LETTER

A New Integral Image Structure for Memory Size Reduction

Su-hyun LEE[†a)], ***Student Member and*** **Yong-jin JEONG**[†], ***Member***

SUMMARY An integral image is the sum of input image pixel values. It is mainly used to speed up the process of a box filter operation, such as Haar-like features. However, large memory for integral image data can be an obstacle in an embedded environment with limited hardware. Therefore, an efficient method to store the integral image is necessary. In this paper, we propose a memory size reduction method for integral image. The method uses four types image information: an integral image, a row integral image, a column integral image, and an input image. Using this method, integral image memory can be reduced by 42.6% on a 640 × 480 8-bit gray-scale input image. The same idea can be applied for bigger size images.

key words: *integral image, image processing, image recognition, embedded system*

1. Introduction

The idea of integral image was first introduced in "Summed-area tables for texture mapping" [1]. The characteristic of integral images to sum input values is used to speed up the process of a box filter operation, such as Haar-like features. As an example, the Viola and Jones object detector [2] and SURF (Speeded Up Robust Features) [3] can be executed in real-time on many platforms because of the computational efficiency of the integral image.

Although the integral image has the advantage in its operation speed, it needs several times more memory than the input image memory, due to its extended word length. Thus, this can be an obstacle when implementing the algorithm in an embedded environment with limited hardware.

Research, such as [4], reduces integral image memory using modulo operation and rounding. However, this approach has drawbacks, including rounding errors and an additional constraint of fixed size of the box filter. In this paper, we propose a new memory architecture for storing data lossless integral image with greatly reduced size. The hardware architecture needs simple adders and saves a split integral image in the memory.

2. Integral Image

Let i denote an input image, and ii denote an integral image. The integral image represents the sum of input image pixel values from the top left location (0, 0) up to and including the location (x, y). The integral image is expressed

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y'). \tag{1}$$

As described in [4], the size of the integral image memory requires 27-bit word length memory for 640×480 input image size.

3. Integral Image Reduction Structure

First, Eq. (1) is transformed into

$$\begin{aligned} ii(x,y) &= \sum_{x' \le x, y' \le y} i(x', y') \\ &= \sum_{x' \le x-1, y' \le y} i(x', y') + \sum_{y' \le y} i(x, y') \\ &= \sum_{x' \le x, y' \le y-1} i(x', y') + \sum_{x' \le x} i(x', y) \\ &= \sum_{x' \le x-1, y' \le y-1} i(x', y') + \sum_{y' \le y-1} i(x, y') \\ &\quad + \sum_{x' \le x-1} i(x', y) + i(x, y) \end{aligned} \quad . \tag{2}$$

The column integral image (ii_{column}) and row integral image (ii_{row}) are represented as

$$ii_{column}(x,y) = \sum_{y' \le y} i(x, y') \tag{3}$$

and

$$ii_{row}(x,y) = \sum_{x' \le x} i(x', y). \tag{4}$$

Using Eqs. (3), and (4), we transform Eq. (2) into

$$ii(x,y) = \begin{array}{l} \textit{If } x = even,\ y = even \\ \quad \sum_{x' \le x, y' \le y} i(x', y') \\ \textit{If } x = odd,\ y = even \\ \quad ii(x-1, y) + ii_{column}(x, y) \\ \textit{If } x = even,\ y = odd \\ \quad ii(x, y-1) + ii_{row}(x, y) \\ \textit{If } x = odd,\ y = odd \\ \quad ii(x-1, y-1) + ii_{column}(x, y-1) \\ \quad +ii_{row}(x-1, y) + i(x, y) \end{array} \quad . \tag{5}$$

The integral image can be stored by summing separated images using Eq. (5). Figure 1 shows how to reconstruct the

Manuscript received April 25, 2013.
Manuscript revised December 2, 2013.

[†]The authors are with Kwangwoon University, Seoul, Korea.
a) E-mail: metal477@kw.ac.kr

DOI: 10.1587/transinf.E97.D.998

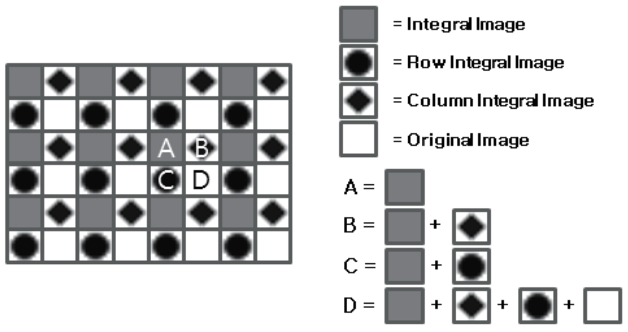

Fig. 1 Reconstruction of the integral image.

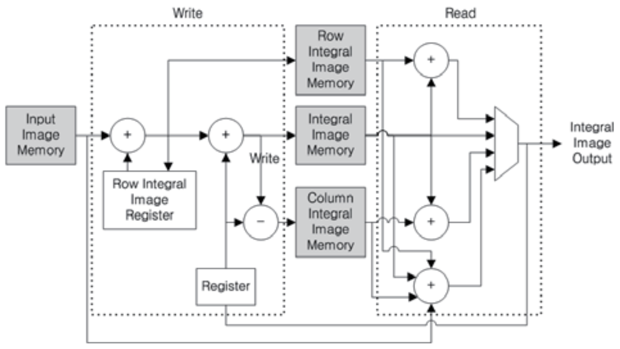

Fig. 2 Hardware architecture for the integral image reconstruction.

integral image from the separated images.

Using the method of Fig. 1, the input image memory can be reused. This uses column and row integral images smaller than the integral image. Hence, memory size can be greatly reduced. Figure 2 shows hardware architecture to reconstruct the integral image based on Fig. 1.

The hardware architecture in Fig. 2 requires 320×240×27-bit integral image memory, 320×240×18-bit row integral image memory, and 320×240×17-bit column integral image memory in the 8-bit input image of 640 × 480 size.

4. Analysis

Table 1 shows reduction rate of the integral image memory using the proposed memory structure. When using the proposed integral image memory structure, integral image memory is reduced over 40%.

Comparison with [4] is difficult, because the data size depends on the box filter size. However, the method of this paper, and the method of [4] can be used together. The paper [4] specifies the data size of the integral image, using the modulo operation according to the maximum size of the box filter. When we use both [4] and the proposed method, integral image data is replaced with the modulo operated data. If the modulo operated data size is smaller than the row and column integral image data size, the row and column integral image data is replaced with the modulo operated data. Even if it were so, more efficient memory usage is possible, because a quarter of the input image data is reused. Table 2 shows comparisons of integral image memory capacity.

Table 1 Proposed integral image memory capacity and reduction rate. (8-bit word length of input image)

Image size	Integral image memory capacity (bits)		Reduction rate (%)
	General structure	Proposed structure	
320×240	1,920,000	1,113,600	42.0
640×480	8,294,400	4,761,600	42.6
800×600	12,960,000	7,460,000	41.7
1024×768	22,020,096	12,582,912	42.9
1280×1024	38,010,880	21,626,880	43.1
1600×1200	55,680,000	32,160,000	42.2
2048×1536	94,371,480	53,477,376	43.3

Table 2 Comparison of integral image memory capacity. (input image: 640 × 480 size, 8 bit word length)

Maximum Filter Size	Integral image memory capacity (bits)		
	[4]	Proposed structure	Proposed structure with [4]
16×16	4,915,200	4,761,600	3,686,400
256×256	7,372,800	4,761,600	4,224,000

Table 3 The execution overhead of the proposed integral image structure. (8-bit word length of input image, CPU i5 3.3 Ghz, Main Memory 3 GB)

Image size	Execution Time (msec)				Overhead (msec)	
	General structure		Proposed structure			
	Write	Read	Write	Read	Write	Read
320×240	0.54	0.17	0.58	0.47	0.04	0.30
640×480	1.64	0.70	1.74	1.84	0.10	1.14
800×600	2.56	1.13	2.80	3.13	0.24	2.00
1024×768	3.85	1.86	4.36	4.88	0.51	3.02

The proposed structure is efficient for memory usage, but it has additional operations. We compared the proposed structure with a general structure in PC environment using SURF algorithm [3]. Table 3 shows the execution overhead of the proposed integral image memory. Read operations need more overhead than write operations, because read operations have conditional branches and additional calculations like Eq. (5). Execution overhead of VGA (640 × 480) size image is 1.24 msec, which is not a problem in real-time processing. And the results show that larger images have more overhead. However, it needs just one more clock for read in hardware environment, because parallel operation is possible using the shown in Fig. 2.

5. Conclusion

Integral image is used for many image processing algorithms, but its huge memory size is a main obstacle for hardware implementation. We propose new integral image memory structure for memory size reduction. As a result, we could get reduction rate over 40%. The proposed method needs additional operations. However, it does not limit the filter size like [4]. And the proposed method and the method of [4] can be combined together for further reduction.

Acknowledgments

This work has been supported in part by the Research Grant of Kwangwoon University in 2013.

References

[1] F. Crow, "Summed-area tables for texture mapping," SIGGRAPH'84: Proc. 11th Annual Conference on Computer Graphics and Interactive Techniques, pp.207–212, New York, NY, USA, 1984.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proc. CVPR 2001, pp.511–518, 2001.

[3] H. Bay, T. Tuytelaars, and V. Gool, Luc, "Speeded up robust features (SURF)," Computer Vision and Image Understanding, vol.110, no.3, pp.346–359, June 2008.

[4] H.J.W. Belt, "Word length reduction for the integral image," 15th IEEE International Conference on Image Processing, ICIP 2008, pp.805–808, San Diago, USA, 2008.