A Method to Implementation of Lane Detection Under Android System Based on OpenCV

Xiao-Xu Wei and Lei Meng

Abstract Lane detection has drawn great attention in vehicle assistant driving and intelligent navigation systems, some of them are developed using Android operating system. This chapter presents a method to implementation of lane detection under Android system based on Open Source Computer Vision Library. To resolve the problem between C/C++ and Java mixed-language programming, the native code should be written using JNI with the Android NDK which makes available to use native code languages such as C/C++ in the Android at application level. Implementation of image processed with several processing techniques using OpenCV, such as RGB to grayscale conversion, set region of interest (ROI) crop, Canny edge detection, Hough transform, and linear fitness to detect the road lane. The method is verified on Android Virtual Device at the end of the chapter.

Keywords Lane detection • Android system • OpenCV • Android NDK

1 Introduction

With the development of society, traffic safety is attracting more and more attention and is becoming a key problem in automotive industry nowadays. Vision navigation system of intelligent vehicle is developed to remind drivers to avoid danger or to reduce accidents in advance [1]. Lane detection is becoming one of the most important parts in intelligent vehicle navigation systems, which should not be ignored in independence navigation.

X.-X. Wei (✉)
School of Automation, Wuhan University of Technology, Wuhan 430070, China
e-mail: weixiaoxu123123@163.com

L. Meng
School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China

J. Juang and Y.-C. Huang (eds.), *Intelligent Technologies and Engineering Systems*, Lecture Notes in Electrical Engineering 234, DOI 10.1007/978-1-4614-6747-2_14,

Now some of the vehicle navigation systems are developed on Android operating system. The Android platform has become more and more popular, which is a Linux-based operating system for mobile devices, especially on smart phones and tablet computers. So a method is provided to implement lane detection under Android system based on Open Source Computer Vision Library (OpenCV). Some people have achieved lane detection in Windows system [1]; the method in this chapter will try to achieve lane detection in Android system.

2 Works About Development Environment

Android applications are usually developed in the Java language by using the Android Software Development Kit (SDK) [2].

OpenCV is released under a BSD license and hence it is free for both academic and commercial use. It has C++, C, Python, and Java interfaces and supports Windows, Linux, Android, and Mac OS. If the code is written in optimized C/C++, the library can take advantage of multicore processing [3]. Therefore, the mixed-programming language involves C/C++ and Java direct function call to each other, the native code is written using Java Native Interface (JNI) and compiled through Android NDK.

Java language uses JNI to access codes written in other language (such as C/C++). Android NDK is a toolset which makes available to use native code languages such as C/C++ in the Android at application level [4]. Now most versions of the NDK provide a simplified build system through the new ndk-build build command. It adds support for easy native debugging of generated machine code on production devices through the new ndk-gdb command [5]. In addition, Cygwin is a collection of indispensable tools which provide a Linux look and feel environment for Windows. Android NDK can be used in Windows after installing Cygwin.

3 Implementation on Android Platform

To resolve the problem between C/C++ and Java mixed-language programming, the native code should be written by using JNI. Therefore, there are two parts to complete the implementation: firstly, algorithm of lane detection is written in C++ language by using JNI and then compiled by Android NDK to generate shared libraries that can be called through Java code; secondly, according to the Android application framework, Java code is written for showing processing results [6]. Figure 1 shows the processing of implementation.

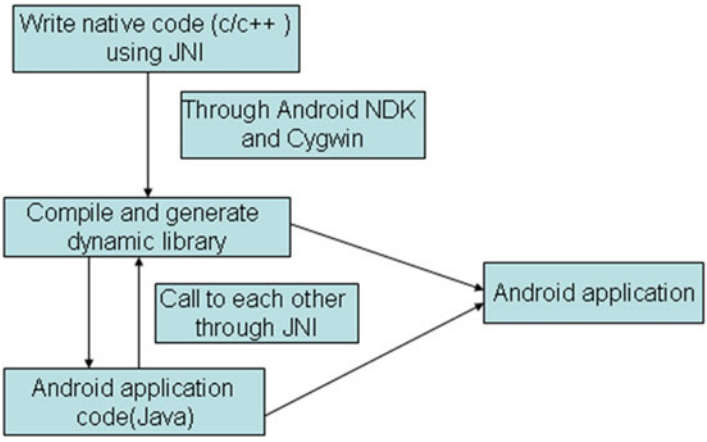

Fig. 1 Implementation process of the method

3.1 Algorithm of Lane Detection

The image processing algorithm of lane detection is implemented by OpenCV. There are several steps in this algorithm.

Load Image and Grayscale. This algorithm starts with loading image through JNI which is usually a path of temporary image. When the color image is loaded, it is converted to grayscale by a function in OpenCV. Grayscale gives the value of every pixel separate sample [7]. Grayscale image is base of edge detection.

Set the Region of Interest. In order to reduce redundant image data quantity based on effective information, the input image should be set on the region of interest (ROI). The ROI is set depending on specific conditions which are not hard absolute. In this chapter, the ROI is 2/3 of the image from the bottom, the remaining 1/3 of the image contains some useless elements (such as cars, buildings, and the sky). Setting the ROI is using the function of cvSetImageROI () in OpenCV [1]. After that, the processing algorithm will only deal with the ROI of the image, and the processing speed is also increased.

Edge Detection. Edge detection is an important technology in the preprocessing procedure. The main consideration in this algorithm is to save full information without missing any edges, so Canny operator is adopted to detect edges [7]. CvCanny () function in OpenCV library is used to implement edge detection, but the thresholds in the function are fixed experiential. To adapt the changing environment, Canny method adopts a self-adapting edge detection based on Otsu method to replace the two experiential thresholds in CvCanny function. Hough transform is adopted to gain the line of lane in edge space because of the character of traffic lane.

Hough Transform. At present, a common method to detect lines is Hough transform. In this part, cvHoughLines2 () function which is one of the image transform functions in OpenCV library is used to complete Hough transform on the image.

The function prototype is CvSeq * cvHoughLines2 (CvArr* image, void *line_storage, int method, double rho, double theta, int threshold, double param1 = 0, double param2 = 0). The detailed description of parameters is follows [8]:

Image: The input 8-bit single-channel binary image.
LineStorage: The storage for the lines detected.
Method: The Hough transform variant, there are CV_HOUGH_STANDARD of stander Hough transform (SHT), CV_HOUGH_PROBABILISTIC of progressive probabilistic Hough transform (PPHT), CV_HOUGH_MULTI_SCALE of multiscale variant of SHT [1].
Rho and Theta: The resolution desired for the lines.
Threshold: The value in the accumulator plane that can reach the routine to return a line.
Param1 and Param2: The parameters associated with the method of Hough transform. The parameters are not used because we use the SHT in algorithm of lane detection.

Lane Boundary Detection. This part tells how to realize lane boundary detection for middle lane. At first, we set the region of interest where middle lane is located as shown in Fig. 2a.

Then building x-0-y coordinates, respectively return much value of Y corresponding color to red point coordinates. In the actual operation, we find that the effective value point of R channel is in a range. If the three channel values of RGB image are, respectively: P[B], P[G] and P[R], the returning effective red pixels value should be satisfied condition:

$$90 < \text{p[R]} <= 255 \quad \&\& \quad \text{p[G]}! = 255 \quad \&\& \quad \text{p[B]}! = 255.$$

Average of multiple returned values is the X axis of central point, then linear fitting with least-square method. So the boundary of the liner fitting result is shown as Fig. 2b (green line is boundary).

The boundary is calculated with coordinates in ROI by the above method, and therefore it needs conversion of coordinate to fit the whole image. Slope of this linear fitting equation does not change, just changing coordinates according to coordinates of ROI movement.

The lane is shown in the initial image through each pair of points on the line. As is described in Fig. 3, the coordinate of point A, B, C, and D can be calculated according to the green line which was achieved in Fig. 2b. Then left and right lane boundary can be detected in the same way.

3.2 Compile Shared Library

At first, function declaration corresponding C/C++ should be established; the subject of the declaration is jstring Java_com_haveimgfun_LibImgFun_ImgFun

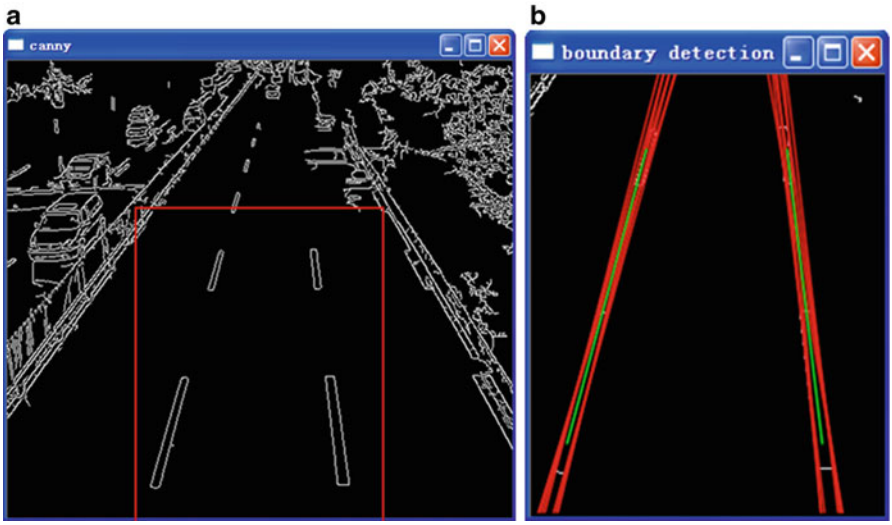

Fig. 2 (**a**) Set the region of interest. (**b**) Result of liner fitting

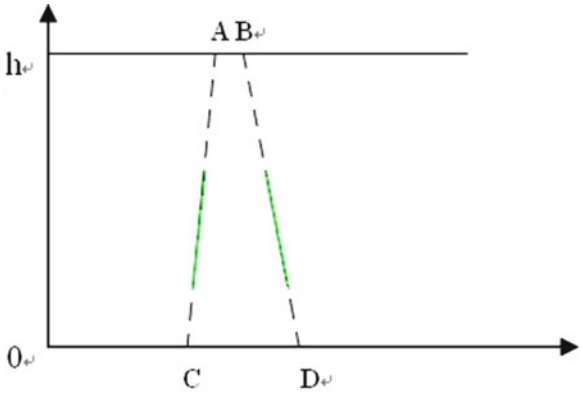

Fig. 3 Lane shown in the initial image

(JNIEnv* env, jobject thiz, jstring path, jstring path1), env points to environment of JAVA, and this represents Java class instance included in this native method. The next two parameters is the path of image. Then, a header file contained the function declaration can be generated through the tool of Java provided by Android JDK. The generated header file is copied to a folder named jni created under project directory. The C++ source files introduced that header file are created to write functions showing lane detection at the same time [6].

To compile shared library, two script files are needed to be written, which are Android.mk and Application.mk. The file of Android.mk describes source files that

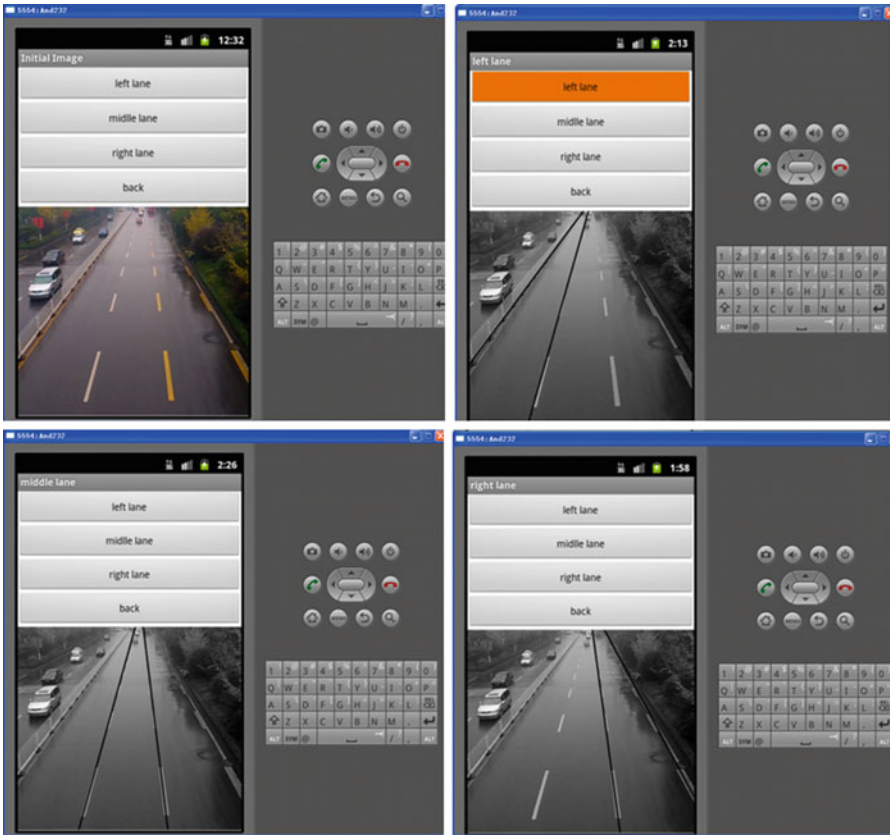

Fig. 4 Result of this method running on the AVD, *top right* is the initial image

need to be compiled through NDK and components formed. The file of Application.mk describes some additional information to help compile about this application. The last step is to open Cygwin, enter into the folder called jni, and execute command of "$NDK/ndk-build." After that, a shared library libxxx.so will be generated finally so that the native method can be called in Java code for Android applications [6].

4 Verify on Android AVD

To verify the effectiveness and feasibility of the method in this chapter, an Android application was created on Android Virtual Device (AVD). An AVD is a device configuration for the Android emulator that allows to model different configurations of Android-powered devices.

Design code user program and UI according to the programming standard of Android SDK, and then package it as a file named xxx.apk, install file on the AVD [6]. When the AVD is loaded, a virtual SD card function should be created. It needed certain capacity to store the images. The results are shown in Fig. 4 after running the application.

5 Conclusion and Future Work

In this work, lane detection is implemented under Android platform based on OpenCV. The application in Java code can easily call the algorithm written in C/C++ code through Android NDK and JNI. In the future you can try to use this method in real-time lane detection. And we also can improve the detection algorithm to increase accuracy and reduce run time. Maybe your phone can be a navigation system of intelligent vehicle someday.

References

1. Ye W, Yuetian S, Yunhe X, Shu W, Yuchen Z (2010) The implementation of lane detective based on OpenCV. In: 2010 second WRI global congress on intelligent systems (GCIS). Wuhan, 2010, pp 278–281
2. http://developer.android.com/reference
3. OPENCV, http://opencv.org/
4. Paik JH, Seo SC, Kim Y, Lee HJ, Jung H-C, Le DH (2011) An efficient implementation of block cipher in android platform. In: 2011 5th FTRA international conference on multimedia and ubiquitous engineering (MUE), Crete, 2011, pp 173–176
5. Lee S, Jeon JW (2010) Evaluating performance of android platform using native C for embedded systems. In: 2010 international conference on control automation and systems (ICCAS), Gyeonggi-do, 2010, pp 1160–1163
6. Hongbin W, Denao L, Jumin Z, Yuhu X (2011) Implementation of human face detection based on OpenCV in android system. In: Computer engineering & software
7. Sankaraiah S, Deepthi RS (2011) Highly optimized OpenCV based cell phone. In: 2011 I.E. conference on sustainable utilization and development in engineering and technology, The University of Nottingham, Malaysia Campus, 2011, pp 47–52
8. http://www.emgu.com/wiki/