Fig. 4. Front Car detected as lines in Real-time result.

V. Contribution

- Chang Chen: Matlab simulation, Android App Development, Key Ideas and Simulation Description writing
- Yingyi Luo: PowerPoint making, Video editing, Abstract, Introduction, Method Description, New knowledge and Challenging parts writing

References

[1] MathWorks Student Competitions Team (2021). Computer Vision for Student Competitions : All Files (https://github.com/sseshadr/auvsi-cv-all), GitHub. Retrieved December 3, 2021.

[2] L. Hu, Zhao, Shu, "Edge detection and evaluation based on Canny algorithm, " Journal of Heilongjiang Engineering College, vol. 2, 2003.

[3] J. Canny, "A computational approach to edge detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679–698, 1986

[4] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," Commun. ACM, vol. 15, no. 1, p. 11–15, jan 1972. [Online]. Available: https://doi.org/10.1145/361237.361242

[5] S. Wang, H. Lin, Y. Sun, L. Zheng, and H. Zhang, "An improved matching algorithm in the application of robot vision system," in 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA). IEEE, 2017, pp. 89–92.

[6] Y. Li, B. Yen, and Y. Hioka, "Performance evaluation on multi-channel wiener filter based speech enhancement for unmanned aerial vehicles recordings," in INTER-NOISE and NOISE-CON Congress and Conference Proceedings, vol. 263, no. 3. Institute of Noise Control Engineering, 2021, pp. 3584–3594.

[7] S. S. Ganguli, G. K. Nayak, N. Vedanti, and V. Dimri, "A regularized wiener–hopf filter for inverting models with magnetic susceptibility," Geophysical Prospecting, vol. 64, no. 2, pp. 456–468, 2016.

[8] A. Madhukumar, A. Premkumar, and H. Abut, "Wavelet quantization of noisy speech using constrained wiener filtering," in Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No. 97CB36136), vol. 1. IEEE, 1997, pp. 39–43.