ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 22-02-13
100

j.inst.Korean.electr.electron.eng.Vol.26,No.2,240~248,June 2022
https://doi.org/10.7471/ikeee.2022.26.2.240

부동소수점 형식 이미지를 위한 효율적인 중간값 필터 알고리즘

An Efficient Median Filter Algorithm for Floating-point Images

김 진 욱*★

Jin Wook Kim*★

Abstract

Floating-point images that express pixel information as real numbers are used in HDR images. There have been various researches on efficient median filter algorithms, but most of them are applicable to 8-bit depth images and there are only a few number of algorithms applicable to floating-point images, including Gil and Werman's algorithm. In this paper, we propose a median filter algorithm that works efficiently on floating-point images by improving Kim's algorithm, which improved Gil and Werman's algorithm. Experimental results show that the execution time is improved by about 10% compared to the Kim's algorithm by reducing the redundant work for the repetitively used binary search tree and applying the inverted index.

요 약

픽셀의 정보를 실숫값으로 표현하는 부동소수점 형식 이미지는 HDR 이미지 등에서 사용된다. 효율적인 중간값 필터 알고리즘에 관한 연구는 다양하게 이뤄졌지만 대부분 8비트 깊이 이하의 이미지에 적용할 수 있고 부동소수점 형식 이미지에 적용할 수 있는 알고리즘은 Gil과 Werman의 알고리즘을 비롯하여 제한적으로만 존재한다. 본 논문에서는 Gil과 Werman의 알고리즘을 개선한 Kim의 알고리즘을 다시 개선하여 부동소수점 형식 이미지에 대해 효율적으로 동작하는 중간값 필터 알고리즘을 제안한다. 반복적으로 사용되는 이진 탐색 트리에 대한 중복 작업을 줄이고 역인덱스를 적용하여 실험 결과 Kim 알고리즘보다 약 10% 수행시간이 향상됨을 보인다.

Key words : median filter, floating-point image, HDR, BST, inverted index

* Professor, 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. 6, 2022; revised Jun. 21, 2022; accepted Jun. 23, 2022.

Ⅰ. 서론

부동소수점 형식 이미지는 각 픽셀이 갖는 정보를 실숫값으로 표현하는 이미지로 HDR(High Dynamic Range) 이미지에서 주로 사용된다[1]. HDR 이미지는 실세계를 더 정확하게 표현하는데 부합하여 도로교통 모니터링이나 군사적 응용 분야 등 점점 더 많은 곳에 사용되고 있다[2, 3].

이미지에서의 중간값 필터(median filter)는 각 위치에서 필터 범위에 속하는 픽셀들의 정보 중에서 중간값을 찾아 해당 위치의 픽셀 정보로 변환해 준다[4]. 중간

값 필터는 salt & pepper 잡음 제거, 마스킹, 형태학 필터, KESM (Knife-Edge Scanning Microscopy) 등 다양한 분야에서의 이미지 처리에 활용된다[4-8].

이미지에 중간값 필터를 효율적으로 적용하기 위한 다양한 연구가 진행됐다. 각 픽셀의 정보의 양이 8비트를 넘지 않는 8비트 깊이 이하의 이미지에 대해서는 히스토그램 방식의 알고리즘들이 개발되었다. 이미지 크기가 $n\times m$이고 필터 크기가 $(2r+1)\times(2r+1)$인 경우, Huang[9]은 $O(nmr)$ 시간 알고리즘, Weiss[10]는 $O(nm\log r)$ 시간 알고리즘을 제안하였고, Perreault-Hebert[11]와 Alekseychuk[12]은 각각 $O(nm)$ 시간 알고리즘을 제안하였다. 특히 Alekseychuk 알고리즘은 16비트 깊이 이미지에도 적용할 수 있으나, 32비트 깊이 이상의 이미지에서는 속도 저하가 커서 실용적이지 못하다[12]. 한편 히스토그램 방식의 중간값 필터 알고리즘을 FPGA나 GPU 등에 적용하는 연구도 진행되고 있다[13, 14].

부동소수점 형식 이미지 또는 16비트 깊이 이상의 이미지에 적용할 수 있는 중간값 필터 알고리즘으로는 QuickSelect[15]를 이용한 $O(nmr^2)$ 시간 알고리즘, 평균과 표준편차를 이용한 $O(nmr^2)$ 시간의 Binmedian 알고리즘[16], 이진 탐색 트리를 이용한 $O(nm\log^2 r)$ 시간의 Gil-Werman 알고리즘[17]이 있다. 또한, Kim 알고리즘[18]은 Gil-Werman 알고리즘의 이진 탐색 트리를 개선하여 2배 이상의 속도 향상을 얻었다.

본 논문에서는 Kim 알고리즘을 개선하여 더 효율적인 중간값 필터 알고리즘을 제안한다. Kim 알고리즘에서 필터가 이동하며 이진 탐색 트리를 업데이트할 때 반복되어 처리되는 부분을 본 논문에서는 한 번에 모아서 처리하도록 개선한다. 이를 위해 이진 탐색 트리에 대한 역인덱스를 새롭게 적용한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 용어와 Kim 알고리즘을 소개한다. 3장에서는 Kim 알고리즘을 개선한 알고리즘을 제안하고, 4장에서는 실험적으로 분석한 결과를 제시한다. 마지막 5장에서는 결론을 맺는다.

II. 관련 연구

1. 용어 정의

$n\times m$ 크기의 이미지 정보는 $n\times m$ 크기의 이차원 배열 F에 저장되며, 이때 i행 j열에 있는 픽셀의 정보(단, $0\le i<n,\ 0\le j<m$)는 부동소수점 형식으로 $F[i][j]$에 저장된다. i행 j열에서 반경 r인 필터의 범위는 $(2r+1)\times(2r+1)$ 크기의 부분 배열 $F[i-r..i+r][j-r..j+r]$로 표현한다.

반경 r인 중간값 필터를 i행 j열에 적용하면, $F[i-r..i+r][j-r..j+r]$의 $(2r+1)^2$개의 원소 중 픽셀의 정보가 $((2r+1)^2-1)/2$개의 원소보다 크거나 같고 $((2r+1)^2-1)/2$개의 원소보다 작거나 같은 원소가 중간값이 된다.

트리에서 노드 α의 레벨 $level_\alpha$는 루트에서 α까지 경로상의 노드의 개수로 둔다.

2. Kim 알고리즘[18]

Gil-Werman 알고리즘[17]을 개선한 Kim의 중간값 필터 알고리즘은 IOBBST와 rowIOBBST라는 두 종류의 이진 탐색 트리를 사용한다. IOBBST(Implicit Offline Balanced Binary Search Tree)는 각 노드가 각 픽셀의 정보를 갖는 이진 탐색 트리로, 주어진 집합에 속하는 픽셀의 노드만 활성화시키고 나머지 노드는 비활성화시킬 수 있다. rowIOBBST는 각 노드가 각 행의 정보를 갖는 이진 탐색 트리로, 주어진 집합에 대해 같은 행에 해당하는 픽셀의 개수를 해당하는 노드에 유지할 수 있다. 표 2는 rowIOBBST 노드의 구성요소를 나타낸다.

Table 1. a node specification for an rowIOBBST.
표 1. 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

그 외 이진 탐색 트리로서의 특성은 둘 다 동일하다. 집합이 주어진 상태에서, k번째 크기의 원소를 트리 높이에 비례한 시간에 찾을 수 있고 원소의 값의 범위를 주면 그 범위에 속하는 원소의 개수를 역시 트리 높이에 비례한 시간에 찾을 수 있다. 또한 완전 이진 트리로 구성하여 IOBBST와 rowIOBBST 둘 다 1차원 배열로 구현할 수 있다.

Kim 알고리즘은 $n\times m$ 크기의 이차원 배열 F와 필터의 반경 r이 주어지면 다음과 같이 동작한다. 우선 배열 F를 $(2r+1)\times(2r+1)$ 크기의 구역으로 나누고, 각 구역에 대해 해당 구역의 필터 범위가 모두 포함되는 $(4r+1)\times$

$(4r+1)$ 크기의 부분 배열을 이용하여 $IOBBST$를 만들고 이를 T로 둔다. 그리고 T의 각 노드 α에 대해 α를 루트로 갖는 T의 부분트리 T_α를 이용하여 T_α의 각 노드의 행 값으로 $rowIOBBST$를 만들고 이를 ER_α로 둔다. 즉, T의 노드의 개수는 $(4r+1)^2$개이고 ER_α의 노드의 개수는 T_α에 존재하는 서로 다른 행 값의 개수로 최대 $4r+1$개다.

다음으로 한 구역의 가장 왼쪽 열의 중간값을 모두 구하기 위해 먼저 필터 범위에 해당하는 $(4r+1)\times(2r+1)$개 픽셀을 이용하여 T와 ER_α의 노드 정보를 수정한다. 그 후 각 행의 중간값을 구하기 위해 필터 범위에 해당하는 행의 최솟값과 최댓값을 이용하여 중간값을 찾는다.

한 열의 중간값을 모두 구한 뒤 다음 열의 중간값을 모두 구하기 위해서는 바뀐 필터 범위를 T와 ER_α의 노드 정보에 반영해야 한다. 즉, 필터 범위를 벗어나는 가장 왼쪽 열의 $4r+1$개 픽셀에 대해 T의 해당 노드들을 비활성화하고 ER_α의 해당 노드들의 픽셀 개수를 감소시키며, 필터 범위에 새로 포함된 가장 오른쪽 열의 $4r+1$개 픽셀에 대해 T의 해당 노드들을 활성화하고 ER_α의 해당 노드들의 픽셀 개수를 증가시킨다. 이후 각 행의 중간값을 구하기 위해 필터 범위에 해당하는 행의 최솟값과 최댓값을 이용하여 중간값을 찾는다.

Ⅲ. 제안 알고리즘

Kim 알고리즘에서 $rowIOBBST$인 ER_α는 가장 빈번하게 사용되는 자료구조이기에 ER_α의 사용을 효율적으로 개선한 알고리즘을 제안한다. 우선 ER_α 사용에서 중복되는 부분을 설명한 후 개선된 알고리즘을 제시한다.

1. 필터 범위 이동에 따른 ER_α 업데이트

Kim 알고리즘에서 중간값을 구할 열이 바뀔 때 달라지는 필터 범위가 T와 ER_α에 반영되는 과정을 살펴보자.

그림 1은 a부터 e까지 다섯 개의 픽셀이 필터 범위에 추가(또는 제외)되는 경우 T와 ER_α에 반영되는 위치 일부를 나타낸다. 우선 T를 살펴보면 변경되는 a부터 e까지의 각 픽셀에 대응되는 다섯 노드(검은색 원)가 있고 각 노드로부터 루트까지 경로(실선)가 표시되어 있다. 즉, T에서는 이 경로상의 모든 노드가 업데이트된다.

추가로, 경로상의 모든 노드에 대해 해당 노드들의 ER_α가 업데이트된다. 예를 들어, 픽셀 e를 살펴보면 먼저 그림 1과 같이 대응되는 노드의 ER_e에서 픽셀 e의

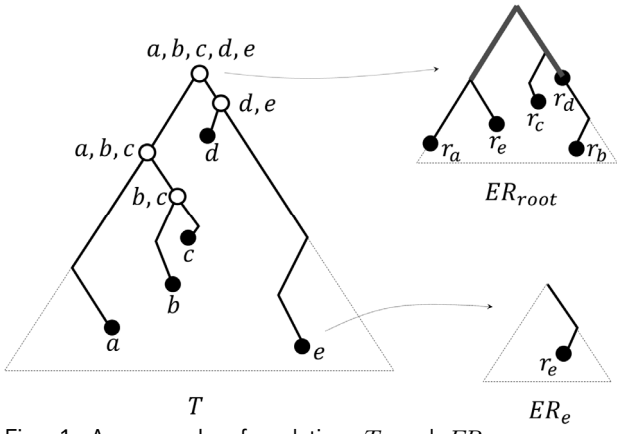


Fig. 1. An example of updating T and ER_α.
그림 1. T와 ER_α의 업데이트 예

행 값인 r_e에 대응되는 노드(검은색 원)와 그로부터 루트까지의 경로(실선)상의 모든 노드가 업데이트된다. 구체적으로, r_e에 대응되는 노드는 activecount를 1 증가(또는 감소)시키고, 경로상의 모든 노드의 count를 1 증가(또는 감소)시킨다.

그 후 T에서 루트까지 경로를 따라가며 각 노드의 ER_α에 대해 동일한 작업이 진행된다. 그림 1과 같이 경로의 마지막인 루트의 ER_{root}를 보면 r_e에 대응되는 노드(검은색 원)와 그로부터 루트까지의 경로(실선)상의 모든 노드가 업데이트된다.

한편, T에서 다섯 노드의 루트까지 경로는 서로 중복되는 부분이 반드시 생긴다. 그림 1에서 흰색 원으로 표현된 노드들은 서로 다른 경로들이 처음으로 만나는 노드를 나타낸다. 예를 들어, b,c로 표현된 노드부터 루트까지는 ER_α에서 r_b와 r_c에 대한 업데이트가 모두 필요하고, a,b,c로 표현된 노드부터 루트까지는 ER_α에서 r_a, r_b, r_c에 대한 업데이트가 모두 필요하다.

T의 루트는 모든 경로가 만나는 곳으로, 그림 1의 ER_{root}와 같이 r_a부터 r_e까지의 모두에 대해 업데이트가 필요하다. 특히 붉은색 선으로 나타낸 경로상의 노드들은 count를 1 증가(또는 감소)시키는 작업을 중복해서 하게 된다.

2. ER_α 업데이트를 개선한 알고리즘

$n\times m$ 크기의 이차원 배열 F와 필터의 반경 r에 대한 중간값 필터 알고리즘은 그림 2와 같다. Kim 알고리즘과 달라진 부분은 초기화(3행), T와 ER_α 업데이트(4, 6, 10행), ER_α의 count 업데이트(7행)이다.

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

Fig. 2. Improved median filter algorithm.
그림 2. 개선된 중간값 필터 알고리즘

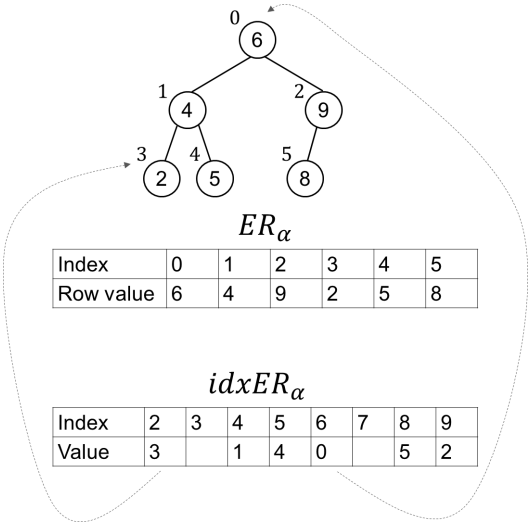


Fig. 3. An example of ER_α and $idxER_\alpha$
그림 3. ER_α와 $idxER_\alpha$의 예

우선 3행의 초기화 단계에서 ER_α의 역인덱스 $idxER_\alpha$를 추가로 만든다. 행 값이 t_1부터 t_2까지인 k개의 노드로 구성된 ER_α는 인덱스가 0부터 $k-1$까지인 1차원 배열에 t_1부터 t_2까지의 행 값이 저장되므로, 역인덱스 $idxER_\alpha$는 ER_α의 인덱스와 값을 뒤바꾸어 인덱스가 t_1부터 t_2까지인 1차원 배열에 0부터 $k-1$까지의 값이 저장된다. 그림 3은 ER_α와 $idxER_\alpha$의 관계를 보여주는 예시이다.

4행과 6행, 10행에서 T의 업데이트는 Kim 알고리즘과 같지만 ER_α의 업데이트는 T에서 노드 α의 레벨에 따라 다르게 처리한다. 경계치 d를 이용하여 $level_\alpha > d$인 경우에는 Kim 알고리즘과 같게 처리하고, $level_\alpha \le d$인 경우에는 필터 범위에서 빠지거나 추가되는 픽셀에 해당하는 노드의 activecount만 변경한다. 이때 ER_α에서 해당하는 노드를 찾기 위해 트리 탐색을 하지 않고 $idxER_\alpha$을 이용하여 상수시간에 처리한다.

$level_\alpha \le d$인 경우 ER_α의 경로상의 노드들의 count 변경은 픽셀별로 하지 않고 모두 모아 6행에서 ER_α별로 진행한다. $level_\alpha \le d$인 각 α의 ER_α에는 다양한 수의

(a) (b) (c)

Fig. 4. (a) Input image with size 1008×1344. (b) output image of median filter algorithm with $r=5$. (c) output image of median filter algorithm with $r=100$.
그림 4. (a) 1008×1344 크기의 이미지. (b) $r=5$인 중간값 필터가 적용된 결과. (c) $r=100$인 중간값 필터가 적용된 결과

노드들이 다양한 위치에서 activecount가 변경되어 있어 ER_α의 모든 노드에 대한 count 계산을 진행한다.

경계치 d의 구체적인 값은 다음 장에서 살펴본다.

Ⅳ. 실험 및 분석

실험환경은 2.4 GHz Intel Core i9 프로세서에 64GB 메모리가 설치된 하드웨어에 운영체제는 macOS Monterey 이며, 컴파일러는 Apple clang version 13.1.6을 이용하였다.

입력 이미지는 1008×1344 크기의 raw 이미지 파일을 Photoshop을 이용하여 32비트 부동소수점 형식의 gray로 변환한 TIFF 파일과 637×502 크기의 HDR 이미지 파일을 gray로 변환한 TIFF 파일을 이용하였다. 그림 4(a)와 그림 5(a)는 실험에 사용된 입력 이미지이며, 여기에 반경 r이 5인 경우와 100인 경우의 중간값 필터가 적용된 결과 이미지가 각각 그림 4(b)와 그림 4(c), 그리고 그림 5(b)와 그림 5(c)이다.

1. 경계치 d

T에서 노드 α의 레벨이 낮을수록 ER_α에서 count 변경이 중복되는 노드들이 많아지고 α의 레벨이 높을수록 ER_α에서 count 변경이 중복되는 노드가 줄어들어 적절한 경계치 d를 정해야 한다.

가. 1008×1344 크기 이미지

필터의 반경 r을 5부터 100까지 5씩 늘려가며 제안 알고리즘에서 count가 변경되는 총횟수를 d를 변화시켜 가며 세었고 일부를 표 2에 나타냈다. d가 1인 경우는 ER_{root}만 모아서 count 계산을 한 경우를 의미한다.

Table 1. The number of times that the value of count was changed depending on d (r is radius size).

표 1. 경계치 d에 따른 count 값이 변경된 횟수 변화(r은 반경)

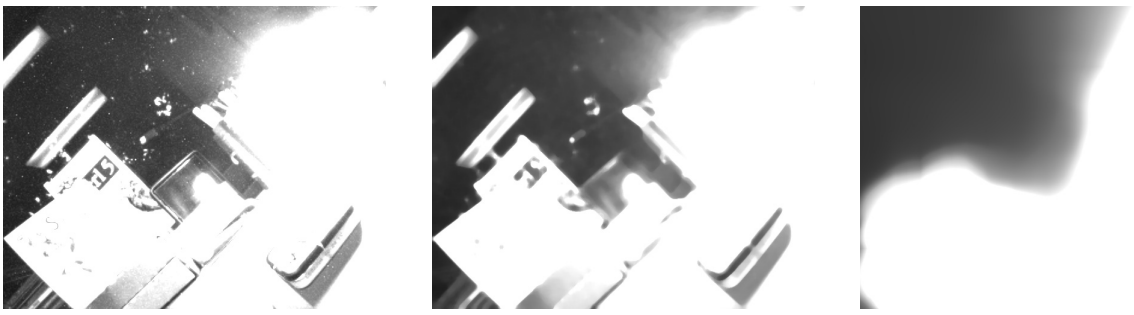
d	$r=5$	$r=50$	$r=100$
1	158 751 871	514 022 536	627 610 705
2	140 867 752	474 090 624	582 914 826
3	125 681 336	436 988 549	541 060 471
4	115 451 096	404 940 571	504 041 829
5	112 586 005	382 257 520	476 119 722
6	117 138 848	376 986 388	465 184 782
7	126 993 213	401 773 497	485 368 984
8	135 175 226	469 400 827	558 219 678

반경 r이 5인 경우 경계치 d가 5일 때 count 값을 변경한 횟수가 가장 적고, r이 50과 100인 경우에는 d가 6일 때 가장 적음을 알 수 있다. 나머지 r에 대해 20까지는 d가 5일 때가, r이 25 이상이면 d가 6일 때가 가장 적었다.

Table 3. Time for proposed algorithm depending on d (r is radius size).

표 3. 경계치 d에 따른 수행시간 변화(r은 반경)

d	$r=5$	$r=50$	$r=100$
1	1.929769	5.700306	9.830350
2	1.799481	5.547290	9.870563
3	1.731230	5.504685	9.666978
4	1.647420	5.417230	9.588368
5	1.592637	5.266487	9.536999
6	1.551383	5.219816	9.385611
7	1.563925	5.458904	9.186101
8	1.594401	5.080188	9.213331



(a) (b) (c)

Fig. 5. (a) Input image with size 637×502. (b) output image of median filter algorithm with $r=5$. (c) output image of median filter algorithm with $r=100$.

그림 5. (a) 637×502 크기의 이미지. (b) $r=5$인 중간값 필터가 적용된 결과. (c) $r=100$인 중간값 필터가 적용된 결과

표 3은 같은 경우에 대한 수행시간을 나타내는데, count 값의 변경 횟수 변화와 유사한 형태를 보이지만 최소시간을 보이는 d는 좀 더 커졌다. 반경 r이 5인 경우 d가 6, r이 100인 경우 d가 7일 때 가장 빠른 수행시간을 보였다. r이 50인 경우는 d가 9일 때 가장 빨랐으며, 20가지 서로 다른 r에 대해 가장 빠른 수행시간을 보인 d를 정리해보면 그림 6과 같다. d가 8인 경우 8가지의 반경에서 가장 좋은 결과를 보였다.

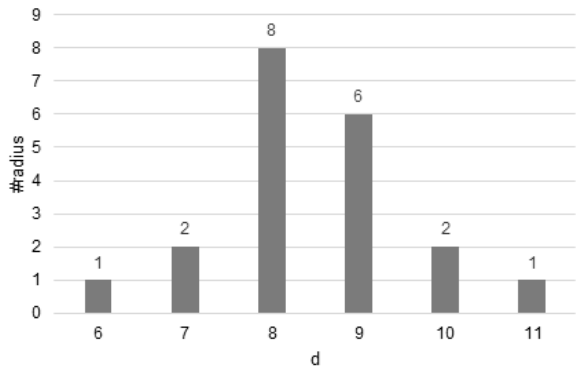


Fig. 6. Number of radius size showing the fastest execution time by d.
그림 6. 경계치 d별로 가장 빠른 수행시간을 보인 반경의 개수

나. 637×502 크기 이미지

필터의 반경 r을 5부터 100까지 5씩 늘려가며 제안 알고리즘에서 count가 변경되는 총횟수를 d를 변화시켜가며 세었고 일부를 표 4에 나타냈다.

Table 4. The number of times that the value of count was changed depending on d (r is radius size).
표 4. 경계치 d에 따른 count 값이 변경된 횟수 변화(r은 반경)

d	$r=5$	$r=50$	$r=100$
1	36 073 861	107 963 345	127 040 775
2	31 962 993	99 410 388	117 825 307
3	28 543 979	91 486 514	109 211 018
4	26 214 741	84 669 250	101 710 673
5	25 471 673	79 831 309	96 250 579
6	26 451 251	78 505 708	94 551 455
7	28 672 073	83 033 123	99 523 483
8	30 536 785	95 796 706	115 096 412

반경 r이 5인 경우 경계치 d가 5일 때 count 값을 변경한 횟수가 가장 적고, r이 50과 100인 경우에는 d가 6일 때 가장 적음을 알 수 있다. 나머지 r에 대해 15까지는 d가 5일 때가, r이 20 이상이면 d가 6일 때가 가장 적었다.

Table 5. Time for proposed algorithm depending on d (r is radius size).
표 5. 경계치 d에 따른 수행시간 변화(r은 반경)

d	$r=5$	$r=50$	$r=100$
1	0.411759	1.169442	2.134484
2	0.403674	1.162965	2.315362
3	0.389348	1.138641	2.364425
4	0.374519	1.097739	2.296443
5	0.359389	1.084434	2.276171
6	0.348722	1.049523	2.204511
7	0.346143	1.045325	2.197082
8	0.349049	1.032161	2.192736

표 5는 같은 경우에 대한 수행시간을 나타내는데, count 값의 변경 횟수 변화와 유사한 형태를 보이지만 최소시간을 보이는 d는 좀 더 커졌다. 반경 r이 5인 경우 d가 7, r이 50과 100인 경우 d가 8일 때 가장 빠른 수행시간을 보였다. 5부터 100까지 20가지 서로 다른 r에 대해 가장 빠른 수행시간을 보인 d를 정리해보면, 그림 7과 같이 8이 15가지로 가장 많았고 7이 3가지, 9가 2가지로 나타났다.

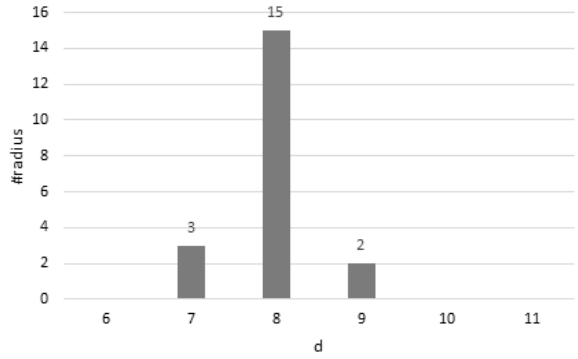


Fig. 7. Number of radius size showing the fastest execution time by d.
그림 7. 경계치 d별로 가장 빠른 수행시간을 보인 반경의 개수

2. 수행시간 비교

제안 알고리즘, Kim 알고리즘, Gil-Werman 알고리즘의 세 가지 중간값 필터를 이용하여 수행시간을 비교하는 실험을 진행하였다. 실험은 10번씩 반복하여 평균값을 결과로 이용하였다. 제안 알고리즘의 경계치 d는 8을 이용하였다.

가. 1008×1344 크기 이미지

Table 6. Time versus radius size r for three algorithms.
표 6. 세 알고리즘에 대한 실험 결과(r은 반경)

r	Proposed	Kim	Gil-Werman
5	1.594401	1.762098	4.068397
10	2.187647	2.514190	6.163209
15	2.609062	3.103623	7.741886
20	2.994472	3.563040	9.298363
25	3.309837	3.976902	10.592756
30	3.642986	4.350942	11.612142
35	4.190959	4.958548	13.715134
40	5.141780	5.382008	14.476039
45	4.568531	5.362887	17.298753
50	5.080188	5.585763	20.580985
55	6.349445	7.054225	20.690023
60	6.600366	7.504962	22.480175
65	6.904161	7.448193	24.803000
70	7.274525	7.911984	26.019688
75	7.467066	8.947364	31.345986
80	7.958840	8.625829	28.587866
85	7.877660	8.567923	36.388280
90	8.404113	9.304050	35.562595
95	8.679258	9.318752	31.525550
100	9.213331	9.534224	34.156462

표 6은 세 알고리즘의 수행시간을 비교한 표이다. Gil-Werman 알고리즘은 Kim 알고리즘보다 2.3배에서 4.2배 느렸고 제안 알고리즘보다 2.6배에서 4.6배 느렸다. Kim 알고리즘과 제안 알고리즘을 비교해보면, 반경 25일 때 제안 알고리즘은 3.3098초로 Kim 알고리즘의 3.9769초에 비해 16.8% 빠른 결과를 보였고, 반경 100일 때 제안 알고리즘은 9.2133초로 Kim 알고리즘의 9.5342초에 비해 3.4% 빠른 결과를 보였다. 제안 알고리즘은 실험한 모든 반경에 대해 평균적으로 Kim 알고리즘보다 11.0% 빠른 결과를 보였다. 그림 8은 표 6을 그래프로 나타낸 것이다.

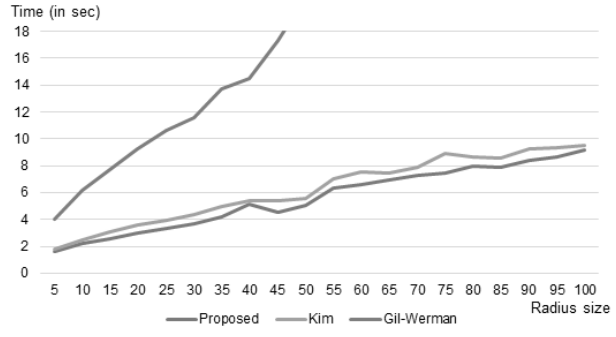


Fig. 8. Time versus radius size for three algorithms.
그림 8. 세 알고리즘에 대한 실험 결과. x축은 반경, y축은 수행시간(초)

나. 637×502 크기 이미지

Table 7. Time versus radius size r for three algorithms.
표 7. 세 알고리즘에 대한 실험 결과(r은 반경)

r	Proposed	Kim	Gil-Werman
5	0.349049	0.405510	0.897498
10	0.487957	0.577686	1.395984
15	0.577245	0.693747	1.830353
20	0.668534	0.792942	2.155327
25	0.714293	0.860045	2.422420
30	0.778866	0.937451	2.676466
35	0.964197	1.072450	3.064251
40	1.141591	1.160770	3.844871
45	0.962974	1.123074	5.965864
50	1.032161	1.142836	6.728715
55	1.387545	1.370296	8.194640
60	1.463353	1.525860	6.344965
65	1.486519	1.432863	11.293167
70	1.630097	1.668511	9.256854
75	1.586523	1.892662	6.995731
80	1.715230	1.826448	9.738166
85	1.718095	1.827349	14.043203
90	1.850035	1.883130	12.846469
95	1.935946	1.867651	10.164403
100	2.192736	2.134484	11.232059

표 7은 세 알고리즘의 수행시간을 비교한 표이다. Gil-Werman 알고리즘은 Kim 알고리즘보다 2.3배에서 7.7배 느렸고 제안 알고리즘보다 2.6배에서 8.2배 느렸다. Kim 알고리즘과 제안 알고리즘을 비교해보면, 반경 25일 때 제안 알고리즘은 0.7143초로 Kim 알고리즘의 0.8600초에 비해 16.9% 빠른 결과를 보였고, 반경 90일 때 제안 알고리즘은 1.8500초로 Kim 알고리즘의 1.8831초에 비해 1.8% 빠른 결과를 보였다. 반경 65일 때에는 제안 알고리즘은 1.4865초로 Kim 알고리즘의 1.4329초보다 3.7% 느린 결과를 보인 경우도 있었지만, 제안 알고리즘은 실험한 모든 반경에 대해 평균적으로 Kim 알고리즘보다 7.8% 빠른 결과를 보였다. 그림 9는 표 7을 그래프로 나타낸 것이다.

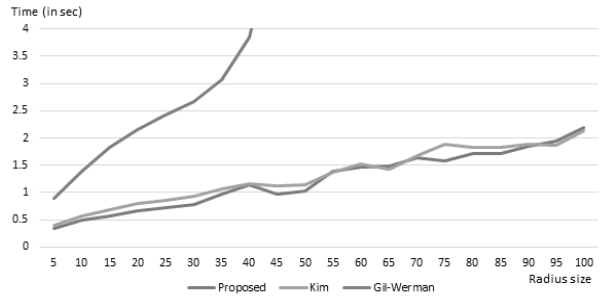


Fig. 9. Time versus radius size for three algorithms.
그림 9. 세 알고리즘에 대한 실험 결과. x축은 반경, y축은 수행시간(초)

Ⅲ. 결론

본 논문에서는 부동소수점 형식의 이미지에 중간값 필터를 효율적으로 적용하는 알고리즘을 제안하였다. 제안 알고리즘은 Kim 알고리즘에서 사용하는 rowIOBBST의 업데이트 방식을 개선하여 실험적으로 수행시간을 단축하게 함을 보였다.

향후 연구로는 rowIOBBST의 count를 모아서 계산할 때 보다 효율적으로 처리하는 방안과 함께 최적의 경계치 d를 찾기 위한 보다 엄밀한 분석이 필요하다. 또한, 제안 알고리즘을 FPGA나 GPU에 적용하는 방안도 연구할 필요가 있다.

References

[1] M. Le Pendu, C. Guillemot and D. Thoreau, "Adaptive Re-Quantization for High Dynamic Range Video Compression," in *Proc. of 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp.7367-7371, 2014. DOI: 10.1109/ICASSP.2014.6855031

[2] A. Darmont, *High Dynamic Range Imaging: Sensors and Architectures*, SPIE press, 2013.

[3] T. Fang, "On Performance of Lossless Compression for HDR image Quantized in Color Space," *Signal Processing: Image Communication*, vol.24, no.5, pp.397-404, 2009.
DOI: 10.1016/j.image.2009.01.002

[4] R. Chityala and S. Pudipeddi, *Image Processing and Acquisition using Python*, 2nd ed. CRC Press, 2021.

[5] 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

[6] D. Mayerich, B. H. McCormick and J. Keyser, "Noise and Artifact Removal in Knife-Edge Scanning Microscopy," *ISBI*, pp.556-559, 2007.
DOI: 10.1109/ISBI.2007.356912

[7] 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.

[8] J. Lee, W. An and Y. Choe, "Mapping the Full Vascular Network in the Mouse Brain at Submicro-meter Resolution," *39th Annual International Conference of the IEEE EMBC*, 2017.
DOI: 10.1109/EMBC.2017.8037564

[9] 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

[10] B. Weiss, "Fast Median and Bilateral Filtering," *ACM Trans. Graph.*, vol.25, pp.519-526, 2006.
DOI: 10.1145/1141911.1141918

[11] 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

[12] A. Alekseychuk, "Hierarchical Recursive Running Median," *19th IEEE International Conference on Image Processing*, pp.109-112, 2012.

[13] K. D. Sherwin and K. I. Wang, "Median Filters on FPGAs for Infinite Data and Large, Rectangular Windows," *ACM Trans. Reconfig. Technol. Syst.*, 2022. DOI: 10.1145/3530273

[14] P. Szántó and B. Fehér, "Hierarchical Histogram-based Median Filter for GPUs," *Acta*

Polytechnica Hungarica, vol.15, no.2, pp.49-68, 2018.
[15] 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, p.173, 1975.
[16] R. J. Tibshirani, "Fast Computation of the Median by Successive Binning," arXiv:0806.3301v2 [stat.CO], 2009.
[17] 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
[18] J. W. Kim, "Computing Median Filter for over 16-bit Depth Images," *j.inst.Korean.electr.electron.eng*, vol.24, no.2, pp.507-513, 2020.
DOI: 10.7471/ikeee.2020.24.2.507

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.