Vision-Based Vehicle Detection based on Embedded Systems

Ying-Che Kuo, Yen-Feng Li

Department of Electrical Engineering, National Chin-Yi University of Technology
35, Lane 215, Sec.1, Chung-Shan Rd., Taiping, Taichung, Taiwan. 411, R.O.C
kuoyc@ncut.edu.tw

Institute of Electronic Engineering
35, Lane 215, Sec.1, Chung-Shan Rd., Taiping, Taichung, Taiwan. 411, R.O.C
takolyf@gmail.com

***Abstract*— Over the last decade have seen growing importance placed on research in vehicle detection. While implementing the vehicle detecting system, the space of the car and the cost of the hardware platform are the decisive consideration. In this paper, we proposed a modified vehicle detection algorithm to make it can be implemented on the embedded platform. The embedded platform is based on an INTEL XScale PXA270 SoC and limited hardware resources. An application program realizes the proposed algorithm and integrates V4L2 and MiniGUI open source codes under LINUX operating system. The proposed algorithm verifies the preceding vehicles driving on the highway at daytime. In our experiments, the system can correctly verify and track vehicles. And the processing time of this system is less than the human reaction time.**

***Keywords*— Vision-based Vehicle Detection, Embedded System.**

1. INTRODUCTION

In recent years, there are many researches studied in the preceding vehicle detection of vision-based driving assistance system. The vehicle detection in an image has two classes in general. One class uses the fixed camera to take the picture at a fixed place. The objects (vehicles) are detected by using the last image frame to deduct the present one. The applications of this class include car license recognition and intelligent transportation system (ITS). The other class is a car-mounted image recognition system. The image is captured by the car-mounted camera. The scenery in front of the windshield of a car is captured in an image. This system detects the preceding vehicles by extracting many features of a vehicle from the captured image. Our study is classified to this type and mostly uses on the driving assistance system.

Extraction of the vehicle characteristic is a very important part in driving assistance system. Sun et al. [1] proposed vehicle feature extraction and classification method which using Gabor filter and support vector machine (SVM). In the proposed vehicle extraction algorithm, he used genetic algorithms (GAs) to optimize filter banks, and used clustering to find the filters that feature parameters are similar, and deleted redundant filter.

To detect the preceding vehicle, there are many methods developed to extract the boundary lines of the vehicle. In [2], they developed a symmetry detection method to extract the boundary lines of the preceding vehicle. A weighted sum of the symmetry map was computed and the symmetry axis was obtained by the analysis of horizontal and vertical edges to verify the preceding vehicles. Sun et al. [3] proposed multi-scale driven hypothesis to extract the boundary lines of the preceding vehicle. They computed the horizontal and vertical profiles of the edge images in different resolutions. And they performed some analysis to find the highest peaks to extract the preceding vehicle.

In [4] their detected vehicles using three features: underneath (footprint), vertical edge (profile), and symmetry property. They utilized the Sobel edge operation to extract underneath location and vertical edge profile. Based on the symmetry property, they defined a window whose size was according to the typical aspect ratio of vehicles. Then they carried out the symmetry operation and found the greatest symmetry value in the area of window.

Betke et al. [5] utilized image binarization with a threshold value to remove most noises of gray level image. After image binarization, the high

light pixels were including light sources such as the street lamp and car light. Then they found the equidistant light source in pairs of movement. Those light sources in pairs were vehicles.

Most vehicle detection researches utilize PC-based framework to implement this system. The drawback of such system is it needs a lot of computing resources while processing the image. In this paper, we propose some modifications of the vehicle detection algorithm for daytime to make this system can be developed in an embedded platform.

2. VEHICLE DETECTION ALGORITHM

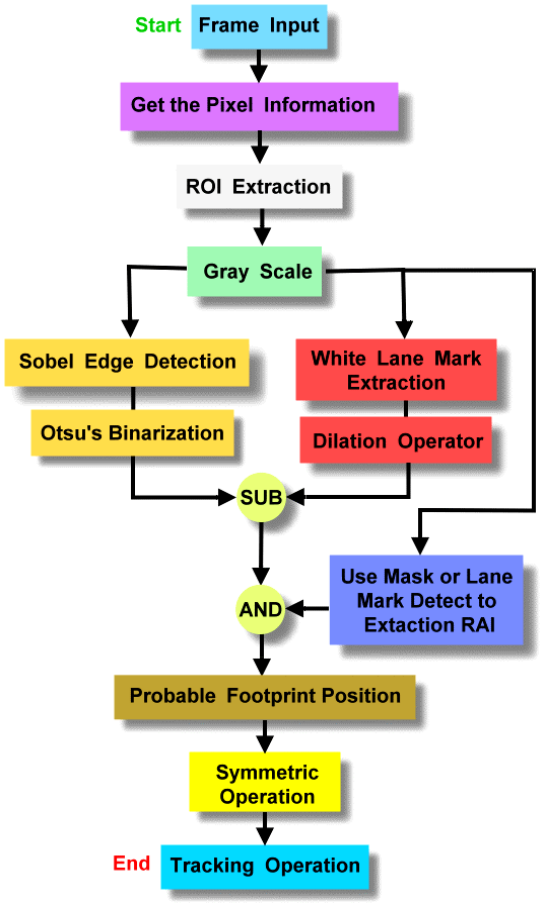

Fig. 1 The flow diagram of the vehicle detection algorithm

In the proposed vision-based vehicle detection system, an image is captured using a video camera settled in a car. The camera is moving with the car and the captured image may have several preceding vehicles on the same image. Recognizing objects with moving camera is much more challenging than doing with a stationary camera. Therefore, the foreground (the preceding vehicles) needs to be distinguished from the captured image. The procedures of our proposed detection algorithm are shown in Fig. 1 and mentions as follows.

2.1 Region of Interest

The image captured in front of the car usually includes the sky, road and wayside scenery. Some areas of the image are the unnecessary information for detection. For examples, the vehicle can not appear on the area of the sky and the areas that close to our car body are unnecessary too. So, we can designate a particular area on the image where the vehicle might appear. This area is commonly called ROI (Region of Interest).

While recognizing the proceeding vehicles on an image, we assume the shooting angle of the camera is parallel to the sky line. Therefore, the sky line will locate in the centre of the image in the vertical direction in general. And we define the sky line as Y_t (the top boundary of ROI). The bottom boundary of ROI, named it Y_b, is demarcated in ten meters in front of the car.

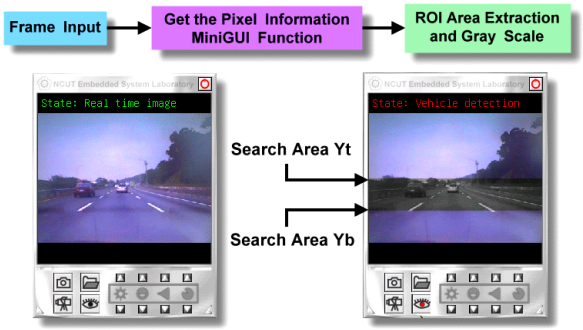

Fig. 2 ROI area and gray level operation

Consequently, the top and bottom boundaries of ROI can all be labeled and shown in Fig.2. Next, the color image in ROI area will be transformed from RGB color space into JPEG YC_bC_r color space. Equation (1) [6] mentions this transformation.

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ Cb &= -0.1687R - 0.3313G + 0.5B + 128 \\ Cr &= 0.5R - 0.4187G - 0.0813B + 128 \end{aligned} \quad (1)$$

To simplify calculations and make it can be implemented in an embedded platform with limited resource, we only use the luminance component (Y) of the image for subsequent

image processes. The result of the gray level image is shown in Fig.2.

2.2 Profile and Characteristic of the Vehicle Extraction

In the next process, we use Soble edge detection [7] to strengthen the profiles of all objects in the ROI area. A 3x3 Soble filtering operation is described in Fig. 3 and the operation formula is mentioned in (2).

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t)\, f(x + s, y + t) \qquad (2)$$

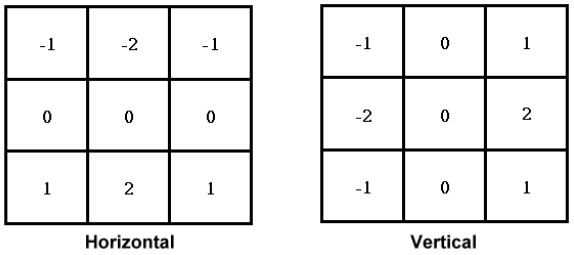

Fig. 3 A 3x3 Sobel filtering

After Sobel edge detection, profiles of objects have already been strengthened. Then we process the gray level image to be a binary level image by binarization method with the Otsu's threshold selection [8].

Otsu's threshold selection is one of binarization method and adopted in this paper. Otsu's threshold method counts each gray level occurrence number in the image. At the beginning, divide gray level of 0 to 255 into two classes, and then calculates the appearance probability of the two classes in the image. The basic formula of probability is mentioned in (3).

$$P_i = \frac{n_i}{N}, \quad P_i \geq 0, \quad \sum_{i=1}^{L} P_i = 1 \qquad (3)$$

L represents the number of gray levels, and N represents the number of pixels. C_0 and C_1 represent foreground and background of the two classes respectively. ω_0 and ω_1 are the sums of the probability of C_0 and C_1. μ_0 and μ_1 are the sums of the average of C_0 and C_1. The formulas of ω_0, ω_1, μ_0 and μ_1 are mentioned in (4~7).

$$\omega_0 = \Pr(C_0) = \sum_{i=1}^{k} P_i = \omega(k) \qquad (4)$$

$$\omega_1 = \Pr(C_1) = \sum_{i=k+1}^{L} P_i = 1 - \omega(k) \qquad (5)$$

$$\mu_0 = \sum_{i=1}^{k} i \Pr(i \mid C_0) = \sum_{i=1}^{k} \frac{i\, P_i}{\omega_0} = \frac{\mu(k)}{\omega(k)} \qquad (6)$$

$$\mu_1 = \sum_{i=k+1}^{L} i \Pr(i \mid C_1) = \sum_{i=k+1}^{L} \frac{i\, P_i}{\omega_1} = \frac{\mu_T - \mu(k)}{1 - \omega(k)} \qquad (7).$$

And then, calculate the variances σ_0^2 and σ_1^2 of C_0 and C_1 respectively as follows.

$$\sigma_0^2 = \sum_{i=1}^{k} (i - \mu_0)^2 \Pr(i \mid C_0) = \sum_{i=1}^{k} (i - \mu_0)^2 \frac{P_i}{\omega_0} \qquad (8)$$

$$\sigma_1^2 = \sum_{i=k+1}^{L} (i - \mu_1)^2 \Pr(i \mid C_1) = \sum_{i=k+1}^{L} (i - \mu_1)^2 \frac{P_i}{\omega_1} \qquad (9)$$

Finally, σ_w^2 is the multiplication of variances and probabilities of the two classes and indicated in (10). We find the minimum value of σ_w^2 with different values k (range from 0 to L). The value k with the minimum value σ_w^2 is thought to be the Otsu's threshold value.

$$\sigma_w^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_1^2 \qquad (10)$$

Once the Otsu's threshold value is got, all the pixels in the image are segmented to one of the two binary levels (0 or 255). The gray level of pixel is set to 0 if it's the primary gray level is less than the threshold vale. On the contrary, set to 255 if gray level is larger than the threshold value. Fig. 4 is an example of the histogram

statistic of the gray levels in an image. Fig. 5 has shown the result of the Sobel edge detection and Otsu's threshold selection processing.

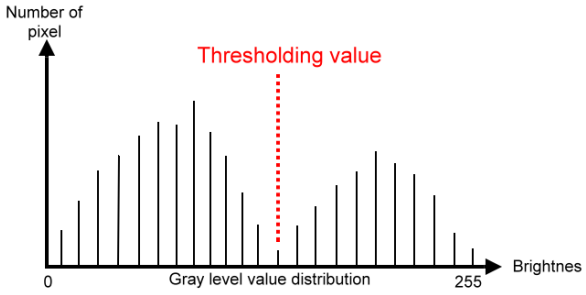

Fig.4 The histogram of gray level

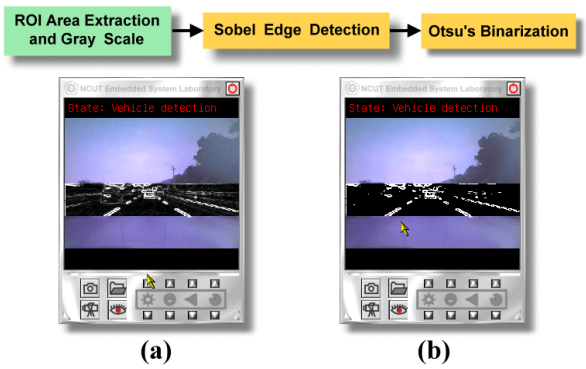

Fig. 5 (a) the result of Soble edge detection, (b) the result of Otsu's threshold selection method

2.3 Removal of the Lane Mark

There are some noises in ROI area of the captured image we don't want. It includes lane mark and non-road scenery in the image. Lane mark detection will be mentioned in this section, and non-road scenery deletion introduced in the next section. Since lane mark has the higher gray level pixels comparing with that of the road. At first, we take the gray levels of five pixels of road at five different areas randomly. The average of all of the five gray levels is the threshold for dichotomizing all pixels in ROI area. The method of binarization is to set pixel that has gray level greater than the threshold as 255, set as 0 on the contrary. Then we make dilation to the binarized image. Fig. 6 is the result after binarization and dilation processing.

In the next procedure, the lane mark is removed through the following SUB operation.

```
if (E(x, y) == 255 && W(x, y) == 0)
    NLM(x, y) = 255;
else
    NLM(x, y) = 0;
```

E(x, y) indicates the pixel in the image that has processed by Sobel edge detection and Otsu's threshold selection method (shown in Fig. 5(b)). W(x, y) indicates the pixel in the image that has processed by binarization and dilation (shown in Fig. 6(b)). Fig. 7 shows the result of this SUB operation and the removal of the lane mark.

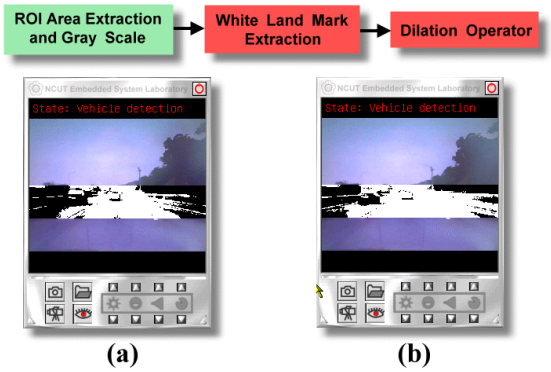

Fig. 6 (a) the result of binarization, (b) the result of dilation

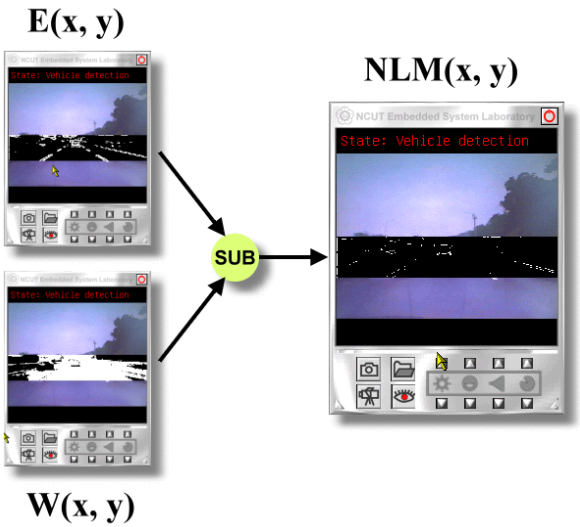

Fig. 7 Result of removal of lane mark

2.4 Lane Mark Detection and RAI Area Produced

The image with no lane mark is obtained in the previous section. But the image still includes non-road noises in ROI area. Non-road scenery normally appeared in the yellow-colored area in Fig. 8.

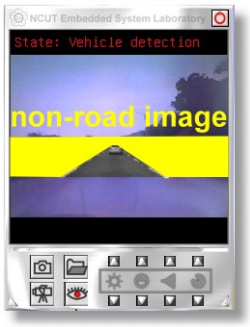

Fig. 8 Non-road area indicated by yellow color

Fig. 9 describes the RAI area which is the area excepted non-road area in ROI. The RAI area is called Road Area Image and marked by green color in Fig. 9.

Fig. 9 RAI area indicated by green color

To verify the RAI area, the lane mark needs to be detected in advance. The procedures of lane mark detection are described as follows and shown in Fig. 10.

1) The start point of this detection is at the center in horizontal direction and Y_b (the bottom of ROI) in vertical direction.
2) Find the first bright pixel form the start point toward both left and right sides in horizontal direction.
3) Once the first bright pixels are found both in left and right sides, recording the positions of the found pixels.
4) If the bright pixel can not be found while the horizontal position has already surmounted last position that found in last procedure, using one pixel inward of last procedure in horizontal direction as this record.
5) Move the start point to the center in horizontal direction and one pixel position toward Y_t (the top of ROI) in vertical direction. Repeat step 2 to 4 until Y_t is reached in vertical direction.

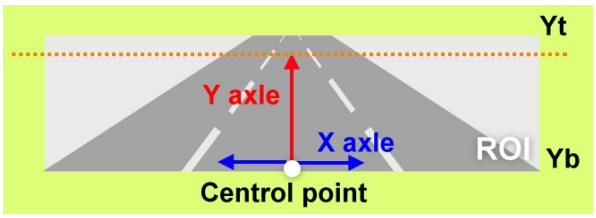

Fig. 10 Explanation of lane mark detection method

Fig. 11 RAI mask of ROI

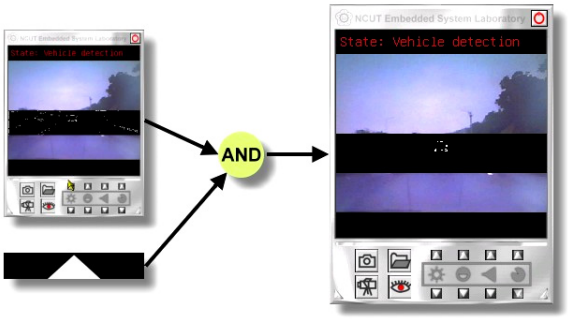

Fig. 12 RAI generation

The RAI area is between two parallel lane marks, so we set all the pixels of RAI area with gray level 255. The area outside of the RAI area is no needs for vehicle detection, so set all the pixels in this exclusive area with gray level 0. Fig. 11 shows RAI mask that produced at lane mark detection procedure. The non-road scenery in the image will be removed by utilizing an AND operation with no lane mark image and RAI mask. Finally, only the scenery inside the RAI area left. An example of this operation is shown in Fig. 12. After these procedures, we obtained the image that may include the vehicle footprint.

2.5 The Vehicle Footprint

In this paper, we focus on the preceding vehicle detection. The preceding vehicle driving at the same lane normally will appear in the RAI area of the captured image. The purpose of Sobel edge detection is to strengthen the profile of the vehicles. And we can find out the bottom of

vehicle keeps very strong orthogonal information, shown in Fig. 13, after Sobel edge detection and binarization have been processed.

Fig. 13 Orthogonal information in the image processed by Sobel edge detection and binarization

In this section, we try to find the footprint of the preceding vehicle in RAI area. The so-called footprint is the image information on the vehicle bottom. We can find the width of the vehicle with the localization of the footprint, and define the height of the vehicle with the width. First of all, we calculate the number of pixels with gray level 255 in column order. Such a method can get a histogram, the positions of the two highest peaks among the histogram are the left and right boundaries of the proceeding vehicle. Two vertical lines pass through these positions are the left (V_L) and right (V_R) boundaries of the preceding vehicle and are shown in Fig. 14. The same procedure is performed in row order. The supreme peak in the histogram is found. And a horizontal line passing through the supreme peak position indicates the bottom of the vehicle (V_B). Then the footprint of the preceding vehicle is been designated.

After labeling the left, right, and bottom boundaries of the vehicle, we define the height of the vehicle is 0.75 times of its width. And then the profile of the preceding vehicle is demarcated.

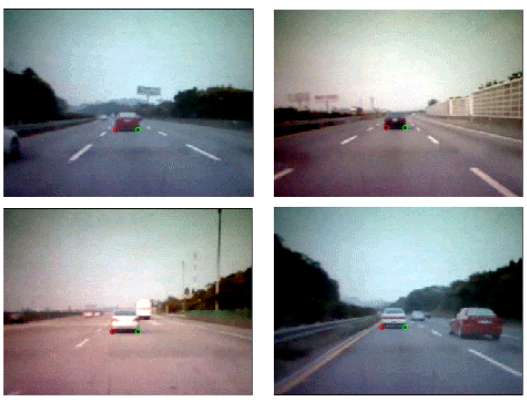

Fig. 14 Result of the footprint labeling

2.6 Verification of the Vehicle

The footprint that we found may not be a vehicle. The symmetry of a vehicle usually is used to verify the found footprint [4] in the researches of vehicle detection. It utilizes the symmetrical characteristic of a vehicle to find the centre axle of the vehicle. The formula of symmetry operation is mentioned following.

$$S(j) = \sum_{i=Ybc}^{Ybc+H} \sum_{\Delta x=1}^{W/2} \sum_{j=X_l-\Delta k}^{X_r+\Delta k} |Pc(j+\Delta x,\ i) - Pc(j-\Delta x,\ i)|$$

$$j_{sym} = \arg\min_{j} S(j) \text{ and } \min S(j) < S_{th} \qquad (11)$$

$S(j)$ is symmetry measure with the symmetry axis located at $j = x$. X_l and X_r are the left and right searching boundaries of the footprint. Y_{bc} is the bottom searching boundary of the footprint. We Let width $W = |X_r - X_l|$ and Height $H = 0.9W$. The result after symmetry operation shows in Fig. 15. If the symmetrical axis is in the area of centre of the vehicle footprint, then prove whether image area is a vehicle.

Fig. 15 Result of symmetry operation

2.7 Vehicle Tracking

In the preceding sections, we have already labeled the vehicle in the image. But, if every image frame needs such heavy and complicated image processing, it will cause the burden of system certainly. To make the system can be implemented in an embedded system, vehicle tracking is adopted to reduce computing efforts.

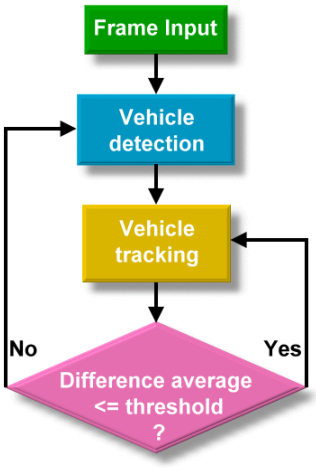

Fig. 16 Relation between vehicle detection and tracking

In this thesis, we utilize the area difference average in vehicle tracking method. At first, store vehicle image area into memory space after vehicle detection. Then the image area of same location in next frame deducts to previous image area, and calculates absolute value of difference and average value. Finally, check whether the difference average is less than and equal to the threshold value. If the difference average greater than the threshold value, we carry on the vehicle detection in the next frame. On the contrary, we continue using the vehicle tracking. The relation between vehicle tracking and vehicle detection can be expressed with Fig. 16.

3. SYSTEM IMPLEMENTATION

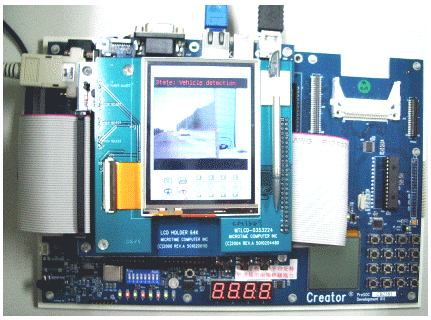

Fig. 17 Embedded system development platform

3.1 System Overview

There are three parts of our implementation for this system. It includes "hardware platform", "system program and device driver" and "user application program". The "hardware platform" includes a INTEL XScale PXA270 SoC, 64M bytes SDRAM, 32M bytes flash ROM, and a few peripheral devices (e.g., TFT-LCD module, Ethernet, AC97, UART, etc) in the hardware platform (see Fig. 17).

TABLE 1

EXPLANATION OF LIBRARIES

Name Of Libraries	ANSI C	V4L2	MiniGUI
Explanation	Library of standard C language.	Access video-information bit stream of webcam.	Build and construct the GUI interface.

The "system program" of the second part of in our implementation includes boot loader, Embedded Linux and cross compiler for system program. The "device driver" can be divided into two blocks. The first block is driver module of LCD, it embeds in driver module group of Embedded Linux, and we only need to start-up it before cross compile. The second block is an image capture driver; it is the UVC (USB Video Camera driver). The UVC driver is a webcam driver of LINUX. It make the user can use webcam on LINUX platform by UVC driver.

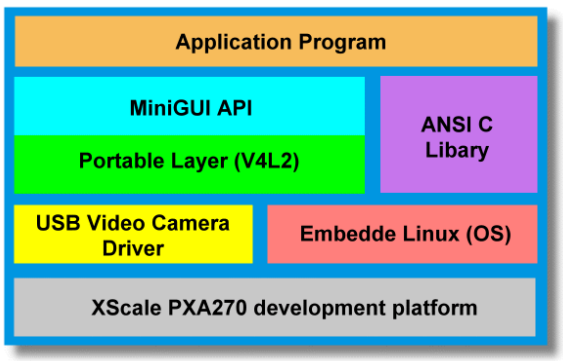

Fig. 18 the software structure of the system

"User application program" includes two parts of library and application program. Library includes image capture library, GUI (Graph User Interface) program library and ANSI C library three components. Table 1 shows the specific functions of these three components.

Application program was written by C language and executed under LINUX operation system. The functions of the designed application

program include image capture, vehicle image process and GUI procedure. Fig. 18 shows the software structure of our system.

3.2 Software Development

There are two essential steps to building and constructing our system except the selection of the hardware platform and operating system at the beginning.

1) Image Capture in Embedded Linux: The V4L2 [9] is an API for executing the image capture function under LINUX. It only needs hardware driver to offer input and output function (ioctl) and makes the image capture programming easily. Fig. 19 shows the operation procedures of V4L2. We can use timer function of MiniGUI to set up capture frame quantity in every second.

2) Building GUI Framework: At present time, most embedded systems have a touch screen user interface. We choose MiniGUI [10] open source code to implement the graphic user interface (GUI) for plentiful user operating interface. The reason is not that MiniGUI has special fortes but its code size is smaller than general GUI open source code. The code size is one of the decisive conditions while we implement an embedded system. Besides, the system designer must also consider the question in many aspects such as the cost, hardware resource of the system and user's habit, etc.

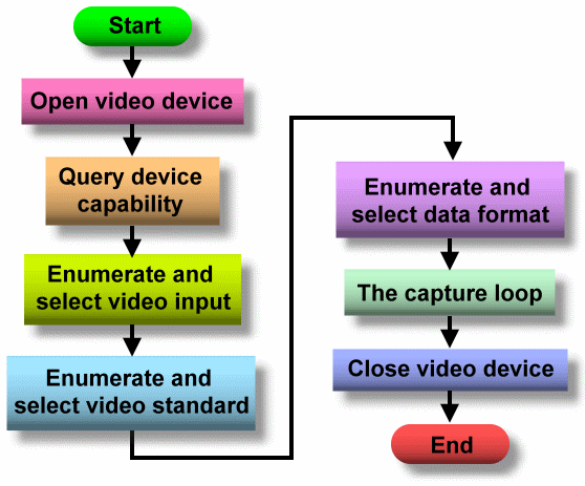

Fig. 19 The operation procedures of V4L2

3.3 Condition and Restriction

Our proposed system is only applied for driving on the highway at daytime. When the speed of a vehicle equals to 100 kilometers per hours, Fig. 20 shows the safe distance is 112 meters [11].

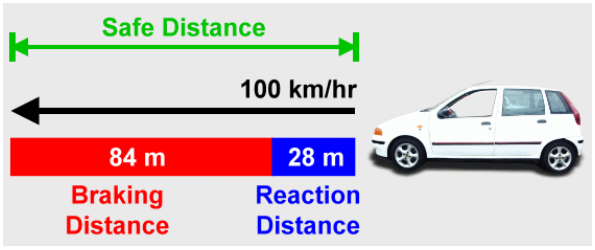

Fig. 20 The vehicle safe distance at 100 kilometers per hour

When the speed of a vehicle is reaching to 100 kilometers per hour, the vehicle moves 27.778 meters forward every second. On the emergent occasions, the driver has only 0.828 seconds to react and step on the brake pedal to stop. So our system must runs within 0.828 seconds through the image capture to sending the result warning message. We use timer function of LINUX to obtain our system processing time and shown in Table 2. In our experiments, our system only takes less than 0.4 seconds in each image processing.

TABLE 2

SYSTEM PROCESS TIME

Range of Sample Result	Image Capture Time	Algorithm Process Time
Maximum Time	192267 (μ sec)	299645 (μ sec)
Minimum Time	117263 (μ sec)	218862 (μ sec)
Average Time	136498 (μ sec)	244820 (μ sec)

4. EXPERIMENTATIONS AND RESULTS

In our experiments, for reducing the cost of testing and ensuring the safety of personnel, a camera is fixed in the test car and records the various driving conditions on the highway. A film recorded by the test car is used for testing in laboratory. In our laboratory, the film is played on a screen to imitate the driving conditions on the highway. The proposed system is implemented and shown in Fig. 21. The camera of the proposed system captures the image from the screen. And the proposed algorithm detects the vehicles in the captured image.

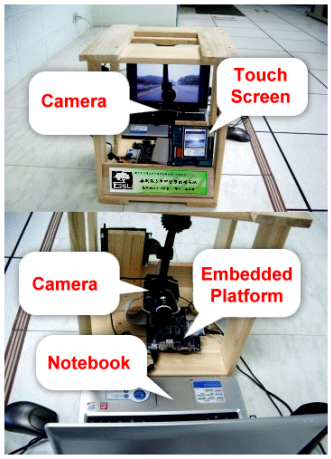

Fig. 21 The system mechanism structure

In Fig. 22, a tracked black vehicle is indicated by a yellow-colored frame. Fig. 23 shows a white vehicle is tracked in the highway ramp. Fig. 24 shows the tracking result of heavy goods trunk and bus. Because of the width and height are different from general vehicle, so the system is unable to label the image area of heavy trunk completely.

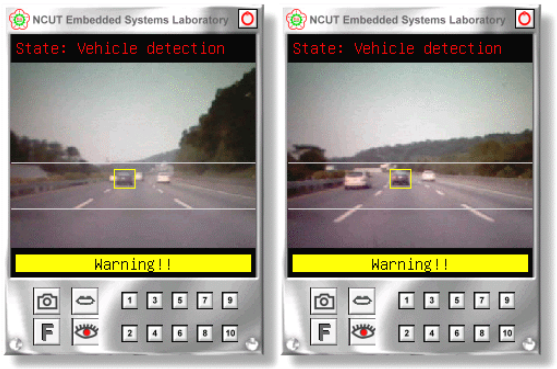

Fig. 22 Result of tracked black vehicle

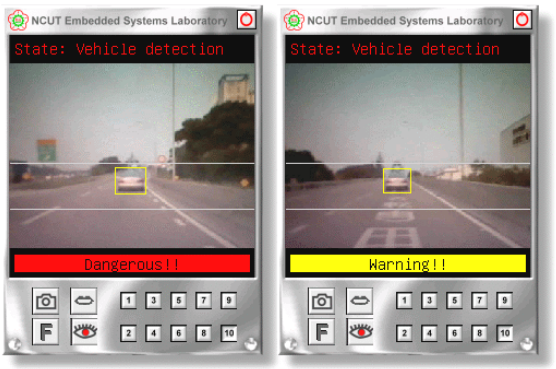

Fig. 23 Result tracked the white vehicle in the freeway ramp

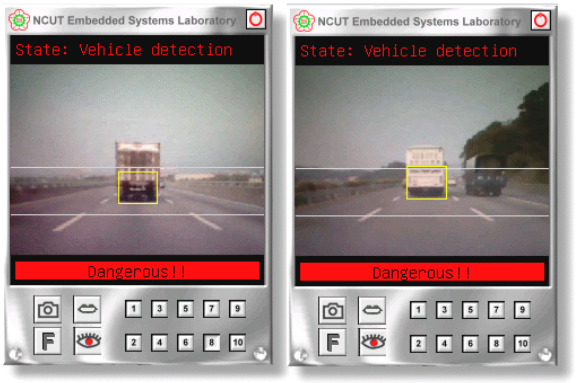

Fig. 24 Track of heavy goods vehicle and bus

5. CONCLUSION

In this paper, we proposed a modified vehicle detection algorithm to make it can be implemented on the embedded platform. The embedded platform is an INTEL XScale PXA270 SoC based platform. It has limited resources and satisfies the cost-effective consideration.

The proposed algorithm verifies the preceding vehicles driving on the highway at daytime. In our experiments, the system can correctly extract and track vehicles. And the processing time of this system is less than the human reaction time. This makes our proposed system more close to commercial products.

REFERENCES

[1] Z. Sun, G. Bebis, and R. Miller, *On-road vehicle detection using Gabor filters and support vector machines*, International Conference on Digital Signal Processing, Greece, July, 2002.

[2] A. Bensrhair, M. Bertozzi, A. Broggi, P. Miche, S. Mousset, and G. Toulminet, *A cooperative approach to vision-based vehicle detection*, In Proc. ITSC, Japan, October 2001.

[3] Z. Sun, R. Miller G. Bebis and, D. DiMeo, *A real-time precrash vehicle detection system*, Applications of Computer Vision, 2002. (WACV 2002).

[4] S. S. Huang, C. J. Chen, P. Y. Hsiao, and L. C. Fu, *On-board vision system for lane recognition and front-vehicle detection to enhance driver's awareness*, in Proc. IEEE Int'l Conf. on Robotics and Automation, New Orleans LA, Apr.26 - May 1, 2004.

[5] M. Betke, E. Haritaoglu and L.S. Davis, *Real-time multiple vehicle detection and tracking from a moving vehicle*, Machine Vision and Applications, pp. 69-83, 2000.
[6] E. Hamilton, *JPEG File Interchange Format*, C-Cube Microsystems, September, 1992.
[7] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice-Hall, Inc., 2002.
[8] N. Otsu, *A Threshold Selection Method from Gray-Level Histograms*, IEEE Transactions on Systems, man, and cybernetics, vol. SMC-9, no.1, January 1979.
[9] M. H. Schimek. (2008) V4L2 API Specification. [Online]. Available: http://v4l2spec.bytesex.org/
[10] (2008) Beijing Feynman Software Technology Co., Ltd. [Online]. Available: http://www.minigui.com/
[11] (2008) Ministry Of Transportation and Communications R.O.C. [Online]. Available: http://www.motc.gov.tw/motchypage/safe/speed_001_4.htm