Indonesian Journal of Electrical Engineering and Computer Science
Vol. 19, No. 2, August 2020, pp. 1062~1070
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v19.i2.pp1062-1070

A new fast efficient non-maximum suppression algorithm based on image segmentation

Oday Jasim Mohammed Al-Furaiji[1], Nguyen Anh Tuan[2], Tsviatkou Viktar Yurevich[3]
[1]Department of Computer Science, Shatt Al-Arab University College, Iraq
[2,3]Department of Infocommunication Technologies,
Belarusian State University of Informatics and Radioelectronics, Belarus

Article Info

Article history:

Received Jan 6, 2020
Revised Mar 9, 2020
Accepted Mar 23, 2020

Keywords:

Feature point
Image segmentation
Local extrema
Local maxima
Non-maximum suppression
Region growing

ABSTRACT

In this paper, the problem of finding local extrema in grayscale images is considered. The known non-maximum suppression algorithms provide high speed, but only single-pixel extrema are extracted, skipping regions formed by multi-pixel extrema. Morphological algorithms allow to extract all extrema but its maxima and minima are processed separately with high computational complexity by iterative processing based on image reconstruction using image morphological dilation and erosion. In this paper a new fast efficient non-maximum suppression algorithm based on image segmentation and border analysis is proposed. The proposed algorithm considers homogeneous areas, which are formed by multi-pixel extrema and are the local maxima or minima in relation to adjacent areas, eliminating iterative processing of non-extreme pixels and assigning label numbers to local extrema during their search. The proposed algorithm allowed to increase the accuracy of local extremum extraction in comparison with known non-maximum suppression algorithms and reduce the computational complexity and the use of RAM in comparison with the morphological algorithms.

Corresponding Author:

Oday Jasim Mohammed Al-Furaiji,
Department of Computer Science,
Shatt Al-Arab University College, Basra, Iraq.
Email: odaymohammed@mail.ru

1. INTRODUCTION

Non-Maximum Suppression (NMS) is the task of finding all local maxima in an image. The term ‘non-maximum suppression’ is first appeared in an edge detection context as a method to reduce thick edge responses to thin lines [1]. This type of directional NMS operates one dimensionally (1-D) perpendicular to the edge. This term is extended to isotropic NMS to locate two-dimensional (2-D) feature points from an image [2]. The feature points are selected as local maxima of a corner image over some neighborhood. The NMS approach to corner detection was subsequently adopted by many interest point detectors [3-5].

Image processing often requires the determination of initial elements, which can be local 2-D extrema (local maxima and local minima). To search for local extrema, some non-maximum suppression algorithms [6-11] and morphological algorithms [12] are used. In which, the search for extrema is computed within small blocks (masks), usually 3×3 or $(2n + 1) \times (2n + 1)$ pixels in size. Such algorithms have low computational complexity, but they extract local extrema by iterative processing (local single-pixel maxima and local single-pixel minima are extracted separately), skip multi-pixel extrema or extreme regions (for a homogeneous region consisting of several adjacent pixels with the same values, none of the pixels in this region is detected as a local extremum).

The morphological algorithm is used for the extraction of local maxima and minima in an image by image reconstruction based on dilatation and erosion operations. It gives accurate results compared to known non-maximum suppression algorithms, and also it allows to extract both single-pixel extrema and extreme regions. However, the morphological algorithm has high computational complexity, which is associated with separate processing of maxima and minima, as well as iterative processing of the neighborhoods of all pixels.

Many image processing tasks require the determination of initial elements for the search of which algorithms can be used based on the Laplace operator [13] and the Gaussian Laplacian (LoG) [14]; Hessian determinant (DoH) [13, 15]; watershed [16, 17]; iterative binarization of regions by a variable threshold [18]. Image elements extracted using these algorithms are individual pixels, contours, or spots, which are characteristic elements, but not necessarily local extrema. Local extrema are used as base points for segmentation [19] and texture analysis of images [20]. The aim of this work is to develop a new algorithm for extracting local extrema in grayscale images with low computational complexity, high accuracy and less cost of RAM.

2. RESEARCH METHOD

2.1. Previous algorithms

A straightforward approach to NMS over a rectangular neighborhood is described in Figure 1(a). The input image pixels are visited in a raster scan order (from left to right, then from top to bottom). Each visited pixel is compared to other pixels in its neighborhood also in a raster scan order. The central pixel c is a non-maximum if a larger or equal neighbor is found. The algorithm then skips to the next pixel in the scan line. The straightforward method is simple to implement but it can take a long time to process an image. On average, however, the straightforward method requires $O(n)$ comparisons per pixel.

The complexity of a raster scan algorithm can be significantly reduced by visiting the neighboring pixels in a different order. An algorithm with a local spiral order is presented in [10] as shown in Figure 1(b). By comparing with closer neighbors first, the central pixel is guaranteed to be a 3×3-neighborhood local maximum before it is tested against a larger neighborhood.

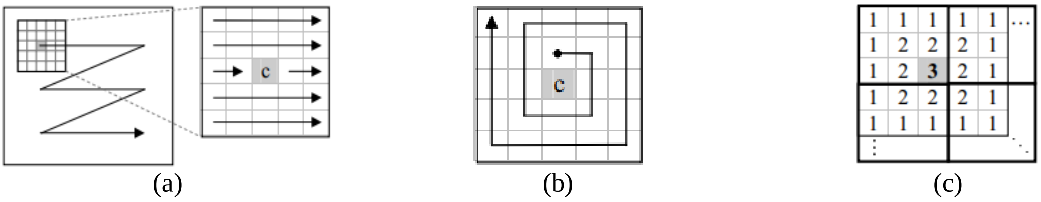


Figure 1. Previous solutions for 2-D non-maximum suppression (5×5 neighborhood), Raster scan order [9, 11], (b) Spiral order [10], (c) Block partitioning [9, 11]

Because the number of 3×3 local maxima in an image is usually small ($\leq 25\%$ of the total number of pixels), the spiral order algorithm quickly finds any non-maximum and skips to the next pixel. The number of $(2n+1) \times (2n+1)$ local maxima also decreases rapidly as the neighborhood size increases. As a result, the computational complexity of this algorithm is roughly constant (≤ 5 comparisons per pixel to detect a 3×3 non-maximum) of the irrespective neighborhood size.

In [11], the above algorithms are extended to reduce the number of comparisons to fewer than two comparisons per pixel. The algorithm first searches for 1-D local maxima along the scan line. Each 1-D maximum is then compared against its neighbors in adjacent rows. A buffer of two binary masks is kept for a current and a next scan line. As a new central pixel is processed, its future neighbors are masked out if they are smaller than the central pixel. Masked pixels will be skipped for the next processing. This 1-D non maximum suppression algorithm therefore requires one comparison per pixel as shown in Figure 2.

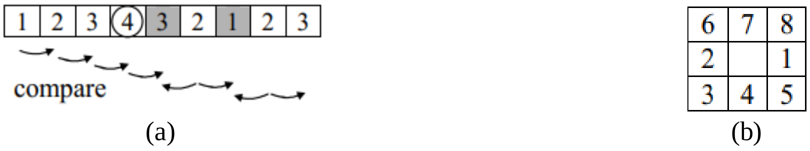
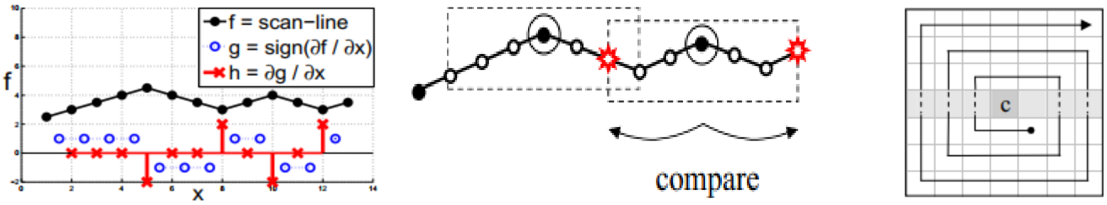


Figure 2. 3-neighborhood non-maximum suppression and 3×3-neighborhood scan order, (a) 1D non-maximum suppression, (b) 3×3 scan order

The scan-line algorithm for 3 × 3–neighborhoods can be extended to handle $(2n+1)\times(2n+1)$ – neighborhoods for $n > 1$. First, $(2n+1)$– neighborhood maxima on the current scan-line are located. These 1-D maxima serve as candidates for the 2-D maxima. Each candidate is then compared against its $(2n+1)\times(2n+1)$– neighborhood in a spiral order similar to that of Forstner's method [10]. In this case, the neighbors on the same scan-line have already been compared and can therefore be skipped. This results in a maximum of $(2n)\times(2n+1)$ neighbors to be compared per candidate. In practice, the average number of comparisons per candidate is much smaller than the spiral traverse order as shown in Figure 3(c).



(a) (b) (c)

Figure 3. Scan-line algorithm for $(2n+1)\times(2n+1)$ non-maximum suppression ($n = 3$) [11], (a) 1D peak and trough detection, (b) 1D non-maximum suppression, (c) Spiral traverse

From the above NMS algorithms, the extended algorithm to find local maximum and local minimum can be described by the following:

$$e_{MAX}(y,x)\begin{cases}1, & \forall y'\forall x'(f(y,x) > f(y+y',x+x'),\\ 0, & \forall y'\forall x'(f(y,x) \le f(y+y',x+x').\end{cases} \tag{1}$$

$$e_{MIN}(y,x)\begin{cases}-1, & \forall y'\forall x'(f(y,x) < f(y+y',x+x'),\\ 0, & \forall y'\forall x'(f(y,x) \ge f(y+y',x+x').\end{cases} \tag{2}$$

where$y = \overline{0, Y-1}, x = \overline{0, X-1}, (y' \in \{-1,0,1\}) \wedge (x' \in \{-1,0,1\}) \backslash ((y'=0) \wedge (x'=0))$. As a result of combining the matrices E_{MAX} and E_{MIN}, a matrix of local extrema E is obtained, the values of the elements of which are calculated using the expression:

$$e(y,x) = e_{MAX}(y,x) + e_{MIN}(y,x). \tag{3}$$

Regional extrema are detected better by morphological reconstruction by dilation and erosion [12], in which a *regional minimum* M of an image f at elevation t is a connected component of pixels with the value t whose external boundary pixels have a value strictly greater than t. M is a *regional minimum* at level $t \Leftrightarrow M$ is connected and

$$\begin{cases}\forall p \in M, f(p) = t,\\ \forall q \in \delta^{(1)}(M)\backslash M, f(q) > t.\end{cases} \tag{4}$$

Similarly, a *regional maximum* M of an image f at elevation t is a connected component of pixels with the value t whose external boundary pixels have a value strictly less than t. M is a *regional maximum* at level $t \Leftrightarrow M$ is connected and

$$\begin{cases}\forall p \in M, f(p) = t,\\ \forall q \in \delta^{(1)}(M)\backslash M, f(q) < t.\end{cases} \tag{5}$$

The regional extrema of an image are defined as the union of its regional minima and maxima. According to (4), the set of all maxima of an image f at level t corresponds to the connected components of the cross-section of f at level t that are not connected to any component of the cross-section of f at level *t+1*. They are therefore not reconstructed by the morphological reconstruction by dilation $CS_t(f)$ from $CS_{t+1}(f)$. The regional maxima of an image f at level t can be written by the following:

$$RMAX_t(f) = RMAX(f) \cap T_t(f) = \; CS_t(f)\backslash[CS_{t+1}(f)]. \tag{6}$$

The set of all maxima is defined by considering the union of the maxima obtained at each level t:

$$RMAX_t(f) = \cup_t \left\{CS_t(f)\backslash R^{\delta}_{CS_t(f)}[CS_{t+1}(f)]\right\}. \tag{7}$$

Since $CS_{t+1}(f) = CS_t(f-1)$ and $RMAX_t(f) \cap RMAX_{t1}(f) = \emptyset$ for $t \neq t1$, the set in (7) can be replaced with an algebraic difference and the union by a summation. The summation is then distributed and the threshold superposition principle gives:

$$RMAX(f) = f - R^{\delta}_f(f-1) \tag{8}$$

If the image data type does not support negative values, the following equivalent definition must be considered:

$$RMAX(f) = f + 1 - R^{\delta}_f(f) \tag{9}$$

Similarly, the regional minima of an image f at level t are denoted by $RMIN_t(f)$: $RMIN_t(f) = RMIN(f) \cap T_t(f) = R^{\varepsilon}_{CS_{t+1}(f)}[CS_t(f)\backslash CS_{t+1}(f)]$. The set of all regional minima is denoted by *RMIN* and defined by threshold superposition:

$$RMIN(f) = R^{\varepsilon}_f(f+1) - f \tag{10}$$

From (9) and (10) the final morphological algorithm to find all local extrema is shown in Figure 4.

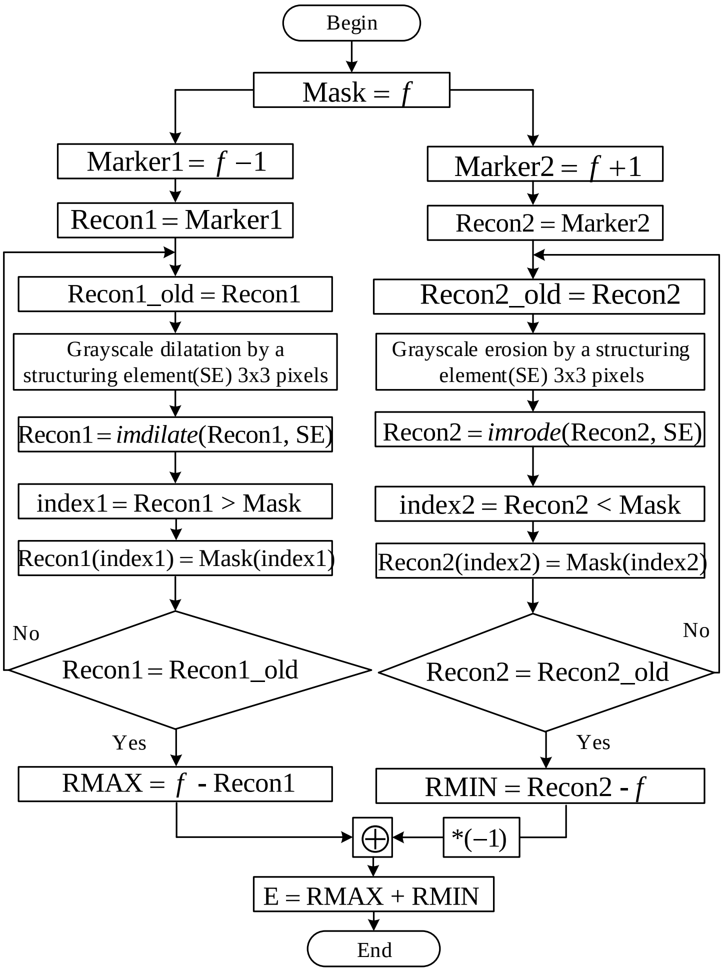


Figure 4. Finding regional extrema by morphology using matlab's built-in functions

From (1) and (3) the main disadvantages of block algorithms for isolating local extrema are the following:

a) Redundancy of processing. The independent formation of matrices by (1) also leads to redundancy of processing, since the matrix pixels are re-processed when the matrix is formed to find the local minimum.

b) Skipping multi-pixel extrema. In (1) it is shown that if a region consisting of several adjacent pixels with the same values is a local maximum in the image, none of these pixels in this region is detected as a local maximum. The same is true for local minima by (2). The number and area of such regions grow during quantization, filtering, and restoration of images after lossy compression. In such cases, skipping multi-pixel extrema leads to incomplete image segmentation, errors in detection, localization, and parameterization of objects.

c) The need for additional processing of the resulting matrix to assign label numbers to local extrema.

From (4) to (10) and Figure 4, the main disadvantages of morphological algorithms for isolating local extrema, leading to high computational complexity are the following:

a) Separate processing of local maxima and local minima.

b) Iterative processing of neighbourhoods of all pixels.

c) The need for additional segmentation for assigning label numbers to extreme regions.

2.2. Mathematical model and proposed algorithm

To eliminate the above disadvantages a new fast efficient non-maximum suppression algorithm (ENMS – Efficient Non-Maximum Suppression) to extract all local extrema in grayscale images with low computational complexity and high accuracy is proposed. The essence of the algorithm consists in scanning the image to search for non-segmented pixels, using them as starting points for growing of homogeneous regions and comparing the values of their boundary pixels with the values of the corresponding pixels of adjacent segments: the region is a local maximum (or minimum) if the values of all its boundary pixels are greater (or less) values of all adjacent pixels by the following:

$$e(y,x)=\begin{cases} -\boldsymbol{n_E}, (\forall p(y,x)\in M, f(p)=t)\wedge \\ \wedge\left(\forall q(y,x)\in\delta^{(1)}(M)\backslash M, f(q)>t\right), \\ \boldsymbol{n_E}, (\forall p(y,x)\in M, f(p)=t)\wedge \\ \wedge\left(\forall q(y,x)\in\delta^{(1)}(M)\backslash M, f(q)<t\right), \\ \boldsymbol{0},\ otherwise. \end{cases} \tag{11}$$

where $y=\overline{0,Y-1}$, $x=\overline{0,X-1}$, $n_E\in[0,N_E]$ – the number of current extreme region $R(n_E)$, N_E – the total number of extreme regions. In (11) the values of the elements $e(y,x)$ of the matrix E of local extrema indicate that the corresponding pixels of the image belong to the maximum (n_E), minimum ($-n_E$), or non-extremum (0).

The proposed algorithm based on segmentation and border analysis consists of the following steps [21, 22]:

Step 1. Buffering the original image $f(X,Y)$ and obtaining its size.

Step 2. Formation of zero matrix of segmentation SM (X,Y) and matrix of extrema $E(Y,X)$ having the same size as the original image.

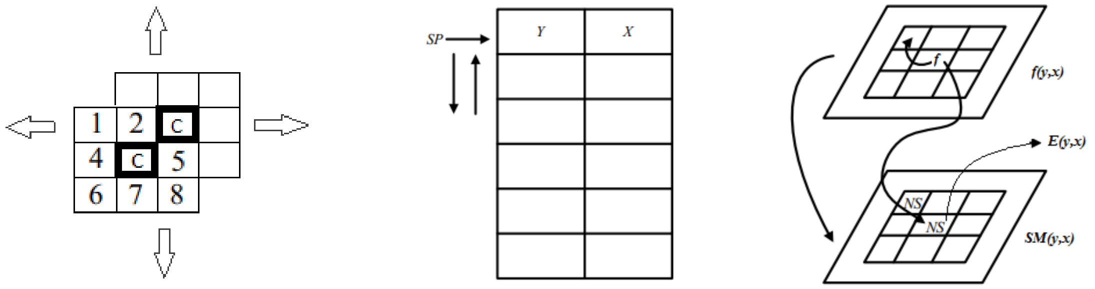
Step 3. Formation of two stacks Y and X for temporary storage of pixel coordinates during processing as shown in Figure 5(b).

Step 4. Setting initial values for the counter of segments NS, the counter of extrema N_E and the pointer of region growing SP.

Step 5. Beginning of the segmentation loop. A selection of the current pixel of the original image and the corresponding element of the segmentation matrix SM. If the current element of the segmentation matrix is zero (i.e., the current image pixel has not yet been segmented), then the coordinates of the current image pixel are entered in the Y and X stacks, the values of the segment counter and stack pointer are increased by one: $NS = NS + 1$, $SP = SP + 1$; the current element of the segmentation matrix SM is assigned the value NS.

Step 6. Region growing loop. Check stacks Y and X for the temporary storage of coordinates. If the stack is not empty, the following operations are performed: extracting the current pixel coordinates from the Y and X stacks; decreasing stack pointer per unit ($SP = SP - 1$); a selection of eight pixels of the original image adjacent to the current pixel as shown in Figure 5(a) and (c); loading on the stack of coordinates of adjacent pixels whose values coincide with the current pixel, with a corresponding increment in the value of the stack pointer; assigning elements of the segmentation matrix SM corresponding to adjacent pixels whose values match the current pixel to NS values. Continue processing the stack until $SP > 0$.

Step 7. Non-extremum suppression. Comparing the values of their boundary pixels of current region (step 6) with the values of the corresponding pixels of adjacent segments: the region is a local maximum (or minimum) if the values of all its boundary pixels are greater (or less) values of all adjacent pixels. Then, the value of the extreme counter is increased by one $N_E = N_E + 1$ and elements of the matrix E are assigned to label N_E as in (11).
Step 8. Go to the beginning of the segmentation loop (step 5) if not all pixels of the image are processed.
As a result of this algorithm, a matrix of extrema is formed, the value of each element of which indicates the label of the extrema or its absence. This data is used for further image processing.



(a) (b) (c)

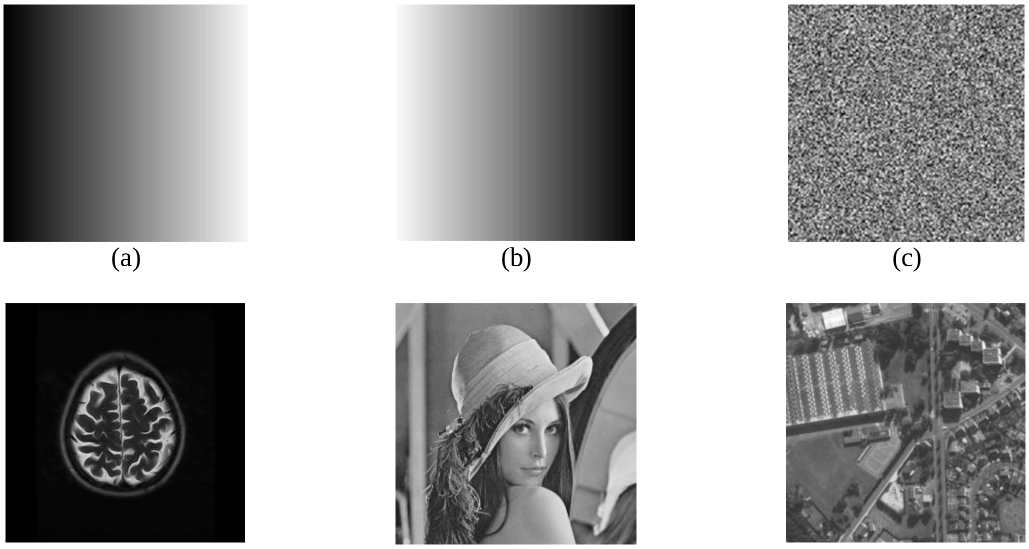
Figure 5. Region growing process for finding local extrema, (a) Extending mask in raster scan order, (b) Stacks of coordinates for temporary storages, (c) The matching of pixels in the original image and segmentation matrix

3. RESULTS AND DISCUSSION

3.1. Experiment results

The proposed algorithm ENMS is compared with some other known algorithms: the straightforward [9, 11], the spiral scan order [10], scanline3×3 [11] and the gray-scale morphology algorithm [12]. Besides, the Matlab's built-in functions *imregionalmax*, *imregionalmin*, *imdilate* and *imerode* were used for the morphology implementation using Matlab ®Image Processing Toolbox (R2015a).

Six gray-scale images were used in this experiment: worst and best are 256×256 images of worst and best-case scenarios for the straightforward algorithm; noise is a 256×256 image of uniformly distributed noise; 256×256, 512×512 and 1024×1024 refer respectively to Harris corner [3] images of the Brain, Lena and City square image of the corresponding size as shown in Figure 6.



(a) (b) (c)

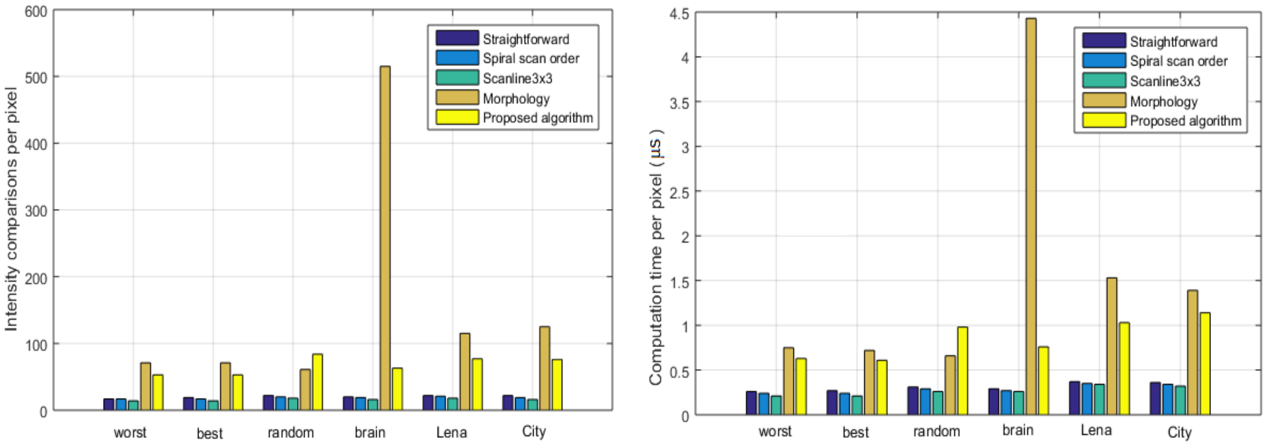
(d) (e) (f)

Figure 6. Test grayscale images, (a) Worst case, (b) Best case, (c) Random, (d) Brain, (e) Lena, (f) City

The number of operations per pixel and the average runtime over 10 executions were recorded for each test image on an Intel 3.2 GHz system with 6 GB of RAM, the result of which is shown in Figure 7, Figure 8, Figure 9, and Table 1. The execution time shown in Figure 7(b) is generally in agreement with the algorithm complexity in Figure 7(a). On average, our method runs at a speed of 0.85µs per pixel. This is 1.9 times faster than the morphology method. Besides, the morphology method is only faster than ours in its random scenario.

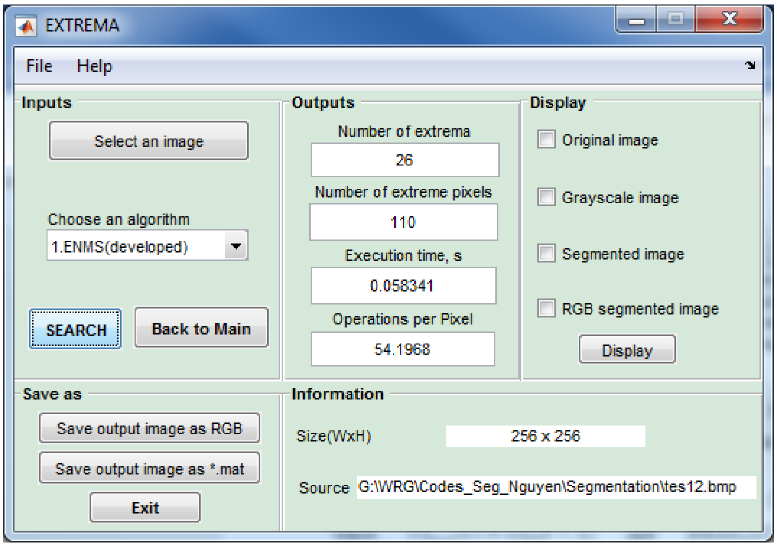
Moreover, our algorithm implemented in pure Matlab code is almost faster than Matlab's built-in functions *imdilate* and *imerode* written in C++. The execution time of our algorithm is quite stable over a wide range of image contents and sizes. It is also independent of the number of detected local extrema. The proposed algorithm can run faster for finding local extrema in low-frequency images as shown in Figure 6(d).

When using computing platforms Intel Core i3 3.1GHz, 6GB of RAM, Windows7, Matlab2015a (IWM platform), Intel Core i3 3.1GHz, 6GB of RAM, Windows7, Open CV 2.x, C ++ (IWC platform), Raspberry Pi 3, ARM-A53, Linux, Open CV 2.x, C ++ (RLC platform in Figure 10) the algorithm ENMS in comparison with Scanline3x3 algorithm requires 3.3, 2.9, 3.4 times more processing time and 5.8 times more use of RAM, respectively. However, implementation of the ENMS algorithm on IWM and IWC platforms, as compared with the implementation of the morphological algorithm, requires 1.9, 1.7 times less processing time and 2.1 times less use of RAM, respectively.



(a) (b)

Figure 7. Comparison of different algorithms for 3×3 non-maximum suppression, (a) Number of operations per pixel, (b) Computation time per pixel



Inputs

Choose an algorithm: 1.ENMS(developed)

Outputs

Number of extrema: 26

Number of extreme pixels: 110

Execution time, s: 0.058341

Operations per Pixel: 54.1968

Display

□ Original image

□ Grayscale image

□ Segmented image

□ RGB segmented image

Save as

Information

Size(WxH): 256 x 256

Source: G:\WRG\Codes_Seg_Nguyen\Segmentation\tes12.bmp

Figure 8. User interfaces of ENMS algorithms using matlab 2015b

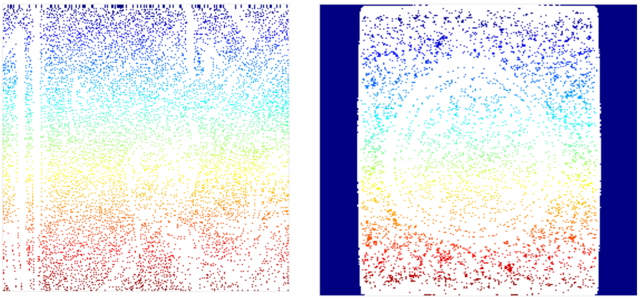


Figure 9. Example of getting segmented extreme images, (a) Lena's segmented image, (b) Brain's segmented image

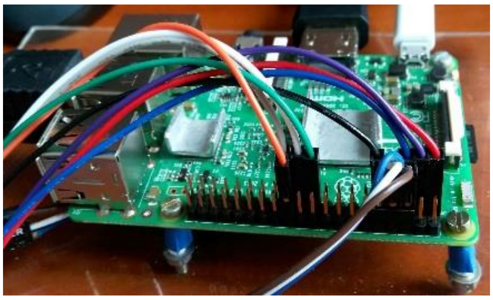


Figure 10. Implementation on raspberry Pi 3 model B

Table 1. Result of searching for local extrema

Image	Algorithm	Extrema	
		Total	Pixels
Worst case (256×256)	Straightforward [9]	0	0
	Spiral scan order at [10]		
	Scanline3×3 at [11]		
	Morphology at [12]	2	1024
	ENMS (proposed)		
Best case (256×256)	Straightforward [9]	0	0
	Spiral scan order at [10]		
	Scanline3×3 at [11]		
	Morphology at [12]	2	1024
	ENMS (proposed)		
Random (256×256)	Straightforward [9]	6069	6069
	Spiral scan order at [10]		
	Scanline3×3 at [11]		
	Morphology at [12]	6498	6704
	ENMS (proposed)		
Brain (256×256)	Straightforward [9]	2392	2392
	Spiral scan order at [10]		
	Scanline3×3 at [11]		
	Morphology at [12]	3197	20729
	ENMS (proposed)		
Lena (512×512)	Straightforward [9]	24826	24826
	Spiral scan order at [10]		
	Scanline3×3 at [11]		
	Morphology at [12]	28581	33612
	ENMS (proposed)		
City (1024×1024)	Straightforward [9]	23872	23872
	Spiral scan order at [10]		
	Scanline3×3 at [11]		
	Morphology at [12]	37995	58399
	ENMS (proposed)		

3.2. Extentions and applications

The ENMS algorithm can be used to find distinctive feature points in an image. To improve the repeatability of a detected corner across multiple images, the corner is often selected as a local extremum whose corners is significantly higher (or lower) than the close-by second highest (or lowest) peak [4, 23].

For some applications such as multi-view image matching, an evenly distributed set of interest points for matching is desirable. An oversupplied set of ENMS point features can be given to an adaptive non-extreme suppression process [24, 25], which reduces cluttered corners to improve their spatial distribution. Moreover, the ENMS algorithm can be used to video denoising by detecting extreme points in a video frame, aligning these points to estimate global shift, and average aligned video frames are used to improve Signal-to-Noise Ratio [11].

4. CONCLUSION

A mathematical model and a new fast efficient NMS algorithm to find all extrema in low-frequency images are proposed. The proposed algorithm ENMS, as well as the morphological algorithm, allows to detect all single-pixel local extrema and extreme regions consisting of pixels with the same values. Moreover, the proposed algorithm, in comparison with the morphological algorithm, requires less than 1.7-2.1 times depending on the computing platform with averaging over image types in time and 2.1 times in use of RAM. For future work, the proposed algorithm ENMS can be used for image segmentation based on region growing. Moreover, the local extrema could be used to find the feature points in image recognition and image matching.

REFERENCES

[1] A. Rosenfeld and A. C. Kak, "Digital Picture Processing," Academic Press, New York, pp. 457, 1976.

[2] L. Kitchen and A. Rosenfeld, "Gray-Level Corner Detection," *Pattern Recognition Letters*, vol. 1, no. 2. pp. 92-102, 1982.

[3] C. Harris and M. Stephens, "Combined corner and edge detector," *Proceeding of the Fourth Alvey Vision Conference*, pp. 147-151, 1988.

[4] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[5] K. Mikolajczyk and C. Schmid, "Scale and Affine Invariant Interest Point Detectors," *International Journal of Computer Vision*, vol. 60, pp. 63-86, 2004.

[6] M. Van Herk, "Fast Algorithm for Local Minimum and Maximum Filters on Rectangular and Octagonal Kernels," *Pattern Recognition Letters*, vol. 13, no. 7, pp. 517-521, 1992.

[7] J. Gil and M. Werman, "Computing 2-D Min, Median, and Max Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 5, pp. 504-507, 1993.

[8] M. Brown, et al., "Multi-Image Matching Using Multi-Scale Oriented Patches," *Proceeding of Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 510-517, 2005.

[9] A. Neubeck and L. Van Gool, "Efficient Non-Maximum Suppression," *18th International Conference on Pattern Recognition (ICPR'06)*, pp. 850-855, 2006.

[10] M. A. Forstner and E. Gulch, "A Fast Operator for Detection and Precise Locations of Distinct Points, Corners, and Centres of Circular Features," *Proceeding of the ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, pp. 281-305, 1987.

[11] T. Q. Pham, "Non-Maximum Suppression Using Fewer than 2 Comparisons per Pixel," *International Conference of Advanced Concepts for Intelligent Vision Systems (ACIVS)*, vol. 12, pp. 438-451, 2010.

[12] P. Soille, "Morphological Image Analysis: Principles and Applications," Springer-Verleg Berlin Heidelberg, 2006.

[13] T. Lindeberg, "Scale Selection Properties of Generalized Scale-Space Interest Point Detectors," *Journal of Mathematical Imaging and Vision*, vol. 46, no. 2, pp. 177-210, 2013.

[14] T. Lindeberg, "Image Matching Using Generalized Scale-Space Interest Points," *Journal of Mathematical Imaging and Vision*, vol. 52, no. 1, pp. 3-36, 2015.

[15] H. Bay, et al., "Speeded Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346-359, 2008.

[16] T. Lindeberg, "Detecting Salient Blob-like Image Structures and their Scales with a Scale-Space Primal Sketch: a Method for Focus-of-Attention," *International Journal of Computer Vision*, vol. 11, no. 3, pp. 283-318, 1993.

[17] T. Lindeberg, "Scale-Space Theory in Computer Vision," Springer, 1994.

[18] J. Matas, et al., "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions," *Proceedings of the 13th British Machine Vision Conference*, UK, pp. 384-396, 2002.

[19] R. M. Haralick and L. G. Shapiro, "Image Segmentation Techniques," *Computer Vision, Graphics and Image Processing*, vol. 29, no. 1, pp. 100-132, 1985.

[20] K. Karu, et al., "Is There Any Texture in the Image?" *Pattern Recognition*, vol. 29, no. 9, pp. 1437-1446, 1996.

[21] Q. Zhang, et al., "A Performance Analysis for Real-time Flood Monitoring using Image-based Processing," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 2, pp. 793-803, Feb 2020.

[22] S. H. Khaleefah, et al., "Review of Local Binary Pattern Operators in Image Feature Extraction," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 1, pp. 23-31, Jul 2020.

[23] C. Schmid, et al., "Evaluation of Interest Point Detectors," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 151-172, 2000.

[24] A. Ali, et al., "Retinal Blood Vessel Segmentation from Retinal Image using B-COSFIRE and Adaptive Thresholding," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13, no. 3, pp. 1199-1207, Mar 2019.

[25] M. Brown, et al., "Multi-Image Matching using Multi-Scale Oriented Patches," *Proceeding of Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 510-517, 2005.