Fast Rank Algorithms Based on Multiscale Histograms

Maria V. Storozhilova, Dmitry V. Yurin
Faculty of Computational Mathematics and Cybernetics,
Lomonosov Moscow State University, Moscow, Russia
mariastorozhilova@gmail.com, yurin@cs.msu.su

Abstract

Rank algorithms for ε_V and KNV neighborhood average calculating are used seldom due to their computational complexity. In this paper fast versions of these algorithms have been proposed. They are based on multiscale histograms. Also the impulse noise suppression method is proposed.

Keywords: *Rank algorithms, median, ε_V neighborhood, KNV neighborhood, multiscale histograms.*

1. INTRODUCTION

The main problem of rank algorithms [5] is their computational complexity, thus only median filtering and maximum/minimum elements search are widely spread in practical applications. Rank algorithms do not blur the edges of objects and fit good for the impulse noise suppression. The approach based on use, maintaining and updating of histograms [2] lets to decrease median filtering complexity from $O(r^2 \log r)$ to $O(r)$, where r is a neighborhood radius. One of the latest works in this area describes the algorithm [3] which lets to decrease histogram updating complexity to $O(1)$ during an image processing. However, the approach [3] has maximum performance only for square neighborhood area and does not provide any optimization of median calculation process.

This work considers algorithms for fast calculation of ε_V and KNV neighborhood average and fast search of arbitrary element in a rank series. For histogram construction and updating both [2] or [3] approaches may be used.

2. MULTISCALE HISTOGRAMS

The highest and the roughest level L_0 of multiscale histogram (see Figure 1) contains the total number of points in current pixel's local neighborhood and the sum of their brightness (the interval from 0 to I_{max}, the maximum intensity of the image). The next lower level (L_1) of histogram contains the same information for 2 sections in the intensity range (0 to $I_{max}/2$ and from the $I_{max}/2+1$ to I_{max}), at the level of L_2 - for 4 sections. In other words every element of higher level includes two corresponding elements of lower level.

The lowest level is the usual histogram where each element corresponds to one intensity value. This level of histogram contains the number of pixels in the neighborhood with corresponding brightness values. The number of this level coincides with the number of bits that represents the image intensity (L_{max}).

Let us consider the histogram element v_0 on the L_3 level and its neighborhood (element $h_{L3}[3]$, see Figure 1). The absolute difference between the remote from v_0 tail of $h_{L3}[3]$ element and v_0 position on L_{max} level will be called the distance from the neighborhood to the adjacent left or right element. For example distance from $h_{L3}[3]$ to $h_{L3}[2]$ equals 13-8 = 5, from $h_{L3}[3]$ to $h_{L3}[4]$ equals 19-13 = 6. Left or right element, which distance from the neighborhood is minimal will be called the closest element.

During the histogram construction process for the current pixel, counter in the appropriate cell on each level of the multiscale histogram is incremented or decremented. Thus, for grayscale images with 256 shades of gray 8 histogram levels should be updated simultaneously, i.e. the complexity of the histogram construction process increases 8 times. However, various mean values might be computed with logarithmic complexity (8) instead of linear (256) in the number of 256 intensity gradations. Acceleration is especially notable in the calculation of complex expressions [5], for example average (ε_V(med(KNV(v_0))).

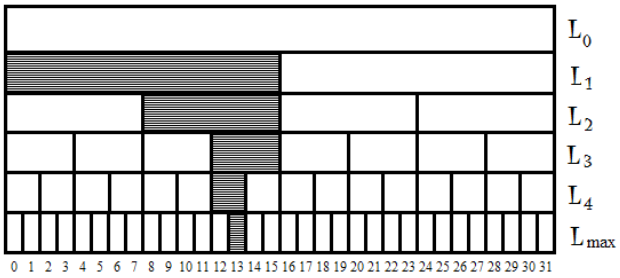

Figure 1: Multiscale histogram.

3. ε_V NEIGHBORHOOD

Let us introduce some useful definitions: the rank series $\{v(r)\}$ is a one-dimensional sequence of N pixels of the neighborhood whose elements are sorted in ascending order with respect to their values: $\{v(r): v(r) \le v(r+1), r=0,1,..N-1\}$. Pixel v_0 rank R is the number of the element in the rank series.

Definition 1. ε_V neighborhood is a subset of pixels $\{v(r)\}$ whose values deviate from the value of the central pixel v_0 at most by predetermined quantities -ε and +ε:

$$\varepsilon_V(v_0) = \{v(r) : v_0 - \varepsilon \le v(r) \le v_0 + \varepsilon\}, r = 0..N-1.$$

ε_V neighborhood average calculation is a simplified analogue of bilateral filter [4]. Bilateral filtering has high computational complexity and fast algorithms give only approximate results. The following method is proposed for mean value calculation in ε_V neighborhood:

Algorithm 1. ε_V neighborhood average.

Input: ε – value of epsilon parameter;

v_0 – intensity of the current pixel;

$v_L := v_0 - \varepsilon$;

$v_R := v_0 + \varepsilon$;

$L_i := L_{max}$ – level number;

$n = 0$ – number of elements in summation;

$s = 0$ – sum of histogram elements;

h – histogram vector;

Output: *average*(v_L; v_R; s; n; L_i);

1: **if** v_R is even **then**

2: $s := s + h_{Li}[v_R] \cdot v_R$;

3: $n := n + h_{Li}[v_R]$;

4: $v_R := v_R - 1$;

5: **if** v_L is odd **then**

6: $s := s + h_{Li}[v_L] \cdot v_L$;

7: $n := n + h_{Li}[v_L]$;

8: $v_L := v_L + 1$;

9: **if** ($v_L < v_R$) **then**

10: *average*($v_L/2$; $v_R/2$; s; L_{i-1})

11: **else**

12: average value is calculated: average := s/n;

First, equidistant segment borders (v_L and v_R) are calculated for the center point on the most detailed level of the histogram. Then, while $v_L < v_R$, algorithm recursively shifts to a higher level. After shift the borders are rounded to the next power of two. For the current level summation involves only the segments that fall under the rounding on each side of the current section.

Thus, number of operations for mean calculation in the local neighborhood of current pixel (for arbitrary central element v_0 and ε value) will not exceed 2*7 additions and comparisons, and one division for an image in 256 shades of gray. I.e. the proposed algorithm has the logarithmic complexity of $O(L_{max})$ rather than linear complexity of $O(2^{Lmax})$. Independence on the choice of v_0 means that the entire class of algorithms for ε_V neighborhood average calculating [5] is implemented and the central element may be the current pixel, or the mean value or the median value, etc.

4. KNV - NEIGHBORHOOD

Definition 2. *KNV*-neighborhood is a subset of a specified number K of pixels $\{v_{n,m}\}$ whose values are nearest to the value of the central pixel v_0:

$$KNV\ (v_0) = \left\{ v(r) : \sum_{r=p}^{p+K-1} |v_0 - v(r)| = \min_p \right\}$$

Considered above algorithm for ε_V neighborhood average has a disadvantage, connected with the problem of right ε choice (it is also a problem for bilateral filtering). The value of ε should depend on image content and must be calculated or set considering a priori image information. Moreover, a single ε value choice for the entire image can be impossible. Therefore algorithms with a priori clear parameter values are of great interest. Particularly – the algorithm for KNV neighborhood average is among them.

From the definition it is clear that, for example, angles of objects ≥ 90º will not be rounded off in the case of two-color image with parameter K = ¼ N; with parameter K = ⅛ N angles less sharp than 45º will not be rounded either. Such a priori clear dependence of the results of algorithm work on its parameter makes it applicable for use, even considering its higher complexity.

Algorithm 2. KNV neighborhood average.

Input: k – value of K parameter;

v_0 – the intensity of the current pixel;

v_L, v_R – left and right borders of summing range;

L_i – the number of the level, where $h_{Li+1}[v_0] > K$;

$n = 3$ – the number of elements, which is necessary to add on a considering histogram level;

$s = 0$ – sum of histogram elements;

h – histogram vector;

Output: *average_KNV*(v_0,v_L,v_R,k,s,n,L)

1: **if** $v_0 - (v_L \cdot 2^{Lmax-L})) > ((v_R + 1) \cdot 2^{Lmax-L} - 1) - v_0$ **then**

2: **if** $h_L[v_R] < k$ **then**

3: $s := s + h_L[v_R] \cdot v_R$;

4: $k := k - h_L[v_R]$;

5: $v_R := v_R + 1$;

6: $n := n - 1$;

7: **else if** $n \neq 2$

8: $s := s - h_L[v_L] \cdot v_L$;

9: $k := k + h_L[v_L]$;

10: $v_L := v_L + 1$;

11: $n = 0$;

12: **else**

13: **if** $h_L[v_L] < k$ **then**

14: $s := s + h_L[v_L] \cdot v_L$;

15: $k := k - h_L[v_L]$;

16: $v_L := v_L - 1$;

17: $n := n - 1$;

18: **else if** $n \neq 2$

19: $s := s - h_L[v_R] \cdot v_R$;

20: $k := k + h_L[v_R]$;

21: $v_R := v_R - 1$;

22: $n = 0$;

23: **if** n ≠ 0

24: *average_KNV* (v_0,v_L,v_R,k,s,n,L);

25: **else**

26: *average_KNV* (v_0,$v_L \cdot 2$,$v_R \cdot 2$,k,s,3,$L+1$);

First the roughest level, containing the number of elements not greater than K, is searched starting with the most detailed level (L_{max}). Initial sum is the element of this level, which contains the central pixel of the neighborhood (v_0). It is proposed to add not more than 3 closest to v_0 elements at each level. If adding one of these elements results that an intermediate sum will contain more than K counts, n=3 or n=1, the opposite border element is subtracted from the sum and step to the lower level is performed. This step allows keeping the symmetry of the neighborhood. At the most detailed level (L_{max}) a single, closest to the center, element is added first. After this step, the neighborhood becomes completely symmetrical. In order to keep exactly K elements in the resulted sum it is necessary to add no more than 2 border elements from each side.

Lemma 1. Proposed algorithm allows constructing symmetrical neighborhood.

Proof. Let L_0 represent the level, at which the first element $h_{L0}[i]$ was added, it contains v_0 – the central pixel of the neighborhood. The L_0 also contains odd number (1) of elements.

Let us propose that, the neighborhood contains odd number M of elements on the level L_0, the central element of the neighborhood contains v_0. Then, at the level of L_0+1 the neighborhood contains 2M corresponding elements (because of a histogram construction, see Figure 1.). Three nearest (see Chapter 2) elements, which are not yet included into the neighborhood, will be added by turn. Let us consider 2 cases.

1) Let addition the first item failed. Then the opposite extreme element is excluded from the neighborhood. In that case the distance between left and right borders of the neighborhood will differ by no more than the size of one element (see Chapter 2) of the level of L_0+1 of histogram.

2) Let addition the first item succeeded. The neighborhood will be more symmetrical after adding the first element on current level, because the element was added from the border where the distance from that border to v_0 is minimal.

The addition of the second element.

1) Let the addition of the second element failed. Then the level handling is over.

2) Let addition the second element succeeded. Then the procedure of addition of the third element is similar to the procedure of addition of the first element.

Thus, on the level of L_0+1 the neighborhood contains the odd number of elements, and the central element of the neighborhood contains v_0. Then the distance between left and right borders of the neighborhood differs by no more than the size of one element of the level of L_0+1 of histogram.

Finally, because of the principle of mathematical induction, all the levels from L_0 to L_{max} contain the odd number of elements, the central element of each level contains v_0 and the dissymmetry of the neighborhood is lesser than the size of the element on each level. Addition of the nearest element on the level of L_{max} guarantees that the neighborhood will be symmetrical.

Lemma 2. The algorithm guarantees that the neighborhood will contain less than K elements for any $L \geq L_0$. Addition of one element on the right and on the left sides guarantees that the neighborhood will contain not less than K elements.

Proof. Let L_0 represent the level, at which the first element $h_{L0}[i]$ was added, it contains v_0 – the central pixel of the neighborhood. The L_0 also contains odd number (1) of elements. Then the neighborhood will contain ≥ K counts with nearest left and right elements.

Let us assume that the histogram contains the odd number of elements on the level of L_0, the central element of the neighborhood contains v_0. Then, at the level of L_0+1 the neighborhood contains 2M corresponding elements (because of a histogram construction Img.1.). Three nearest elements (see Chapter 2), which are not yet included into the neighborhood, will be added by turn. Let us consider 2 cases.

1) Let addition the first item failed. Then the opposite extreme element is excluded from the neighborhood. In that case the number of counts in the neighborhood will be < K. But the neighborhood will contain ≥ K counts with nearest left and right elements.

2) Let addition the first item succeeded. Then the neighborhood contains < K counts.

The addition of the second element.

1) Let the addition of the second element failed. Then the level handling is over. In that case the neighborhood will contain ≥ K counts with nearest left and right elements.

2) Let addition the second element succeeded. Then the procedure of addition of the third element is similar to the procedure of addition of the first element.

Thus, the neighborhood contains ≥ K counts with nearest left and right elements on each level from L_0 to L_{max}. If one more element from each border of the neighborhood is considered together with previous 3 elements on the level of L_{max} (in all 5 elements on the level) then neighborhood will be guaranteed to contain ≥ K counts.

Algorithm 2 may go beyond the borders of the histogram vector while adding or subtracting elements on each level of the histogram. To avoid the growth of difficulty because of permanent index checks, it is proposed to supplement the histogram with zeros on its full size on the right and on the left on each level.

5. SEARCH FOR AN ARBITRARY ELEMENT IN RANK SERIES

The following algorithm is proposed:

Algorithm 3. Search for element with a rank R in a series.

Input: R – element rank in a series;

$L := L_1$ – number of the level, where recursion starts;

h – histogram vector;

$v_0 := 0$ – first element of L_1 level;

Output: $rank(R,v_0,L)$

(For levels L_1 to L_{max}-1)

1: $v_0 := v_0 \cdot 2$;

2: **if** $h_L[v_0] \geq R$ **then**

3: rank(R,v_0,L+1);

4: **else**

5: R := R - $h_L[v_0]$;

6: rank(R,v_0+1,L+1);

(For level L_{max})

1: $v_0 := v_0 \cdot 2$;

2: **if** $h_L[v_0] \geq R$ **then**

3: v_0 – required element;

4: **else**

5: v_0+1 – required element;

Starting from the level L_1 a histogram element with the number of counts less than R is sought. If the first histogram element contains ≥ R elements, the search is continued in the lower-level element that the current element consists of. If the considering histogram element contains less than R elements, their quantity is subtracted from R, and the search is continued in the lower-level element that the next element on current level consists of.

Search of the item with the given rank is performed only using the first element of the lower level, which was obtained in the previous step. Just one check is needed at the level L_{max}. Thus, the search of the element with a given rank for a grayscale image in

256 shades of gray requires not more than 7 subtractions and comparisons.

6. IMPULSE NOISE SUPPRESSION

The following algorithm is proposed. A rank series is created for local neighborhood of central pixel (v_0). If v_0 is contained in K leftmost or rightmost elements in rank series, it is replaced by the mean value. This mean value is calculated over all elements of the rank series except of K leftmost, or rightmost elements, or over N-2K points. Other pixels remain unchanged. Detailed description of the algorithm is not given due to paper size limitations.

7. RESULTS

N is the number of elements in the neighborhood, r is the neighborhood radius, R is the rank for Algorithm 3.

Figure.2 shows source image with added impulse noise and the result of its filtering using the algorithm from section 6.

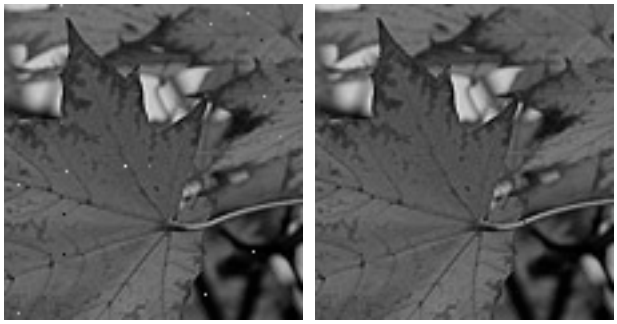

Figure. 2. Source and filtered image

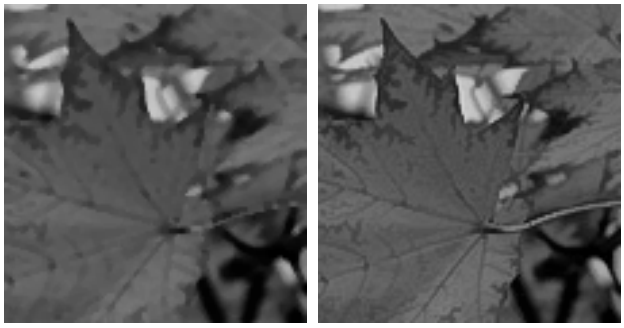

Figure. 3. Image filtered with median (left), with KNV (right)

Figure. 3 demonstrates the smoothing properties of the KNV neighborhood average on denoised image. The smoothed image on the left was calculated with the Algorithm 3 with R = N/2. The right image was calculated with the Algorithm 2, K = 1/4N. Leaf corners became rounder than on the source image after filtering with median. But KNV-based filtering left these corners as sharp as they were on denoised image.

	I_1, r= 12	I_1, r = 37	I_1, r = 62	I_2, r = 37	I_2, r = 62
ε_V	0.031	0.032	0.032	0.110	0.110
KNV	0.093	0.125	0.125	0.244	0.282
rank(N/2)	0.030	0.032	0.032	0.110	0.100
ctmf [3]	0.141	0.172	0.203	0.625	0.656

Figure. 4. Time comparison for images I_1 (size = 375x486) and I_2 (size = 1000x1000).

Figure. 4 demonstrates time statistics of proposed algorithms compared with [3] (times for rank algorithms do not include histogram updating steps).

8. CONCLUSION

This paper proposes methods for fast calculation of ε_V and KNV neighborhood average and fast search of an arbitrary element in a rank series (including minimum, maximum and median) with the use of multiscale histograms. Described algorithms do not suggest any method for histogram construction and maintaining. For this task the classic approach [2] is used. The fast algorithm from [3] is for future work. The algorithms have been implemented in C++ language with wide use of metaprogramming techniques [1] for cycles and recursions unrolling.

9. ACKNOWLEDGMENT

The research is done under support of Federal Target Program "Scientific and Scientific Pedagogical Personnel of Innovative Russia" in 2009-2013 and the Russian Foundation for Basic Research, project no. 09-07-92000-HHC_a.

10. REFERENCES

[1] Alexandrescu A., *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley. ISBN 978-0201704310, February 2001.

[2] Huang T., Yang G., and Tang G., *"A Fast Two-Dimensional Median Filtering Algorithm,"* IEEE Trans. Acoust., Speech, Signal Processing, vol. 27, no. 1, pp. 13–18, 1979.

[3] Perreault S., Hebert P., *"Median Filtering in Constant Time"*, IEEE Transactions on Image Processing, vol. 16, pp. 2389-2394, 2007.

[4] Tomasi C., Manduchi R., "*Bilateral Filtering for Gray and Color Images*", iccv, pp.839, Sixth International Conference on Computer Vision (ICCV'98), 1998

[5] Yaroslavsky L.P., Kim V., "*Rank Algorithms for Picture Processing, Computer Vision",* Graphics and Image Processing, v. 35, 1986, p. 234-258

About the author

Maria V. Storozhilova, is a student at Chair of Mathematical Physics, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Russia. Her contact email is. mariastorozhilova@gmail.com

Dmitry V. Yurin (PhD) is a senior researcher at laboratory of Mathematical Methods of Image Processing, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Russia. His contact email is yurin_d@inbox.ru