論文
3-3-6

Real-time 2-D Separable Median Filter
(실시간 2차원 Separable 메디안 필터)

Jae Gil Jeong*
(정 재 길)

ABSTRACT

A 2-D median filter has many applications in various image and video signal processing areas. The rapid development in VLSI technology makes it possible to implement a real-time or near real-time 2-D median filter with reasonable cost. For the efficient VLSI implementation, the algorithm should have characteristics such as small memory requirements, regular computations, and local data transfers. This paper presents an architecture of the real-time two-dimensional separable median filter which has appropriate characteristics for the VLSI implementation. For the efficient two-dimensional median filter, a separable two-dimensional median filtering structure and a bit-sliced pipelined median searching algorithm are used. A behavioral simulator is implemented with C language and used for the analysis of the presented architecture.

요 약

2차원 메디안필터는 정지영상 및 동영상 신호처리 분야에 많이 활용되고 있다. 최근의 급속한 VLSI기술의 발전은 적절한 비용으로 실시간 2차원 메디안 필터의 구현을 가능하게 하여 주고 있다. 효율적인 VLSI구현을 위해서는 적은 양의 메모리 사용, 규칙적인 계산, 지역 데이터 전달 등의 특성을 갖는 알고리즘과 VLSI 구조가 필수적이다. 본 논문에서는 위와 같은 특성을 갖는 새로운 실시간 2차원 메디안필터의 VLSI구조를 제안하였다. 이를 위하여 메디안필터링 알고리즘을 분석하여 메디안 필터링 알고리즘에 내재되어 있는 병렬처리 특성, 특히 파이프라인 가능성을 최대한 활용할 수 있도록 하였다. 또한 Separable 2차원 메디안 필터링 알고리즘을 사용하여 하드웨어 복잡성을 크게 감소시켰다. Separable 2차원 메디안필터는 기존의 메디안필터와 거의 유사한 성능을 보여주었으며 bit-slice pipeline median searching 알고리즘은 기존의 메디안 필터에서 문제가 되었던 window의 크기에 따라 하드웨어 복잡성이 크게 증가하는 문제를 해결하여 window 크기에 관계없이 2차원 실시간 메디안 필터의 VLSI 구현을 가능하게 하여 주었다. C 언어를 이용한 행위레벨 시뮬레이션을 통하여 성능을 확인하고 분석하였다.

* 정회원 : 배제대학교 정보통신공학과 부교수

논문접수 : 2002. 2. 14.
심사완료 : 2002. 3. 2.

1. Introduction

There are many useful nonlinear image processing algorithms. Among them, a median filter, which was suggested by Turkey[1], has many applications such as most computer-assisted tomographic scan systems, laser-imaging radar, thermographic imaging, and biomedical signal processing application[2].

A median filtering is a simple digital technique for signal smoothing that performs particularly well where the noise is impulsive. Moreover, when the spectra of the desired signal and of the noise are mixed in the same range, the performance of median filters is much better than that of linear filters. These situations frequently arise in image processing.

The median filter is a specific case of the order statistic determination filter. There are algorithms oriented towards implementation on a general purpose computer. These include histogram[10] and partition search algorithms[11, 12]. In all of these, a dynamic data structure is maintained. This is not a desirable property for the pipelined or parallel implementations, specially for the VLSI system. Fisher[14] suggested a systolic implementation, which also has to maintain some intermediate data structure which should be transmitted between the processing elements. There are some algorithms based on threshold decomposition and stacking properties of the median and other rank-order operators[7-9]. There are several algorithms based on the bit-sliced pipelined algorithms [4-6] which are most efficient in terms of complexity, so far.

The rapid development in VLSI technology make it possible to implement a real-time or near real-time median filter with reasonable cost. For the efficient VLSI implementation, the median filtering algorithm should have characteristics such as small memory usage, regular computation or logical operation, and local data communication for multiprocessor system (no global communication).

This paper presents an architecture of the real-time two-dimensional separable median filter which has appropriate characteristics for the efficient VLSI implementation.

2. A separable 2-D median filter

The most frequently used median algorithm is based on maintaining the gray level histogram[10] of the $m \times n$ pixels in the window. The window size is $m \times n$ and the histogram is continuously updated as the window moves. Requiring global memory for the histogram is the biggest problem of this structure. If we implement this structure with a large fast computer such as a supercomputer, this is not a big problem, but with VLSI implementation, it is a big problem. Because this structure requires global communications rather than local inter-processor communications. This becomes a critical timing factor for the real-time processing.

Memory requirements for this structure with $m \times n$ window size, $M \times N$ frame size and 2^L gray levels (L bits per pixel). A memory size for the histogram is $2^L \times \lceil \log_2(m \times n) \rceil$ bits. For example, if $m = 5$, $n = 5$, $L = 8$, $M = 512$, and $N = 512$, then about 16 Kbits of memory are required. The required buffer is not small for the single chip implementation. In some cases, we can use external memory, then memory accessing time will be an important factor for the real-time processing. For real-time processing of above example with 40 frames per second, one output should be computed for about 95 nsec. At least $2 \times m \times n$ memory accesses are required for the one median. For the example, the system requires a very fast memory system which allows 20 accesses for 95 nsec.

The separable 2-D median filter is a two dimensional nonlinear filter which results from

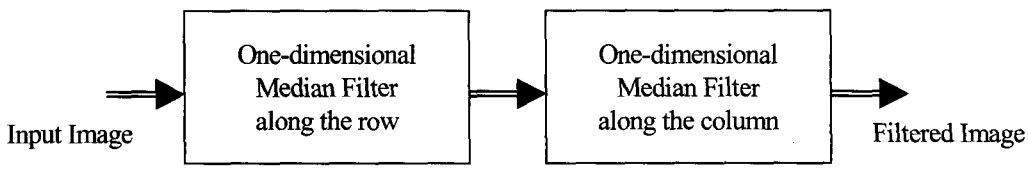
successive application of a one-dimensional median filter of size m applied first along the rows and then a one dimensional median filter of size n applied along the columns of an image (or vice versa). Although the result is not identical to the non-separable two-dimensional median filter using a $m \times n$ window, Narendra[3] showed that the separable filter is comparable in performance to the non-separable two-dimensional median filter.

Let us consider an $M \times N$ image to be median filtered. The complexity of the general algorithm to find a median of n points based on "Quick Sort" is of order n, ($O(n)$). Separable median filtering of $M \times N$ image requires one m point median and one n point median per pixel versus one $m \times n$ point median per pixel. Therefore, complexity of non-separable 2-D median filter is $O(m \times n)$ and complexity of separable 2-D median filter is $O(m) + O(n)$이다.

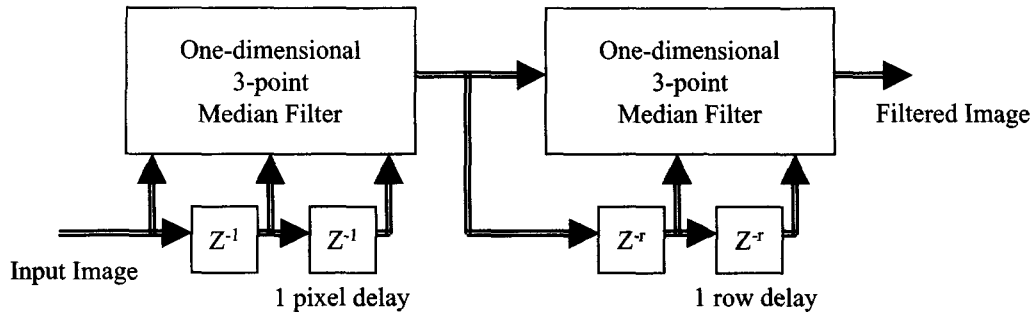
The complexity of separable 2-D median filter is much smaller than the complexity of non-separable 2-D median filter. It is not necessary to store $M \times N$ intermediate row-filtered image before filtering along columns. Most images are stored sequentially in bulk storage in a progressive scanned format. It is only necessary to store the M most recent row-filtered row of the image in active memory at a given instant.

The good properties of this structure is that it can be easily implementable with multiple processors, i.e. this system can be implemented with several simple VLSI chips. Pipelining in the chip is based on the sorting algorithm involved in a one-dimensional median operation. And this structure do not need global memory. We can distribute the memory to the each processing element equally.

The presented design is based on a separable median filtering algorithm, which can be easily modularizable, requires only local communication, and can be easily pipelinable in a global structure with multiple processors. The overall structure of a separable median filter is in [Fig. 1]. And the real-time separable 3 x 3 median filter is shown in [Fig. 2].



[Fig. 1] A structure of the separable 2-D median filter



[Fig. 2] A real-time separable 3x3 median filter

3. A Pipelined Median searching algorithm

3.1 Algorithm

Even though this separable median filter shows good implementable properties, we need a good sorting (median searching) algorithm to implement a real-time median filter for the large window. At video rates (say, 10 million samples per second), the sorting hardware has to find the median of its input in 100 nsec. To achieve these speed in a digital implementation, a pipeline hardwired architecture is necessary. Unfortunately, $O(n)$ algorithms such as "Quick Sort" are difficult to implement in hardwired architectures because they involve complex decision points. More important for the real-time implementation are sorting networks, which use digital comparator modules as basic building blocks to perform sorting[13]. But the number of comparator needed in sorting networks is steeper than linear function of the window size. If window size is large, then it is impossible to build this one-dimensional real-time median filter.

This means more hardware cost for the system. Narendra[3] suggested an all-analog diode comparator network. This all-analog approach has some problems such as output required to be quantized, and the complexity of the diode network in the analog median filter increases exponentially with the number of inputs (window size).

Instead of above median searching algorithm, I used the algorithm based upon the order statistic determination algorithm by Hoctor and Kassam[5]. This algorithm operates within the number of steps less than or equal to the number of bits in the binary representation of the input. The algorithm uses a bit-level thresholding operation on input data with adaptive binary weights, and it does not require any auxiliary data structures. I used a bit level pipeline structure of this algorithm to implement a real-time pipeline one-dimensional median filter. This algorithm can be implemented only with logical operations, and hardware complexity is small enough to be fit in a single chip.

A basic unit of this processor is a cell which can compute a median from the number of data. This cell is designed based on the algorithm by

Hoctor and Kassam[5]. This algorithm is as follows:

When the input data is denoted by $W=\{x_1, x_2, \cdots, x_m\}$ where the window size is m. And, the number of gray level is 2^L. Each pixel has L-bit binary representation, that is, an ordered L-tuple $\{b_L^j, b_{L-1}^j, \cdots, b_1^j\}$, $b_i^j=1$ *or* 0 for $i=1,\cdots,L$ and $j=1,\cdots,m$, and $x_j=\sum_{i=0}^{L} b_i^j 2^{i-1}$.

We define the N-tuple $R_k\equiv\{b_{L-k+1}^1, b_{L-k+1}^2, \cdots, b_{L-k+1}^m\}$ for $k=1,\cdots,L$, as the k-th most significant bit-row of the data window. Also, we denote by $r=1,\cdots,$ *or* m, the index of the order statistic to be selected from the data window, and by $x^{(r)}$ its value. That is, $x^{(r)}$ is an r-th largest element of W.

The algorithm operates by processing the bit-rows of W sequentially, from most significant bit position to least significant bit position. At each step of execution of the algorithm involves the followings:

For a given integer threshold value T_k (for a median $T_1=\frac{m+1}{2}$) and binary N-tuples $P_k\equiv\{p_k^1, p_k^2, \cdots, p_k^m\}$ and $R_k\equiv\{b_k^1, b_k^2, \cdots, b_k^m\}$, we have

$$f(R_i; P_i, T_i)=\begin{cases} 1 & \text{if } \sum_{j=1}^{m} p_k^j(1-B_k^j)< T_k \\ 0 & \text{if } \sum_{j=1}^{m} p_k^j(1-B_k^j)\geq T_k \end{cases} \quad (1)$$

For a median, the initial value of T_k is $T_1=\frac{m+1}{2}$, while the initial value of P_k is $P_1=\{1,1,\cdots,1\}$. A progression of P_k's and T_k's is computed by the following recurrence relations, starting with $k=1$ and proceeding to $k=L-1$.

if $\sum_{j=1}^{m} p_k^j(1-b_k^i)< T_k$ then

$$P_{k+1}=P_k \& R_k$$

$$T_{k+1}=T_k-\sum_{j=1}^{m} p_k^j(1-b_{L-k+1}^n) \quad (2)$$

else

$$P_{k+1}=P_k \& \overline{R_k}$$

$$T_{k+1}=T_k$$

where " &" represents the bit-wise logical "AND" operation and " $\overline{R_k}$" is the complement of " R_k". After L iterations, the value of $x^{(r)}$ i.e. r-th order statistic

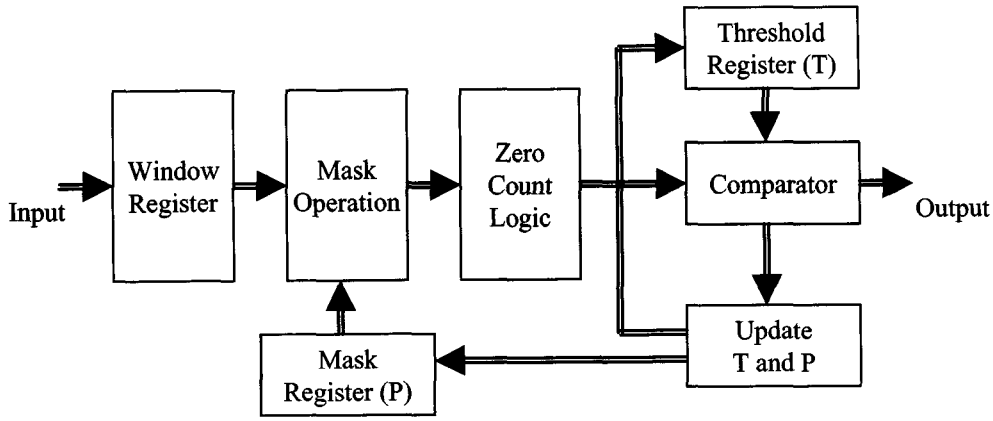
$$x^{(r)}=\sum_{i=1}^{L} f(R_i; P_i, T_i)2^{L-i} \quad (3)$$

where $r=\frac{m+1}{2}$, $x^{(r)}$ is the median.

Based on the above algorithm, we can implement the L-th level pipeline real-time median filter.

3.2 A structure of the Processing Element

Base upon the above algorithm, the structure of the basic processing element (PE) is quite obvious. The PE should be able to process one bit-slice of the window data within a single cycle. The block diagram of PE is shown in [Fig. 3].



[Fig. 3] The block diagram of the PE

When the maximum size of the window is w and the maximum number of bits per pixel is L, the size of the window register must be at least $w \times L$ bits. Because we need to access the bit-slice as well as the word, those provision should be considered to design a window register. The mask operation block is a w-bit latch with 2-input AND gate for each input. The mask register (denoted by P in above algorithm) is a w-bit parallel input and parallel output register and its update logic can be designed with 1's complements, selectors, and AND gates. The threshold register (denoted by T in the above algorithm) is also a w-bit parallel input and parallel output register and its update logic can be built with a w-bit subtractor and selector. The comparator does a w-bit comparison with one bit output. The zero count logic counts the number of zeros for the masked bit-slice of the input data, which can be built with a counter or some combinational logics.

4. Analysis and Simulation

In median filter, comparison is the key operation which determines the performance. <Table 1> shows the comparison result for the number of comparison for various implementations. The numbers shown in the <Table 1> are obtained for the 512 ×512 image and each pixel has 8 bits (256 gray levels). From this table, we can say that the presented implementation has much smaller hardware complexity than other implementations, specially for the large window.

<Table 1> Average number of Comparison

window size	non-separable (quick-sort)	separable (quick-sort)	separable (bit-slice pipeline median search)
3 ×3	10380902	3460301	1572864
5 ×5	42178969	8441037	2621440
7 ×7	99981722	14260634	3670016
9 ×9	186620314	20761805	4718592

For the analysis of the architecture and algorithm, I designed a behavioral simulator with C-language. The simulator takes input from the input image file which can be generated by the other program. I tested the architecture and algorithm using three images. The subjective result is very good as you can see in [Fig. 4] through [Fig. 6].

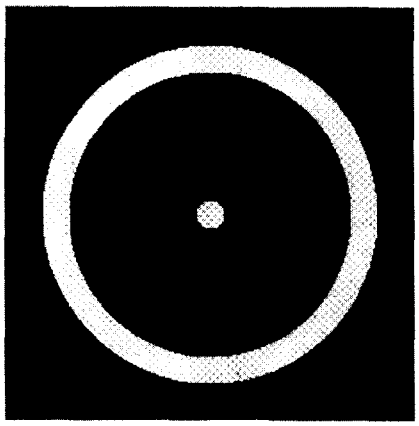
Random salt and pepper noises are added to the original images. The noise added images are filtered with the simulator and compared with the original images. All the test images are 8-bit 256×256 images. And 3×3 windows are used for the simulation.

(a) Original image

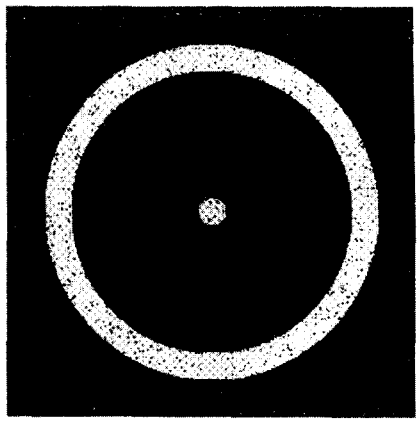
(b) Noise added image

(c) Filtered image

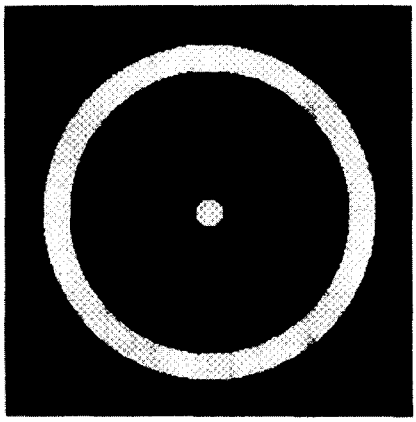
[Fig. 4] Test image 1



(a) Original image

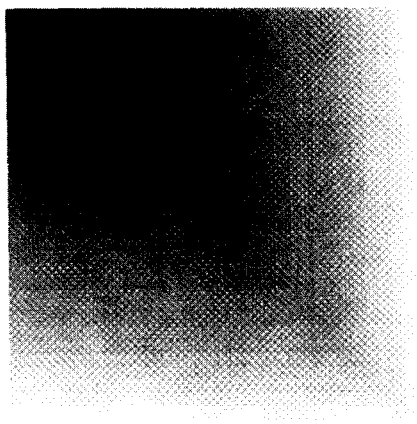


(b) Noise added image

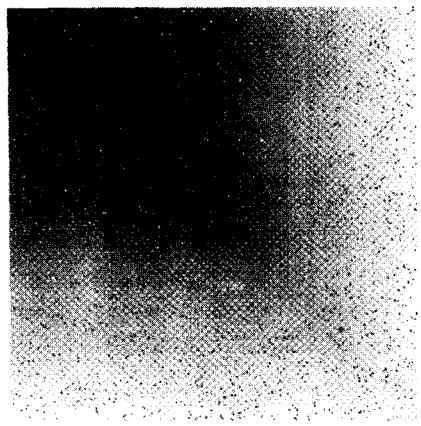


(c) Filtered image

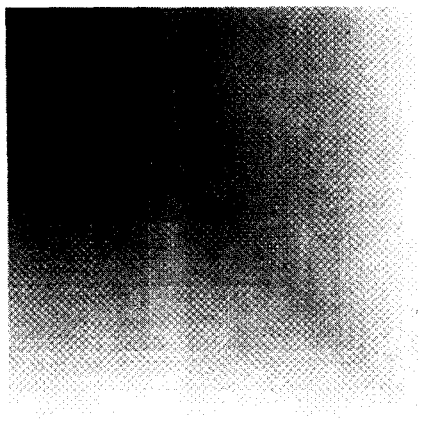
[Fig. 5] Test image 2



(a) Original image



(b) Noise added image



(c) Filtered image

[Fig. 6] Test image 3

5. Conclusion

Even though the output of a separable median filter is different from the output of the non-separable two-dimensional median filter, its performance in image noise smoothing is close to the original. Because of its separable nature, it is computationally more economical than the equivalent size original 2-D median filter, and provides an easily implementable (real-time hardware) structure for smoothing spiky image noise.

Because of pipelinable properties and very simple computation of the separable median filter with an algorithm for the order statistic determination, it is possible to implement real-time VLSI

two-dimensional median filter. The presented real-time two-dimensional median filter is better choice for the VLSI implementation.

Nonlinear algorithms in 2-D digital signal processing will become more important. Our approach starts from algorithm analysis, continues with algorithm decomposition to obtain intrinsic parallelisms in the algorithm. From this, we can find a good VLSI system architecture. The same approach can be applied to various multidimensional nonlinear signal processing algorithms.

※ References

[1] J. W. Turkey, *Exploratory data analysis.* Reading, MA: Addison-Wesley, 1971.

[2] G. R. Arce, N. C. Gallagher, and T. A. Nodes, "Median filters: Theory for one- and two-dimensional filters," Chapter 3, *Advanced in computer vision and image processing*, vol. 2, pp. 89-166.

[3] P. M. Narendra, "A separable median filter for image noise smoothing," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-3, pp.20-29, Jan. 1981.

[4] K. Chen, "Bit-serial realizations of a class of nonlinear filters based on positive boolean functions," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 785-794, Jun. 1989.

[5] R. T. Hoctor and S. A. Kassam, "An algorithm and a pipelined architecture for order statistic determination and L-filtering," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 344-352, Mar. 1989.

[6] D. J. Delman, " Digital pipelined hardware median filter design for real-time image processing, " *Proc. SPIE Conf. Real-Time Signal Processing IV*, SPIE vol. 298, pp. 184-188, 1981.

[7] P. D. Wendt, E. J. Coyle, and N. C. Gallagher, *"Stack filter," IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 898-911, Aug. 1986.

[8] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Median filtering by threshold decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 1183-1188, Dec. 1984.

[9] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Threshold decomposition of multidimensional ranked order operations," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 445-450, May. 1985.

[10] T. S. Huang, G. J. Yang, and G. Y. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 13-18, Feb. 1979.

[11] E. Ataman, V. K. Aatre, and K. S. Rao, " A fast method for real-time median filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 415-421, Aug. 1980.

[12] V. V. B. Rao and K. S. Rao, "A new algorithm for real-time median filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1674-1675, Dec. 1986.

[13] D. E. Knuth, *The art of computer programming*, vol. 3, New York: Addison-Wesley, pp. 220-246, 1973.

[14] A. L. Fisher, "Systolic algorithms for running order statistics in signal and image processing," in *Proc. CMU Conf. VLSI systems and computations*, H. T. Kung, R. F. Sproull, and G. L. Steele jr., Eds. Pittsburgh, PA: Comput. Sci. Press, Oct. 1981.

정 재 길

1980년 2월 한양대학교 전자공학과 졸업

1987년 North Carolina State U. 컴퓨터공학 석사

1991년 North Carolina State U. 컴퓨터공학 박사

1979년12월~1985년6월 국방과학연구소

1991년8월~1992년 8월 한국전자통신연구소

1992년 9월~ 현재 배재대학교 정보통신공학부 부교수