AN EFFICIENT ARCHITECTURE FOR ADAPTIVE PROGRESSIVE THRESHOLDING

*H. Tian, S. K. Lam, T. Srikanthan, C. H. Chang**
Center for High Performance Embedded Systems, Nanyang Technological University,
N4-B3B-06, Nanyang Avenue, Singapore 639798
*Phone: +65-67905873, Fax: +65-67920774, Email: echchang@ntu.edu.sg

ABSTRACT

A new pipelined architecture for adaptive progressive thresholding (APT) is proposed. Unlike the conventional architectures that rely heavily on multipliers and dividers to evaluate the maximum between-class variance, our method employs a reconfigurable logarithmic computing unit to simplify the circuitry and increases the application's agility to operational variation. Our logarithmic conversion algorithm avoid large look-up table by streaming the operand's bits into segments so that several small look-up tables and simple adders and shifters are sufficient to guarantee an accurate result. Besides being cost effective, the proposed architecture has also significantly reduced the latency of the critical pipelined stage.

1. INTRODUCTION

Segmentation is an important step in digital image processing and thresholding technique is instrumental to the success of many segmentation methods. Many algorithms have been proposed to automate the threshold selection adaptively based on the spatial properties of the given image [1][2]. The Adaptive Progressive Thresholding (APT) method presented in [3] can be used for the real-time segmentation of gray level images and a pipelined architecture for the implementation of the APT technique was developed in [3]. However, there are too many complex arithmetic operations such as multipliers and dividers in the present circuit implementation of the pipelined architecture of APT. For truly demanding high-speed or cost sensitive applications, binary logarithms can be used as an alternative to reduce the complex operations to simple arithmetic and improve the overall processing speed [4][5].

In this paper, a novel architecture for computing between-class variance of APT is developed by incorporating a logarithm conversion algorithm. The logarithm conversion adopted is a hybrid approach that strives to balance the size of the lookup table and the amount of computational logic. One distinctive advantage of our method is that it allows the accuracy of the APT to adapt to the application requirement by modulating the lookup table content and addresses, which can be done on a reconfiguration platform dynamically. A comparison with the architecture of [3] in terms of the basic operations required shows that our improved architecture for APT has substantially lowered the area-time complexity.

2. ADAPTIVE PROGRESSIVE THRESHOLDING

Otsu's method [6] is to find the optimal threshold t by maximizing the ratio of between-class variance σ_B^2 and total variance σ_T^2 as follows:

$$t = Arg\{\max_{0\le i\le L-1}(\sigma_B^2/\sigma_T^2)\} \quad (1)$$

In Eq. (1), L is the gray levels and

$$\sigma_B^2 = w_t(1-w_t)\left(\frac{\mu_T-\mu_t}{1-w_t}-\frac{\mu_t}{w_t}\right)^2 \quad \text{and} \quad \sigma_T^2=\sum_{i=0}^{L-1}(i-\mu_T)^2\frac{n_i}{N} \quad (2)$$

where n_i represents the number of pixels with gray level i and N is the total number of pixels in the image. w_t, μ_t and μ_T are defined as:

$$w_t=\sum_{i=0}^{t}\frac{n_i}{N}, \quad \mu_t=\sum_{i=0}^{t}i\frac{n_i}{N} \quad \text{and} \quad \mu_T=\sum_{i=0}^{L-1}i\frac{n_i}{N} \quad (3)$$

The APT algorithm can be perceived as an iterative Otsu's procedure. When the threshold value t is obtained, the image is further divided into two classes. The pixels in the virtual new image will have a gray level range of $\{0,1,2,\ldots,t\}$. This Otsu procedure is applied recursively until a convergence criterion governed by the Cumulative Limiting Factor (*CLF*) is met. The *CLF* for the Δth iteration is defined as:

$$CLF(\Delta)=\frac{\sigma_B^2(\Delta)}{\sigma_T^2}, \quad for\ \Delta\ge 1 \quad (4)$$

The iterative procedure is stopped whenever:

$$CLF(\Delta)\le\alpha\frac{\mu_T}{\sigma_T^2} \quad (5)$$

where α is a constant. The value of $t(\Delta)$ obtained at the final recursion will be the optimal threshold of APT.

3. CONVENTIONAL ARCHITECTURE

In this section, we review the pipelined APT architecture designed for an eight-bit gray level image of size 256×256 proposed in [3] to highlight the drawbacks of the between-class variance computation module. It consists of the cumulative histogram and intensity area computation, the between-class variance computation, and the final threshold computation modules as shown in Figure. 1.

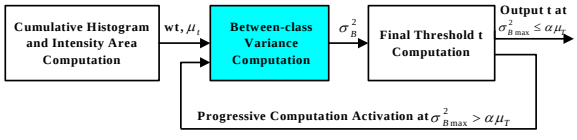


Figure 1. Computational units of APT architecture system

The module for Cumulative Histogram (CH) and Intensity Area (CIA) Computation are used to compute and store w_t and μ_t respectively. The module for Final Threshold Computation consists of a comparison process to identify the optimal threshold t. Both of these modules comprises of relatively simple straightforward operations in comparison to the between-class variance computation process.

3.1 Architecture of between-class variance computation

The layer of the APT architecture developed for the computation of the between-class variance is shown in Figure. 2. The register arrays of CIA and CH are divided into 16 blocks with 16 registers in each block. These register blocks function independently as 16 different arrays. Reg1 and Reg2 hold the contents shifted out from the top of the CIA and CH arrays, respectively. They are used for computing the value of $\alpha\mu_T$ in each iteration. Each block has an address counter to hold the index of the current register for transferring the data. The *i*th block of CH register array is circularly shifted and the value of w_t is computed. μ_t is calculated in a similar manner. Based on the computed values of w_t and μ_t, $1-w_t$, $\mu_T-\mu_t$ and finally $\sigma^2_{B\,\max|i}$ can be obtained. There are 16 parallel blocks performing their computations simultaneously in a pipelined fashion to determine $\sigma^2_{B\,\max|i}$ for i = 1, 2,…,16.

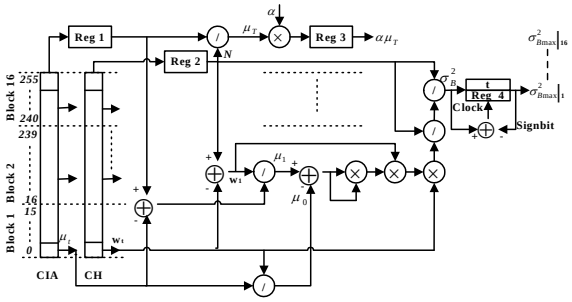


Figure 2. Architecture for the computation of σ^2_B

4. PROPOSED LOGARITHM CONVERSION ARCHTECTURE FOR APT

The bottleneck pipelined stage of the conventional architecture is the computation of between-class variance σ^2_B after the CH and CIA arrays have been calculated. We proposed the use of an efficient binary logarithm conversion unit to convert the computational intensive high precision multiplication and division operations into fixed point circuitry composed of only simple multiplexer, adders and shifters.

4.1 The algorithm

Let $(z_{u-1}\ z_{u-2}\ \ldots\ z_0\ .\ z_{-1}\ z_{-2}\ \ldots\ z_{-v})_2$ be a binary representation of Z and Z_j is the first non-zero bit encountered from the most significant bit of Z. The value of this number can be written as:

$$Z = 2^j + \sum_{i=-v}^{j-1} 2^i z_i = 2^j(1 + \sum_{i=-v}^{j-1} 2^{i-j} z_i) = 2^j(1+x) \qquad (6)$$

where $0 \le x < 1$. Thus,

$$\log_2 Z = j + \log_2(1+x) \qquad (7)$$

where j is the characteristic of the logarithm and $\log_2(1+x)$ constitutes the mantissa. The approximation can be obtained using the linear term in the Taylor's series, $\log_2(1+x) \approx x$. However, this approximation is inaccurate and is subjected to a maximum error of $0.08639(\log_2(1+x)-x)$.

To improve the accuracy, a look-up table can be used to represent the fractional part of Eq. (7), i.e., $\log_2(1+x)$. However, for an n-bit binary representation, the size of the look-up table has a complexity of $O(2^{n-1})$. As the data length increases, this method becomes costly in view of the large VLSI area required. To shrink the size of the look-up table, we partition the lower order bits of the fractional part into groups and use them to access several smaller look-up tables. The values retrieved from these look-up tables are added to obtain the final result.

Consider a 16-bit binary number $Q = (q_{15}\ q_{14}\ \ldots\ q_0)_2$. Let α corresponds to the index, j of the first non-zero bit encountered from the most significant bit of Q. First, we calculate all the fractional logarithm values based on the combinations of the four-bit binary word formed by $(q_{j-1}\ q_{j-2}\ q_{j-3}\ q_{j-4})_2$ which is the next four bits to the right of j. The four-bit binary word $(q_{j-1}\ q_{j-2}\ q_{j-3}\ q_{j-4})_2$ is used as the address to the small look-up table (LUT) and the calculated fractional value β is mapped to the memory location pointed by this address. Now, there remains at most 11 bits to approximate the value of the fractional logarithm of Q. Utilizing a multiplier for this purpose incurs a large computational unit and deficits our objective to reduce the area complexity. It is observed and verified by rigorous experimentation that the size of the multiplier, if used, depends on the number of lower order bits to the right of q_{j-4}. Hence, the remaining lower order bits of Q are divided into four multiplier groups Z_A, Z_B, Z_C and Z_D corresponding to $(q_{j-5}\ q_{j-6})_2$, $(q_{j-7}\ q_{j-8})_2$, $(q_{j-9}\ q_{j-10})_2$ and $(q_{j-11}\ q_{j-12})_2$, respectively. The multiplicands of each multiplier are calculated for every combination of $(q_{j-1}\ q_{j-2}\ q_{j-3}\ q_{j-4})_2$ and they form the content D_A, D_B, D_C and D_D of another four small lookup tables addressable by $(q_{j-1}\ q_{j-2}\ q_{j-3}\ q_{j-4})_2$. The complete approximation algorithm can be mathematically expressed as follows:

$$\log_2(Q) = \alpha + \beta + (D_A * Z_A) + (D_B * Z_B) + (D_C * Z_C) + (D_D * Z_D) \qquad (8)$$

It should be noted that this algorithmic approximation has a worst case error of 0.0009822 for a 16-bit operand, which is, to the best of our knowledge, the smallest among all existing linear approximation method.

4.2 Architecture for logarithm conversion

A simple design of the logarithm conversion unit of a 16-bit decimal number is shown in Figure. 3.

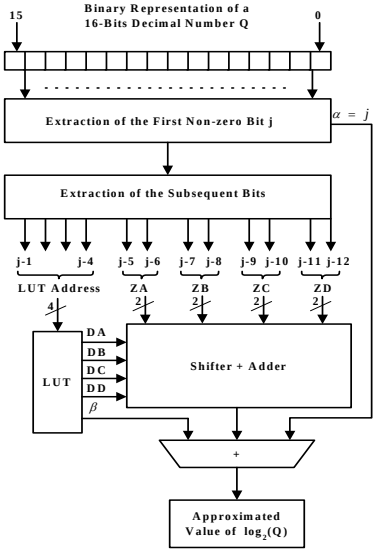


Figure. 3 Architecture of logarithm conversion unit (LCU)

The resulting architecture for a logarithm conversion unit (LCU) of a 16-bit binary number requires only a small look-up table of size 448 bits. The address to the LUT and the values of Z_A, Z_B, Z_C and Z_D can be obtained by the output of a barrel shifter of Q. In the event that the least significant bit has been shifted out, zeros are appended. Since there are only four discrete binary values of Z_A, Z_B, Z_C and Z_D, the multiplications in Eq. (8) can be replaced by a four input multiplexer to select either the constant 0, the extracted content D, a left shifted copy of D (i.e., $2D$) or a shift and accumulated copy of D (i.e., $3D = 2D+D$) with Z as the selector input. Thus, the entire conversion can be achieved with a much simpler circuitry.

4.3 Proposed architecture for APT

From Eq. (2), it is evident that the straightforward implementation of the computation of σ_B^2 requires a large number of computational intensive operations of squaring, multiplication and division. Since the optimal threshold value t corresponds to the maximum value obtained by comparison, this computation requirement can be simplified by maximizing the logarithm of σ_B^2 instead of the exact between-class variance in Eq. (1). σ_B^2 in Eq. (2) can be rewritten as follows:

$$\sigma_B^2 = \frac{\left(w_t \mu_T - \mu_t\right)^2}{\left(1 - w_t\right) w_t} \tag{9}$$

The binary logarithm conversion method is employed to calculate the value of σ_B^2 in Eq. (9) leading to:

$$\log_2 \sigma_B^2 = 2\log_2(w_t \mu_T - \mu_t) - \log_2(1 - w_t) - \log_2 w_t \tag{10}$$

The anti-logarithm function is not required in this case as we are only interested in finding out at which iteration the logarithm of σ_B^2 has attained its maximal value. Figure. 4 shows the architecture for the computation of σ_B^2.

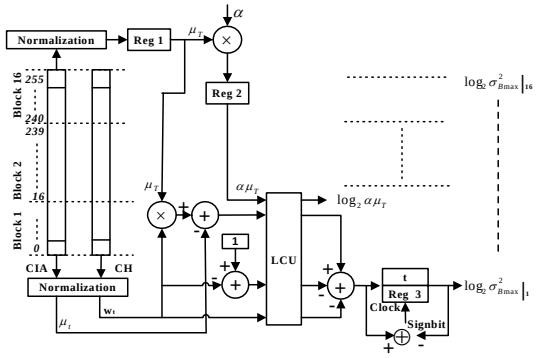


Figure 4. Proposed architecture for computation of between-class variances

5. COMPLEXITY COMPARISION

Table 1 shows an analysis of the computational complexity by comparing the hardware resources used in the conventional and proposed architectures for APT. The resource utilization appeared in Table 1 compares only the two critical architectures shown in Figure. 2 and 4 for the computation of between-class variance.

Table 1. Comparison of conventional and proposed architectures

Operations	Conventional	Proposed
23-Bit Fix-point Divider	32	Nil
7-Bit Fix-point Multiplier	16	Nil
10-Bit Fix-point Multiplier	1	1
17-Bit Fix-point Multiplier	Nil	16
30-Bit Fix-point Multiplier	16	Nil
46-Bit Fix-point Multiplier	16	Nil
4-Bit Adder/substractor	Nil	16
8-Bit Adder/substractor	16	Nil
10-Bit Adder/substractor	Nil	16
16-Bit Adder/substractor	16	Nil
17-Bit Adder/substractor	Nil	16
23-Bit Adder/substractor	16	Nil
16-Bit Right Shifter	33	1
6-Bit Right Shifter	Nil	32
56 bytes LUT	Nil	16

From Table 1, the proposed architecture completely eliminates the dividers required in the conventional architecture. In addition, the number of multipliers has been reduced from 49 to 17. The 17 multipliers used in our proposed architecture are of a smaller size (operands' width of 17 bits versus 46 bits). Although the proposed architecture uses 16 look-up tables, the size of each LUT is only 56 bytes for a 16-bit precision logarithm

approximation. Hence, the total memory required for the LUTs is 896 bytes. The number of transistors required for all the LUTs is still far less than a 46-bit multiplier. The other trivial resources are shifters and adders used to manipulate the content retrieved from the LUT for the logarithm computation (see Figure. 3). Based on the number and complexity of the required operations, we can conclude that the proposed architecture has substantially reduced both the time and space complexities of the conventional hardware architecture for APT.

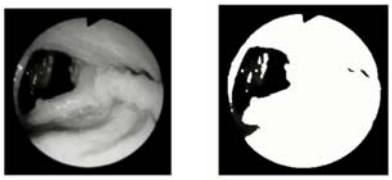
Table 2 shows the maximum error incurred by our logarithmic approximation for various different sizes of LUT. Depending on the accuracy demanded by the application, the size and content of the LUTs can vary.

Table 2. Precision verse size of look-up table

No. of address lines	LUT's size (bits)	Max. error
2	112	0.0102083
3	224	0.0031427
4	448	0.0009822
5	896	0.0003226
6	1792	0.0001113

6. A TYPICAL APPLICATION

A typical application that can be benefited directly from the proposed APT architecture is the detection of the lumen region of endoscopic images in real time. Lumen region is the area in the gastro-intestinal endoscopic image where the mean intensity is the lowest. High-speed segmentation of lumen region is essential to facilitate online navigation for automated micro-robotic endoscopy. A typical endoscopic image with the resolution of 256×256 pixels and its thresholded image obtained by our method are shown in Figure. 5. With the limiting parameter α chosen optimally to be 9.8, the value of CLF and optimal threshold was found to be 0.7789 and 32, respectively for the thresholded image in Figure. 5.



(a) (b)

Figure 5. Segmentation of an endoscopic image: (a) original image (b) thresholded image

Normalization of w_t and μ_t is necessary to ensure that each input to the logarithmic computing unit is represented in 16-bit precision to fulfill the accuracy required by the application. This is achieved by first multiplying the operand by 1024 (2^{10}) and then divide the result by the image size of 256×256 (2^{16}). Effectively, the normalization can be implemented with a constant linear shifter shifting the operand right by 6 bits.

In this application, the limiting parameter, α depends on the camera lighting environment and the reflection characteristics of the objects and should be made adaptive to the light intensity and distribution of the light sources mounted on the endoscope [3]. These requirements can be translated into a set of initialization parameters controlling the precision of the computation and the speed of the maneuver. As noted in the preceding section, the precision of the between-class variance can be adjusted by changing the size and content of the lookup table for logarithm computation. The LCU can be dynamically reconfigurable to adapt to the changes in limiting parameter α required for different endoscopic cameras and illumination conditions. Such versatility is very difficult to address by conventional architecture without incurring excessive area overhead and re-engineering effort.

7. CONCLUSIONS

A novel re-configurable architecture for APT method in real-time segmentation of images is proposed in this paper. The design exploits a hybrid logarithmic conversion algorithm, which combines the advantages of both look-up table based method and computational based approach to overcome the bottleneck of finding the maximum between-class variance. The logarithmic approximation operates on small look-up tables with simple logical shift and addition operations. Thus, it gets rid of the complexity associated with the use of multipliers, divisions and exponentiation present in the conventional APT architectures. More significantly, the ease of synchronizing the precision of the results according to the limiting parameter of the APT makes it capable of adapting to versatile optical characteristics and illumination conditions encountered as a result of equipment variation and changing operation environment of the application.

REFERENCES

[1] P. K. Sahoo, S. Soltani, A. K. C. Wong and Y. C. Chen, ``A Survey of Thresholding Techniques", *Comput. Vision Graphics Image Process*, Vol. 41, pp. 233-260, 1988.

[2] C. A. Glasbey, ``An Analysis of Histrogram-based Thresholding Algorithm", *CVGIP: Graphical Models and Image Proccesing*, Vol. 55, No. 6, pp. 532-537, 1993.

[3] K. V. Asari, T. Srikanthan, S. Kumar, and D. Radhakrishnan, ``A Pipelined Architecture for Image Segmentation by Adaptive Progressive Thresholding ", *Microprocessors and Microsystems*, Vol. 23, No. 8-9, pp. 493-499, 1999.

[4] I. Koren, *Computer Arithmetic Algorithms*, Prentice Hall, 1993.

[5] D. K. Kostopoulos, ``An Algotithm for the Computation of Binary Logarithms", *IEEE Transactions on Computers*, Vol. 40, No. 11, pp. 1267-1270, 1991.

[6] N. Otsu,, ``A Threshold Selection Method from Gray-level Histograms", *IEEE Trans. Syst., Man and Cybern.*, SMC-9, pp. 62-66, 1979.