Chapter 15
An Efficient Embedded System for the Detection and Recognition of Speed-Limit Signs

Hsing-Lung Chen, Ming-Sze Chen and Shu-Hua Hu

Abstract This paper is aimed at detecting and recognizing speed-limit signs in real-time. The recognition of speed-limit signs gives the necessary reminders and warnings to the drivers who may ignore the speed-limit signs. In the previous literatures, most of proposed schemes for real-time detection and recognition of traffic signs employed the software approach. It not only needs a large amount of data processing, but also requires a high demand of hardware. Therefore, we propose a hardware-software co-design scheme on FPGA, where regular computations are executed in parallel with hardware, resulting in detecting and recognizing speed-limit signs in real-time. The experiments show that our proposed scheme can detect and recognize a speed-limit sign in 62.558 ms. The detection rate is about 99 %. And the recognition rate is 100 %.

15.1 Introduction

To make road vehicles fully autonomous has been studied for many years. Different approaches have been employed: color segmentation, control theory, neural networks, etc. Obstacle detection and avoidance is an open research area. Automatic recognition of traffic signs studies started more recently, and are increasing rapidly.

H.-L. Chen (✉) · M.-S. Chen
Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan
e-mail: hlchen@mail.ntust.edu.tw

M.-S. Chen
e-mail: m9702107@mail.ntust.edu.tw

S.-H. Hu
Department of Computer Science and Information Engineering, Jinwen University of Science and Technology, New Taipei, Taiwan
e-mail: shhu@just.edu.tw

It could be used as an assistant for drivers, alerting them about the presence of some specific signs or some risky situations.

Escalera et al. [1] proposed a method for traffic sign detection and classification. They employ color threshold to segment the image and shape analysis to detect signs. Then, a neural network is applied for classification. Hirose et al. [2] proposed the *Simple Vector Filter* to distinguish between objects and background pixels. The genetic algorithm with search limits is proposed to realize a real-time position recognition. Janssen et al. [3] proposed a scheme for traffic sign recognition by using color, shape and pictogram. Martinovic et al. [4] proposed a method for automated localization of certain traffic signs. And the speed limit is determined with number character partition.

Most of previous approaches [5–9] detect the traffic signs first by locating the border of traffic signs and then extract the information within the traffic signs. They employed complicated chromaticity models which need a lot of computations. This paper employs CIE-rg chromaticity space which requires simple computations, making hardware realization feasible. Therefore, we propose a hardware-software co-design scheme on FPGA, where regular computations are executed in parallel with hardware, resulting in detecting and recognizing speed-limit signs in real-time.

15.2 Detection and Recognition of Speed-Limit Signs

15.2.1 CIE-rg Chromaticity Space

It is obvious that the most efficient method for detecting traffic signs in the picture is to employ color characterization. In this paper, we employ CIE-rg chromaticity space as color characterization because of its simple computations. CIE-rg values can be derived from RGB values by Eq. (15.1). The speed-limit signs have a white background and are surrounded by a red ring. Hence, the detection of speed-limit signs can employ the processing of red-color parts. From Fig. 15.1, it is observed that red colors are almost resided in the triangle enclosed by blue lines, which is satisfied with Eq. (15.2).

$$\begin{cases} r = R/(R+G+B) \\ g = G/(R+G+B) \end{cases} \tag{15.1}$$

$$\begin{cases} r \geq 0.329 \\ g \leq 0.324 \\ g \leq r - 0.341 \end{cases} \tag{15.2}$$

It is obvious that the rg values are ranging from 0 to 1. In order to reduce hardware resources, the required floating-point computations need to be transferred to integer computations. Hence, $r'g'$ values are introduced, which are rg

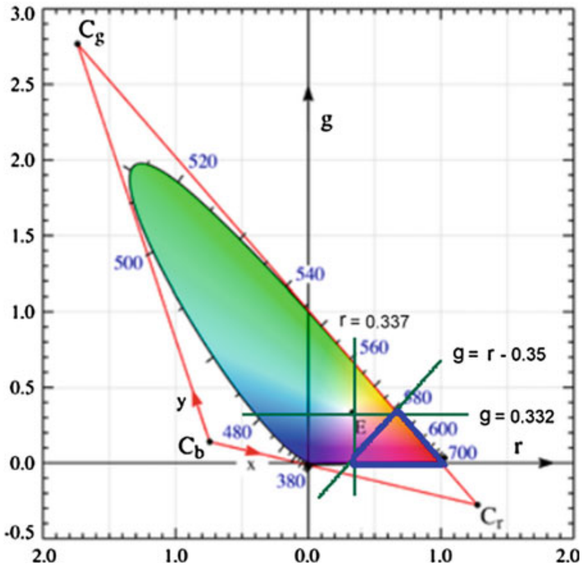

Fig. 15.1 CIE-rg chromaticity space

values multiplied by 1,024 as shown in Eq. (15.3). The pixels with red color can be identified by Eq. (15.4). With Eq. (15.4), the picture with RGB can be transferred to the binary image, where 1 represents red color and 0 non-red color.

$$\begin{cases} r' = R * 1{,}024/(R+G+B) \\ g' = G * 1{,}024/(R+G+B) \end{cases} \tag{15.3}$$

$$\begin{cases} r' \geq 377 \\ g' \leq 332 \\ g' \leq r' - 35 \end{cases} \tag{15.4}$$

15.2.2 Noise Eliminating and Boundary Smoothing

In this paper, the detection of speed-limit signs is conducted by recognizing the boundary of red-colored ring. It is important to eliminate noises and to smooth the boundary of red-colored regions. Here, 5 $\times$ 5 template is employed. Let X denote the center of 5 $\times$ 5 template and S the number of 1's in the template. By Eq. (15.5), noise eliminating and boundary smoothing can be conducted with different δ_l and δ_u. Note that X remains unchanged if $\delta_l < \mathrm{S} < \delta_u$.

$$\mathrm{X} = \begin{cases} 0, & \text{if} \quad S \leq \delta_l \\ 1, & \text{if} \quad S \geq \delta_u \end{cases} \tag{15.5}$$

Noise eliminating process is conducted gradually. At first, the process are conducted several times with $\delta_{\mathrm{l}} = 6$ and $\delta_{\mathrm{u}} = 19$, resulting in small noises being eliminated. Secondary, the process are conducted several times with $\delta_{\mathrm{l}} = 8$ and $\delta_{\mathrm{u}} = 17$ to remove big noises. Finally, the process are conducted several times with $\delta_{\mathrm{l}} = 10$ and $\delta_{\mathrm{u}} = 15$ to ensure that all noises are removed. Finally, boundary smoothing can be derived with $\delta_{\mathrm{l}} = 11$ and $\delta_{\mathrm{u}} = 14$.

15.2.3 Recognizing the Inner Boundary of the Red-Colored Ring

In this paper, assume that the picture is of size 800 × 400 pixels. The speed-limit signs with the diameter of their inner area larger than 40 pixels are considered. Hence, only rows 35, 70, …, 385 need to be scanned for searching all the speed-limit signs. The scanning begins with the left end of each scan-line. The next scan-line is going to be searched when the right end of the current scan-line is reached, as shown in Fig. 15.2a–c. Let pixel (L_x, L_y) represent the second visited boundary pixel with red color and (S_x, S_y) the third visited boundary pixel with red color. When (S_x, S_y) is reached, then scan the boundary clockwise. The case that (S_x, S_y) is in the outer boundary is considered first, as shown in Fig. 15.2d. (S_x, S_y) will be reached again and the scanning is restarted from ($S_{x\text{-}1}$, S_y), as shown in Fig. 15.2e. In the other case, (L_x, L_y) will be reached first and the inner boundary is found, as shown in Fig. 15.2f.

15.2.4 Extracting Boundary Information

The vector from the currently visiting pixel to the next visiting pixel is with one of eight degrees 0°, 45°, …, 315°. The vector codes 0, 1, … 7 denote degrees 0°, 45°, …, 315°, respectively. The labeling of adjacent pixels of the currently visiting pixel is shown in Fig. 15.3. As described before, the scanning of the inner boundary is conducted clockwise. Initially, the base pixel is with red color. Pixel(0) must be with red color because of eliminating noises. Then we search the adjacent pixels from pixel(0) clockwise until the first adjacent pixel with non-red color pixel(j) is visited. The vector code ((j + 1) mod 8) is derived and the scanning is moved to pixel ((j + 1) mod 8). In the following, the vector code is derived similarly except for the selection of the starting adjacent pixel. Since the smoothing has been conducted, the difference between the current vector code and the last vector code is at most 1. Supposed that the last vector code is i, the starting adjacent pixel is set to be pixel ((i + 1) mod 8).

The quasi-curvature of the segment is defined as the difference between the sum of vector directions of the first half and that of the second half. For simplicity, the

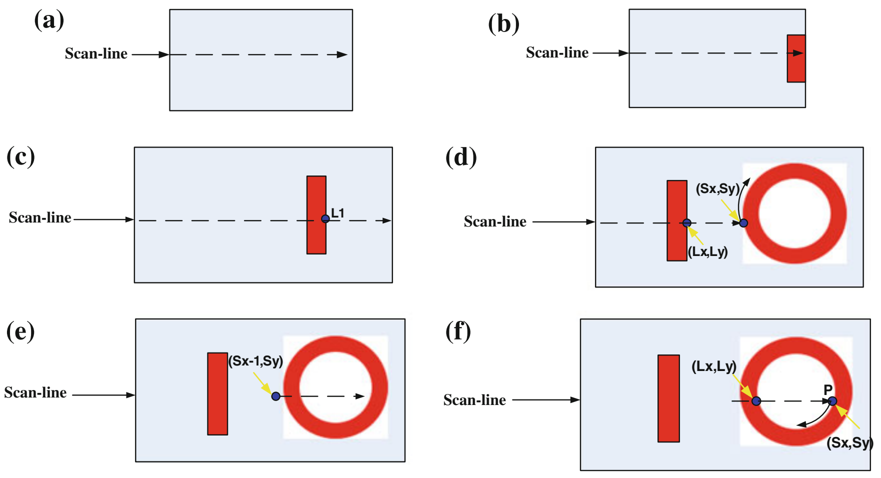

Fig. 15.2 Recognizing the inner boundary of the *red-colored ring*

Fig. 15.3 Labeling adjacent pixels

pixel(3)	pixel(2)	pixel(1)
pixel(4)	Base	pixel(0)
pixel(5)	pixel(6)	pixel(7)

vector code is employed as the vector direction. The length of the segment (2n) will be described in the next section. Supposed that the first half consists of vectors F_1, F_2, …, F_n, and the second half consists of vectors S_1, S_2, …, S_n. Without ambiguity, vector representation is used for vector code. The quasi-curvature of the segment can derived by Eq. (15.6). The difference of two vectors is difficult to calculate correctly. However the difference of two adjacent vectors is simple to calculate correctly. (F_1-S_1) can be reformulated as *n* differences of adjacent vectors, i.e., (F_1\,−\,F_2) + (F_2\,−\,F_3) + … + (F_n\,−\,S_1). Other subtractions in Eq. (15.6) can be reformulated similarly.

$$\begin{aligned} C &= (F_1 + F_2 + \ldots + F_n) - (S_1 + S_2 + \cdots + S_n) \\ &= (F_1 - S_1) + (F_2 - S_2) + \cdots + (F_n - S_n) \end{aligned} \quad (15.6)$$

The segment with the quasi-curvatures −1, 0, or +1 is defined as the straight segment (π). The segment with the quasi-curvature less than −1 is defined as the clockwise curve segment (ρ). Similarly, the segment with the quasi-curvature greater than +1 is defined as the counter-clockwise curve segment (φ). If the inner boundary consists of ρ or $\rho\pi\rho\pi$, a speed-limit sign is detected.

15.2.5 Recognizing Speed

The speed-limit sign has black numbers with white background. The first step is to derive the histogram of gray values of the inner area. The suitable threshold is determined from the derived histogram. The binary image thus can be generated by the determined threshold. By the method described in Sect. 15.2.2, noise elimination and boundary smoothing are applied on the binary image.

From the observations of numbers, the length of the segment (2n) is determined by the length and width of the character as follows.

$$2\mathrm{n} = \min(\mathrm{width} \times 0.6, \mathrm{length} \times 0.4) \tag{15.7}$$

From previous subsection, the boundary can be decomposed into several straight segments, clockwise curve segments and counter-clockwise curve segments. However, some recognized straight segments may not be straight, because their first half and second half are symmetric. The recognized straight segments must be verified by various segment lengths ($(\mathrm{sub} = 2 \times \lfloor n/2 \rfloor^i)$, for $\mathrm{sub} > 2$. Each number character can be coded with a sequence of segment types starting from the top. Each number character may have several codes with various patterns. With all of these codes, the search tree is established. In the recognition process, the derived code is matched through the search tree for recognition.

15.3 System Design

In this paper, we use DE2-70 FPGA board supported by Altera. It contains an embedded processor NIOS II and a CMOS sensor. We need to configure data transfer module and processing module through FPGA. Data transfer module is responsive for data transmission between NIOS II and processing module. Processing module conducts the generation of binary red image, noise elimination and smoothing, which require a lot of computations. Other operations are done by software with NOIS II (Fig. 15.4).

15.4 Performance Evaluation

We have collected 104 pictures. In one of pictures, the red ring of the speed-limit sign is partly covered by tree leaves. The speed-limit signs in 103 pictures are detected correctly. One traffic sign covered by tree leaves can not be detected because the red ring is not recognized. Thus, the detection rate is 99 %. The speed-limit in 103 pictures are recognized correctly. Hence, the recognition rate is 100 %.

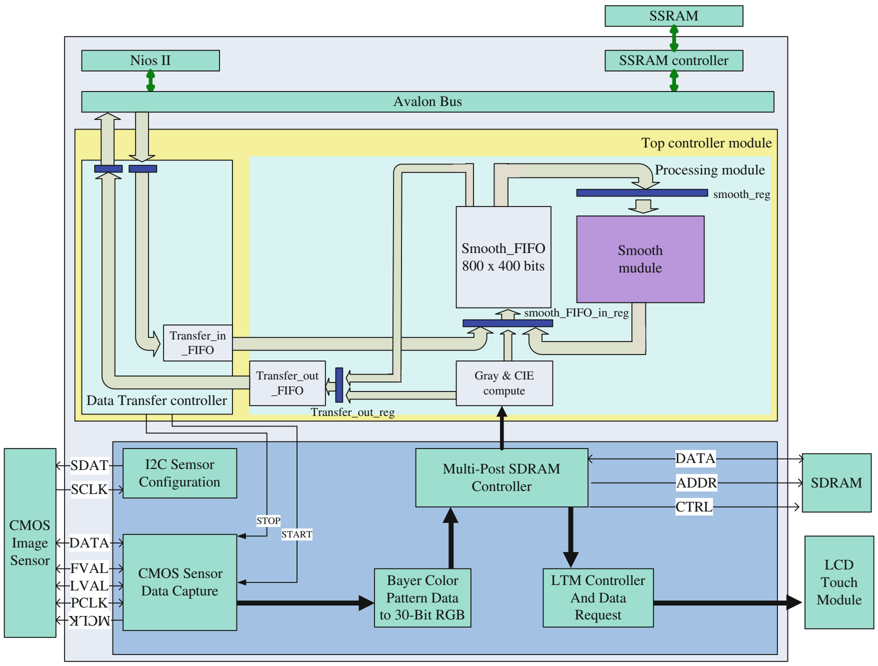

Fig. 15.4 System architecture

In DE2-70, it only needs 62.558 ms totally to recognize a picture, resulting in meeting real-time requirement. Data transfer takes 15.25 ms. Software processing in NOIS II takes lot of times (47.308 ms). Noise elimination and smoothing with hardware costs a little time (0.544 ms), which can be executed in parallel with the data transfer.

15.5 Conclusion

This paper proposes a hardware-software co-design scheme on FPGA, where regular computations are executed in parallel with hardware, resulting in detecting and recognizing speed-limit signs in real-time. The experiments show that our proposed scheme can detect and recognize a speed-limit sign in 62.558 ms. The detection rate is about 99 %. And the recognition rate is 100 %.

The boundary detection algorithm can be enhanced to recognize the border of traffic signs covered by tree leaves. Extending to most of traffic signs is worthy to be investigated in the future.

Acknowledgments This work was support, in part, by National Science Council, Republic of China, under Grant No. NSC 102-2221-E-011-067.

References

1. de la Escalera A, Moreno LE, Salichs MA, Armingol JM (1997) Road traffic sign detection and classification. IEEE Trans Ind Electron 44:848–859
2. Hirose K, Asakura T, Aoyagi Y (2000) Real-time recognition of road traffic sign in moving scene image using new image filter. In: 26th annual conference of the IEEE Ind Electron Soc (IECON 2000) 3, Nagoya, pp 2207–2212
3. Janssen R, Ritter W, Stein F, Ott S (1993), Hybrid approach for traffic sign recognition. In: Intelligent vehicles symposium '93, pp 390–395
4. Martinovic A, Glavas G, Juribasic M, Sutic D, Kalafatic Z (2010), Real-time detection and recognition of traffic signs. In: The 33rd international convention MIPR0 2010 Opatija Croatia, pp 760–765
5. Ach R, Luth N, Techmer A (2008) Real-time detection of traffic signs on a multi-core processor. In: 2008 IEEE intelligent vehicles symposium, Eindhoven, The Netherlands pp 307–312
6. Estable S, Schick J, Stein F, Janssen R, Ott R, Ritter W et al (1994), A real-time traffic sign recognition system. In: Intelligent vehicles symposium '94, pp 213–218
7. Lafuente-Arroyo S, Maldonado-Bascon S, Gil-Jimenez P, Gomez-Moreno H (2008) An intra-image tracking algorithm for traffic sign recognition. In: IEEE international conference on vehicular electronics and safety, 259–264
8. Maldonado-Bascon S, Lafuente-Arroyo S, Siegmann P, Gomez-Moreno H, Acevedo-Rodriguez FJ (2008) Traffic sign recognition system for inventory purposes, In: IEEE intelligent vehicles symposium, pp 590–595
9. Muller M, Braun A, Gerlach J, Rosenstiel W, Nienhuser D, Zollner JM et al (2010) Design of an automotive traffic sign recognition system targeting a multi-core SoC implementation. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, pp 532–537