ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 20-02-20
136

j.inst.Korean.electr.electron.eng.Vol.24,No.2,507~513,june 2020
http://dx.doi.org/10.7471/ikeee.2020.24.2.507

16비트 깊이 이상의 이미지에서의 중간값 필터 계산

Computing Median Filter for over 16-bit Depth Images

김 진 욱*★

Jin Wook Kim*★

Abstract

The median filter that is used in various fields requiring image processing converts to a median value of pixels belonging to a radius r for all pixels in the image of $n \times m$ size. For 8-bit depth images, an $O(nm)$ time algorithm exists but for over 16-bit depth images, there is an $O(nm\log^2 r)$ time algorithm of Gil and Werman. In this paper, we propose an efficient median filter algorithm that works for more than 16-bit depth images. The time complexity of our algorithm is the same as that of Gil and Werman, but theoretical analysis and experimental results show that ours is efficient than above two times.

요 약

중간값 필터는 $n \times m$ 크기의 이미지와 반경 r이 주어지면 이미지의 모든 픽셀에 대해 반경 r에 속하는 픽셀들의 중간값으로 변환하는 것으로, 이미지 처리가 필요한 다양한 분야에서 활용되고 있다. 픽셀의 정보가 8비트인 경우에는 $O(nm)$ 시간 알고리즘이 존재하나 16비트 이상에는 적용이 어렵고 대신 Gil과 Werman의 $O(nm\log^2 r)$ 시간 알고리즘이 존재한다. 본 논문에서는 16비트 깊이 이상의 정보를 갖는 이미지에 대해서도 효율적으로 동작하는 중간값 필터 알고리즘을 제안한다. 시간 복잡도는 $O(nm\log^2 r)$로 Gil과 Werman의 알고리즘과 동일하지만 엄밀한 분석과 실험을 통해 2배 이상 향상됨을 보인다.

Key words : median filter, image filter, Gil-Werman algorithm, BST, HDR

Ⅰ. 서론

중간값 필터는 다양한 이미지 필터의 한가지로 $n \times m$ 크기의 이미지와 반경 r이 주어지면 이미지의 모든 픽셀에 대해 해당 픽셀을 그 픽셀을 중심으로 반경 r에 속하는 픽셀들의 중간값으로 변환한다[1]. 중간값 필터는 마스킹, 잡음 제거, 형태학 필터, KESM 등 이미지 처리가 필요한 다양한 분야에서 활용되고 있다[2-5].

이미지의 각 픽셀이 가질 수 있는 정보의 양이 8비트인 경우 8비트 깊이라고 하며, 8비트 깊이 정보를 갖는 이미지에 대한 중간값 필터는 다양한 알고리즘들이 개발되어 있다. Huang[6]은 $O(nmr)$ 시간, Weiss[7]는 $O(nm\log r)$ 시간, Perreault-Hebert[8]와 Alekseychuk[9]은 각각 $O(nm)$ 시간 알고리즘을 제안하였다. 이들은 $2^8 = 256$ 단계의 히스토그램

* Dept. of Computer Science, Korea National Open University
★ Corresponding author
E-mail : gnugi@knou.ac.kr, Tel : +82-2-3668-4652
※ Acknowledgment
This research was supported by Korea National Open University Research Fund
Manuscript received Jun. 1, 2020; revised Jun. 23, 2020; accepted Jun. 24, 2020.

을 사용하여 같은 정보를 갖는 픽셀 수를 활용하여 중간값을 찾는다.

최근에는 의료영상이나 HDR(High Dynamic Range) 이미지 등에서 8비트 깊이를 넘어 16비트, 32비트, 심지어 부동소숫점을 이용하는 이미지들이 사용된다[10, 11]. 하지만 히스토그램에 기반을 둔 알고리즘들은 히스토그램 크기를 상수로 간주하기에 이미지의 깊이가 깊어질수록 상수가 커져 수행시간이 길어진다. 또한 히스토그램 크기의 저장 공간을 사용하기에 물리 메모리를 벗어나게 되어, [9]는 16비트 깊이까지는 적용이 가능하나 32비트는 적용이 어렵다. 또한 히스토그램 방식은 셀 수 없는 부동소숫점 이미지에는 적용할 수가 없다.

16비트 깊이 이상의 정보를 갖는 이미지에 적용할 수 있는 중간값 필터 알고리즘으로는 Quickselect [12]를 이용한 $O(nmr^2)$ 시간 알고리즘과 Gil과 Werman [13]의 $O(nm\log^2 r)$ 시간 알고리즘이 있다. 본 논문에서는 Gil과 Werman의 알고리즘을 개선하여 16비트 이상의 이미지에서도 효율적으로 동작하는 중간값 필터 알고리즘을 제안한다. Gil과 Werman의 알고리즘은 두 단계의 이진 탐색 트리를 사용하는데 그 중 한 단계의 탐색 트리를 개선함으로써 두 배 이상의 속도 향상을 얻게 된다.

본 논문의 구성은 다음과 같다. 2장에서는 중간값 필터와 관련된 용어 정의와 관련 연구인 Gil-Werman 알고리즘을 소개한다. 3장에서는 Gil-Werman 알고리즘을 개선한 알고리즘을 제안하고, 4장과 5장에서는 이에 대한 이론적 분석과 실험적 분석을 제시한 뒤, 5장에서 결론을 맺는다.

II. 관련 연구

1. 용어 정의

크기가 $n\times m$인 2차원 배열 F는 세로로 n개의 픽셀, 가로로 m개의 픽셀로 이뤄진 이미지를 나타낸다. 배열 F의 원소는 실수로, $F[i][j]$로 표현하고 이는 i행 j열에 위치한 픽셀의 정보를 나타낸다(단, $0\le i\le n,\ 0\le j\le m$). 부분배열 $F[i_1..i_2][j_1..j_2]$는 좌측상단이 i_1행 j_1열이고 우측하단이 i_2행 j_2열인 직사각형 모양에 포함되는 배열 F의 원소들을 나타낸다.

반경 r이 주어진 경우 i행 j열의 중간값(median) 원소는 $B=(2r+1)^2$개의 원소인 $F[i-r..i+r][j-r..j+r]$에 대해 $(B-1)/2$개의 원소보다 크거나 같고 $(B-1)/2$개의 원소보다 작거나 같은 원소로 정의된다. 중간값 원소를 찾는 가장 단순한 방법으로 $O(B\log B)$ 시간에 B개의 원소를 정렬하면 $\lceil B/2\rceil$번째 원소가 중간값 원소가 된다. 보다 효율적인 방법은 Quickselect[12]를 이용하여 $O(B)$ 시간에 $\lceil B/2\rceil$번째 원소를 찾는 것이다.

반경 r인 중간값 필터(median filter)는 $0\le i\le n$, $0\le j\le m$인 모든 i, j에 대해 i행 j열의 반경 r에 대한 중간값 원소를 찾는 작업이다. 이때 좌측상단이 $i-r$행 $j-r$열이고 우측하단이 $i+r$행 $j+r$열인 정사각형을 반경 r에 대한 i행 j열의 필터범위라고 부르자. 반경 r인 중간값 필터는 $n\times m$ 크기의 모든 위치마다 필터범위에 대해 Quickselect를 이용하여 $O(nmB)$ 시간에 구할 수 있다.

2. Gil-Werman 알고리즘[13]

Gil과 Werman의 중간값 필터 알고리즘(이하 Gil-Werman 알고리즘)은 이진 탐색 트리를 두 단계로 사용하여 반경 r인 중간값 필터를 $O(nm\log^2 B)$ 시간에 구한다. 이는 기본적으로 인접한 위치의 필터범위에는 중복된 원소가 많다는 사실에 기반을 둔다.

Gil-Werman 알고리즘에서 사용하는 이진 탐색 트리인 IOBBST(Implicit Offline Balanced Binary Search Tree)는 완전 이진 트리로 다음 두 가지 연산을 트리 높이에 비례한 시간에 각각 수행할 수 있다. 하나는 주어진 집합의 k번째 크기의 원소를 찾는 것이고 다른 하나는 원소의 값의 범위가 주어지면 범위에 속하는 주어진 집합의 원소의 개수를 구하는 것이다. 이를 위해 각 노드는 표 1과 같이 세 개의 필드로 구성된다. 필드 중 active는 해당 노드가 주어진 집합에 속하는 경우 참을 값으로 갖는다.

Table 1. a node specification for an IOBBST
표 1. IOBBST 노드의 구성요소

Field	Type	Value
x	real or integer	the key associated with the current node
active	boolean	true whenever the current node is within a given set
count	integer	the number of active nodes in the subtree rooted at the current node

Gil-Werman 알고리즘은 $(2r+1)\times(2r+1)$개의 중간값 원소를 구하기 위해 이들의 필터범위에 해당하는 $(4r+1)\times(4r+1)$ 크기의 부분배열을 이용한다. 우선 $(4r+1)^2$개의 원소로 IOBBST T를 만들고, T의 각 노드 α마다 IOBBST R_α를 만든다. R_α는 노드 α를 루트로 갖는 T의 부분트리 T_α에 대해 T_α의 각 노드의 행 값을 키로 둔 IOBBST로 R_α의 노드의 개수는 T_α의 노드의 개수와 같다.

중간값 원소는 열 별로 구하는데, 이를 위해 각 노드의 active 값과 행 값을 이용한다. 첫 번째 열의 $2r+1$개 중간값을 구하기 위해, 먼저 필터범위에 해당되는 $2r+1$개의 열에 존재하는 $(4r+1)(2r+1)$개 원소와 대응되는 T와 R_α의 모든 노드의 active 값을 참으로 두고, 연관된 모든 노드의 count도 증가시킨다. 이후 위치별로 필터범위에 해당되는 양 끝 행 값을 이용하여 $\lceil B/2 \rceil$번째 원소를 찾는다.

다음 열의 $2r+1$개 중간값을 구하기 위해서는, 현재 active인 노드들 중 필터범위를 벗어나는 가장 왼쪽 열의 $4r+1$개 원소와 대응되는 T와 R_α의 모든 노드의 active 값을 거짓으로 바꾸고 새로 필터범위에 들어온 한 열의 $4r+1$개 원소와 대응되는 T와 R_α의 모든 노드의 active 값을 참으로 바꾼다. 이후 동일한 방법으로 위치별로 필터범위에 해당되는 양 끝 행 값을 이용하여 $\lceil B/2 \rceil$번째 원소를 찾는다.

$B=(2r+1)^2$개의 중간값 원소를 구하기 위해 Gil-Werman 알고리즘은 T와 R_α 생성에 $O(B\log B)$ 시간, active 값 변경에 $O(B\log^2 B)$ 시간, 중간값 원소 탐색에 $O(B\log^2 B)$ 시간이 소요된다.

III. 제안 알고리즘

Gil-Werman 알고리즘의 두 단계 IOBBST 중 노드별 IOBBST인 R_α에서 효율적이지 못한 부분을 찾고 이를 개선한 알고리즘을 제시한다.

1. IOBBST R_α의 문제점

R_α에는 중복 노드가 존재한다. R_α의 노드의 개수는 T_α의 노드의 개수와 동일하므로 α를 T의 루트 노드로 가정하면 R_α는 $(4r+1)^2$개의 노드로 구성된다. 이때 R_α의 노드는 행 값을 키로 가지므로 결국 R_α에는 $4r+1$개의 서로 다른 키가 존재하고 각 키는 $4r+1$개씩 존재한다.

Gil-Werman 알고리즘에서 다음 열의 중간값 계산을 위해서는 active 값 변경이 필요한데, 하나의 행 값을 키로 갖는 노드가 여러 개 존재하고 또한 active 값이 참인 노드와 거짓인 노드가 모두 존재한다. 예를 들어, 임의의 노드 α에 대한 R_α가 그림 1과 같다면 3행을 키로 갖는 노드 하나의 active를 참으로 변경하기 위해서는 탐색 및 업데이트를 위한 효율적인 정책이 필요하다.

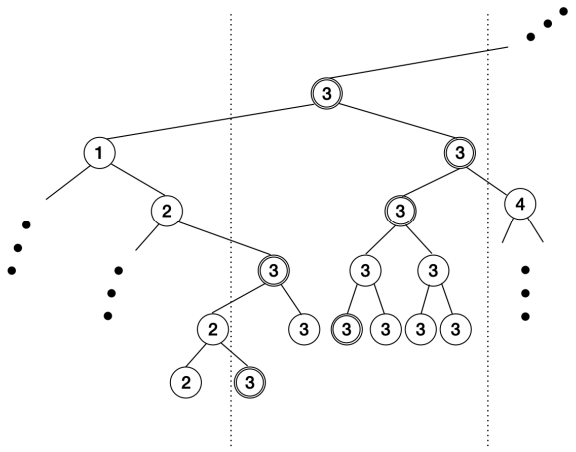

Fig. 1. An example of R_α. The value in a node is a row index and the double circle means that the value of active field is true.

그림 1. R_α 예시. 노드 안의 숫자는 행 값이며 테두리가 두 겹인 노드는 active 값이 참인 노드임

2. rowIOBBST ER_α

IOBBST R_α를 대신할 수 있는 rowIOBBST ER_α를 제안한다. rowIOBBST는 IOBBST와 기본적인 특성은 동일하다. 즉, 완전 이진 트리로 주어진 집합의 k번째 크기의 원소를 찾는 것과 원소의 값의 범위가 주어지면 범위에 속하는 주어진 집합의 원소의 개수를 구하는 것을 모두 트리 높이에 비례한 시간에 각각 수행할 수 있다.

Table 2. a node specification for an rowIOBBST.

표 2. rowIOBBST 노드의 구성요소

Field	Type	Value
x	integer	the index of row (key)
activecount	integer	the number of times that the key value of the current node is within a given set
count	integer	the sum of activecount of nodes in the subtree rooted at the current node

대신 노드의 구성이 표 2와 같다. 필드 x는 노드의 키로 행 값을 나타낸다. 필드 activecount는 해당 노드의 키 값이 주어진 집합에 속하는 횟수를 값으로 갖는다. 필드 count는 현재 노드를 루트로 하는 부분트리에 속하는 모든 노드의 activecount의 합이다.

3. 개선된 중간값 필터 알고리즘

크기가 $n\times m$인 2차원 배열 F와 반경 r이 주어질 때 중간값 필터 알고리즘은 다음과 같다.

```
1 for row=0 to n by 2r+1
2   for col=0 to m by 2r+1
3     Initialize T and ER_α using
          F[row-r..row+3r][col-r..col+3r]
4     Activate the elements in columns
          from col-r to col+r-1
5     for j=0 to 2r
6       Activate the elements in column
            col+j+r
7       for i=0 to 2r
8         Find the median between row
              row+i-r and row+i+r
9       Deactivate the elements in column
            col+j-r
```

알고리즘의 1행과 2행은 주어진 이미지를 $(2r+1)\times(2r+1)$ 크기의 구역으로 나누어 각 구역별로 중간값 원소를 구할 수 있도록 한다.

3행에서 한 구역의 모든 필터범위가 포함되는 $(4r+1)\times(4r+1)$ 크기의 F의 부분배열에 대해 IOBBST T를 만들고, T의 각 노드 α마다 rowIOBBST ER_α를 만든다. ER_α는 노드 α를 루트로 갖는 T의 부분트리 T_α에 대해 T_α의 각 노드의 서로 다른 행 값을 키로 둔 rowIOBBST로 ER_α의 노드의 개수는 T_α에 존재하는 서로 다른 행 값의 개수와 같다. ER_α의 각 노드의 activecount는 0을 초깃값으로 갖는다.

4행과 6행에서 각 원소와 대응되는 T의 모든 노드의 active 값을 참으로 두고 연관된 모든 노드의 count도 증가시키고, 역시 대응되는 ER_α의 모든 노드의 activecount를 1 증가시키고 연관된 모든 노드의 count도 증가시킨다. 9행에서는 반대로 T의 active 값은 거짓으로 두고 연관된 count는 감소시키며, ER_α의 activecount는 1 감소시키고 연관된 모든 모드의 count도 감소시킨다.

8행에서는 T의 루트노드의 ER_α에 대해 필터범위에 속하는 원소의 개수를 구해 중간값 원소의 순위를 구한 뒤, 중간값 원소를 찾을 때까지 자식노드의 ER_α에 대해 역시 필터범위에 속하는 원소의 개수를 구하여 중간값 원소 순위가 존재하는 자식으로 내려간다.

Ⅳ. 이론적 분석

제안 알고리즘의 IOBBST T는 Gil-Werman 알고리즘의 T와 동일하다. 따라서 B 크기의 한 구역에 대해서 T 생성에 $O(B\log B)$ 시간, T의 active 값 변경에 $O(B\log B)$ 시간, 중간값 원소 탐색에서 T 자체에서의 탐색에 $O(B\log B)$ 시간이 소요되며 이는 Gil-Werman 알고리즘에서도 마찬가지이다.

제안 알고리즘의 rowIOBBST ER_α는 최대 $4r+1$개의 서로 다른 행 값을 키로 갖는 노드로 구성되므로 노드 개수는 $O(r)$, 트리 높이는 $\log(4r+1)=O(\log r)$이다. 따라서 Gil-Werman 알고리즘의 IOBBST R_α의 생성 시간이 $O(B\log B)$임에 반해 ER_α은 $O(B\log r)$이 된다.

한 ER_α의 activecount 값 변경과 주어진 범위에 속하는 원소의 개수 구하기는 모두 트리 높이에 비례하므로 모두 $O(\log r)$ 시간에 가능하다. 따라서 한 구역에서 모든 ER_α의 activecount 값 변경에 $O(B\log B\log r)$ 시간, 중간값 원소 탐색에 $O(B\log B\log r)$ 시간이 소요된다.

표 3은 한 구역에 대한 제안 알고리즘과 Gil-Werman 알고리즘의 이론적 분석 결과를 나타낸다. 따라서 제안 알고리즘은 크기가 $n\times m$인 2차원 배열 F와 반경 r이 주어지면 중간값 필터를 $O(nm\log B\log r)$ 시간에 구한다.

Table 3. Analysis for proposed algorithm and Gil-Werman algorithm

표 3. 제안 알고리즘과 Gil-Werman 알고리즘의 분석 결과

Part	Proposed	Gil-Werman
Initialization	$O(B\log B)$	$O(B\log B)$
Activation/ Deactivation	$O(B\log B\log r)$	$O(B\log^2 B)$
Search	$O(B\log B\log r)$	$O(B\log^2 B)$
Total	$O(B\log B\log r)$	$O(B\log^2 B)$

V. 실험적 분석

실험환경은 2.6 GHz Intel Core i7 프로세서에 16 GB 메모리가 설치된 하드웨어에 운영체제는 macOS Catalina이며, 컴파일러는 Apple clang version 11.0.3을 이용하였다.

실험을 위해 제안 알고리즘, Gil-Werman 알고리즘, Quickselect 알고리즘의 세 가지 중간값 필터를 구현하였다. 입력 이미지는 32비트 부동소숫점 형식으로 저장된 TIFF 파일을 이용하였다.

1. 637×502 크기 이미지

그림 2는 637×502 크기의 HDR 이미지[1]를 gray로 변환한 것으로, 각 픽셀의 값은 0.082062부터 1178.623047의 범위를 갖는다. 참고로 픽셀 값을 모두 0.1배하면 그림 3과 같이 보인다.

Fig. 2. Image 1 with size 637×502.
그림 2. 637×502 크기의 이미지 1

Fig. 3. Dividing pixel values of image 1 by 10.
그림 3. 이미지 1의 픽셀 값을 10으로 나눈 이미지.

1) 이미지 출처: https://www.cheshireeng.com/HDReye/Downloads.html

Table 4. Time versus radius size r for three algorithms about image 1.
표 4. 이미지 1에 대한 실험 결과(r은 반경)

r	Proposed	Gil-Werman	Quickselect
5	0.552925	1.177847	0.557932
10	0.794170	1.797830	0.923762
15	0.960467	2.386502	1.834006
20	1.105257	2.897815	3.068162
25	1.235140	3.214367	4.489842
30	1.359997	3.609070	6.222955
35	1.584560	3.908567	8.340539
40	1.626399	4.234857	10.284576
45	1.664384	4.583551	12.570645
50	1.718059	4.712529	15.126373
55	1.853419	5.128850	17.890650
60	1.936914	5.406486	20.815172
65	1.913256	5.463299	23.771859
70	2.057917	6.047436	26.892942
75	2.221242	6.167189	30.445829
80	2.281489	6.326390	33.828120
85	2.266245	6.379532	36.985963
90	2.225984	6.899193	40.935265
95	2.332681	7.250396	44.817098
100	2.446180	7.485989	48.698146

표 4는 이미지 1에 대한 세 알고리즘의 수행시간을 비교한 표이다. 이론적 시간이 $O(nmB)$인 Quickselect 알고리즘이 가장 느리나 반경 15 이하에서는 $O(nm\log^2 B)$인 Gil-Werman 알고리즘보다 빨랐다. 이론적 시간이 $O(nm\log B\log r)$인 제안 알고리즘이 가장 빨랐으며, Gil-Werman 알고리즘에 비해 반경 5일 때 2.13배, 반경 100일 때 3.06배 빠른 결과를 보였다. 그림 4는 표 3을 그래프로 나타낸 것이다.

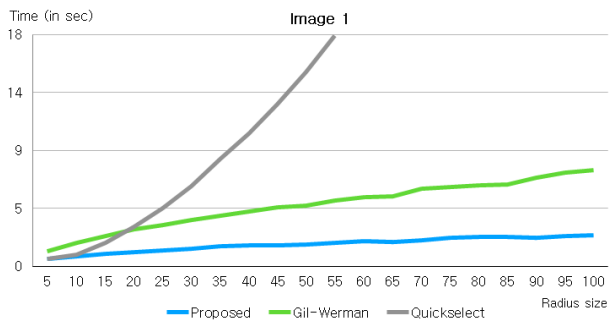

Fig. 4. Time versus radius size for three algorithms about image 1.
그림 4. 이미지 1에 대한 실험 결과. x축은 반경, y축은 수행시간(초)

2. 1008×1344 크기 이미지

그림 5는 1008×1344 크기의 raw 이미지 파일을 Photoshop을 이용하여 32비트 부동소숫점 형식의 gray로 변환한 것으로, 각 픽셀의 값은 0.0부터 0.988799의 범위를 갖는다.

Fig. 5. Image 2 with size 1008×1344.
그림 5. 1008×1344 크기의 이미지 2

Table 5. Time versus radius size r for three algorithms about image 2.
표 5. 이미지 2에 대한 실험 결과(r은 반경)

r	Proposed	Gil-Werman	Quickselect
5	2.375530	4.926497	1.549830
10	3.444059	7.593296	5.512216
15	4.206734	10.138424	11.248050
20	4.881427	12.159454	19.366421
25	5.521458	13.901507	28.788376
30	6.196764	15.610801	39.510149
35	7.092025	16.965451	54.183782
40	7.605752	18.897915	68.667804
45	7.773793	20.304711	84.979750
50	8.099880	21.666970	104.001633
55	8.805219	23.533986	119.721022
60	9.251733	25.629238	140.881127
65	9.623039	26.656616	162.818541
70	10.124270	28.238064	185.820979
75	10.932207	29.159982	210.004372
80	11.359914	30.964569	235.650264
85	11.170300	31.604666	262.233712
90	11.503231	34.229707	290.202158
95	11.621914	34.564918	319.129603
100	12.060254	35.813949	348.908511

표 5는 이미지 2에 대한 세 알고리즘의 수행시간을 비교한 표이다. Quickselect 알고리즘이 반경 5에서는 가장 빨랐지만 이후 급격히 느려졌고, Gil-Werman 알고리즘은 반경 15 이상에서 Quickselect보다 빨랐다. 제안 알고리즘은 Gil-Werman 알고리즘에 비해 반경 5일 때 2.07배, 반경 100일 때 2.97배 빠른 결과를 보였다. 그림 6은 표 4를 그래프로 나타낸 것이다.

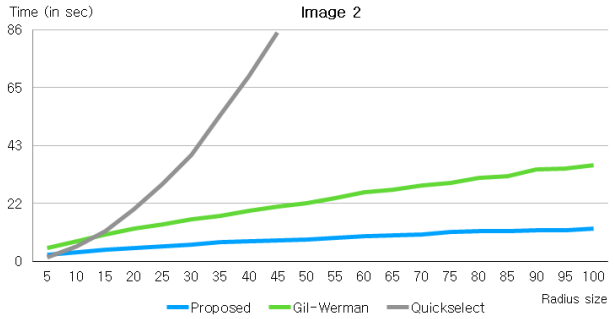

Fig. 6. Time versus radius size for three algorithms about image 2.
그림 6. 이미지 2에 대한 실험 결과. x축은 반경, y축은 수행시간(초)

Ⅵ. 결론

본 논문에서는 각 픽셀이 16비트 깊이 이상의 정보를 갖는 이미지에서 효율적으로 동작하는 중간값 필터 알고리즘을 제시하였다. 제안 알고리즘은 Gil-Werman 알고리즘의 IOBBST를 개선하여 이론적으로나 실험적으로 모두 향상된 결과를 얻었다.

다만 반경이 작은 경우에는 이론적으로 느린 Quickselect 알고리즘보다 실험적으로 느린 결과를 얻기도 하여, 작은 반경에 대해 속도를 높일 수 있는 향후 연구가 필요하다.

References

[1] J. Tukey, *Exploratory Data Analysis*, Addison-Wesley Menlo Park, 1977.

[2] P. Maragos and R. Schafer, "Morphological Filters-Part II: Their Relations to Median, Order-Statistic Filters," *IEEE Trans. on Acoustics, Speech, Signal Processing*, vol.35, no.8, pp.1170-1184, 1987. DOI: 10.1109/TASSP.1987.1165254

[3] D. Mayerich, B. H. McCormick, J. Keyser,

"Noise and Artifact Removal in Knife-Edge Scanning Microscopy," *ISBI*, pp.556-559, 2007.
DOI: 10.1109/ISBI.2007.356912
[4] Y. Choe, L. C. Abbott, D. Han, P. S. Huang, J. Keyser, J. Kwon, D. Mayerich, Z. Melek, and B. H. McCormick, "Knife-Edge Scanning Microscopy: High-Throughput Imaging and Analysis of Massive Volumnes of Biological Microstructures," *High-Throughput Image Reconstruction and Analysis: Intelligent Microscopy Applications*, pp.11-37, 2008.
[5] J. Lee, W. An, and Y. Choe, "Mapping the Full Vascular Network in the Mouse Brain at Submicrometer Resolution," *39th Annual International Conference of the IEEE EMBC*, 2017.
DOI: 10.1109/EMBC.2017.8037564
[6] T. Huang, G. Yang, and G. Tang, "A Fast Two-Dimensional Median Filtering Algorithm," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol.27, pp.13-18, 1979.
DOI: 10.1109/TASSP.1979.1163188
[7] B. Weiss, "Fast Median and Bilateral Filtering," *ACM Trans. Graph.*, vol.25, pp.519-526, 2006.
DOI: 10.1145/1179352.1141918
[8] S. Perreault and Patrick Hébert, "Median Filtering in Constant Time," *IEEE Trans. on Image Processing*, vol.16, no.9, pp.2389-2394, 2007. DOI: 10.1109/TIP.2007.902329
[9] A. Alekseychuk, "Hierarchical recursive running median," *19th IEEE International Conference on Image Processing*, pp.109-112, 2012.
DOI: 10.1109/ICIP.2012.6466807
[10] TC. Lu, CY. Tseng, CC. Huang, KM. Deng, "16-Bit DICOM Medical Images Lossless Hiding Scheme Based on Edge Sensing Prediction Mechanism," in *Genetic and Evolutionary Computing. Advances in Intelligent Systems and Computing*, vol.329. Springer, Cham, 2015.
DOI: 10.1007/978-3-319-12286-1_19
[11] A. Darmont, *High Dynamic Range Imaging: Sensors and Architectures*, SPIE press, 2013.
[12] R. W. Floyd and R. L. Rivest, "Algorithm 489: the Algorithm SELECT - for Finding the i-th smallest of n elements [M1]," *Communications of the ACM*, vol.18, pp.173, 1975.
DOI: 10.1145/360680.360694
[13] J. Gil and M. Werman, "Computing 2-d Min, Median, and Max Filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol.15, no.5, pp.504-507, 1993.
DOI: 10.1109/34.211471

BIOGRAPHY

Jin Wook Kim (Member)

1998 : BS degree in Mathematics, Seoul National University.
2000 : MS degree in Computer Engineering, Seoul National University.
2006 : PhD degree in Computer Engineering, Seoul National University.
2010~2013 : Research Professor, Seoul National University Hospital.
2013~ : Professor, Department of Computer Science, Korea National Open University.