Towards $O\left(N^{1.5}\right)$ estimation of Quasi-Distance Transform

Danijel Žlaus*
Supervised by: Borut Žalik†

University of Maribor
Faculty of Electrical Engineering and Computer Science
Laboratory for Geometric Modelling and Multimedia Algorithms
Smetanova ulica 17, SI-2000 Maribor / Slovenia

Abstract

This paper considers an efficient implementation of Quasi-Distance Transform (QDT), which is a grayscale equivalent to distance transform. Distance transform labels all pixels from the input image according to their L_n distances to the nearest background pixel, where n denotes the distance norm. Although numerous methods exist for the estimation of distance transform for binary images, their generalization for the gray-scale images is vague as there is no clear distinction between the background and the foreground. A concept from mathematical morphology named QDT was introduced for this purpose. QDT is derived from the ultimate erosion operator, which is an extension of morphological erosion. In the context of image processing, erosion is a minima search operator using a window function, whilst ultimate erosion is an operator that outputs two images. The first contains maximum difference between successive minima obtained by gradually increasing the window function i.e. the window size, and the other contains the size of the window at which the maximum difference is registered at each pixel. They are respectively referred to as the transform function and the associated function. QDT is effectively related to the associated function of ultimate erosion. The associated function has to be adjusted to a 1-Lipschitzian continuity in a post-processing step towards finalizing the QDT. The naive implementation of QDT leads to a computational complexity of $O(N^{2.5})$, where N is the number of pixels, making it time-consuming even when processing small-sized images. An efficient method for the estimation of QDT is proposed in this paper, which has computational complexity of $O(N^{1.5})$. This is achieved by a decomposition of the searching windows and reusing the already processed data at neighboring pixels. As shown by the experiments, the proposed implementation of QDT is suitable for applications in image processing.

Keywords: Distance transform, Grayscale, Mathematical morphology, Quasi distance transform

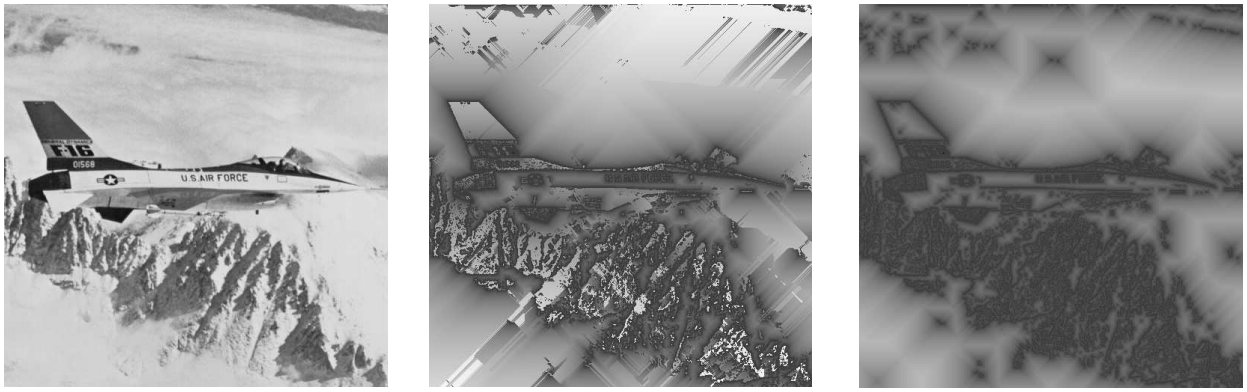
*danijel.zlaus@uni-mb.si
†zalik@uni-mb.si

1 Introduction

Distance transform (DT) is a feature-extraction operation, which typically operates on images. DT assigns to each image pixel a L_n distance value from its nearest pixel of interest, i.e. the background [1]. The commonest L_n distance metrics for distance transform are:

- Manhattan distance (L_1) also known as Taxicab or Cityblock distance, defined as $\|\vec{p}\|_1 = \sum_{i=0}^{n} |p_i|$;
- Euclidean distance (L_2) defined as $\|\vec{p}\|_2 = \sqrt[2]{\sum_{i=0}^{n} p_i^2}$;
- Chebyshev distance (L_∞) or chessboard distance, defined as $\|\vec{p}\|_\infty = \max(p_0, p_1, \cdots, p_n)$.

Numerous DT algorithms have been developed, most of which focus on the Euclidean DT [2]. However they only allow processing of binary images, where background and foreground are well defined. Since there is no obvious distinction between background and foreground on a grayscale image, a segmentation step is usually applied prior to distance transform. A recommended preprocessing step is to either amplify features of interest, be it edges, textures, or certain shades of gray, or to suppress features that are not of interest. A binary threshold operation is applied on the modified image to create a binary image. This image is then suitable for any distance transform. The described method is prone to errors, which are mainly a result of the binary threshold operation. In case of automated processing of images, this limitation requires consistent image lighting to ensure somewhat consistent results. If image lighting is dependant on the environment, binary segmentation may fail to successfully separate the foreground and background, therefore requiring manual intervention to correct threshold parameters. The Quasi Distance Transform (QDT), a concept from mathematical morphology, was introduced by S. Beucher [3]. It operates directly on grayscale images and furthermore requires no parameters, thus making it a promising candidate for automated feature extraction without the need for binary segmentation. For an image segmentation method which uses QDT see Hanbury et. al. [4].



(1.a) Input image f (1.b) $s(f)$ (1.c) QDT

Figure 1: Example of $s(f)$ and QDT. Histogram normalization was applied to images 1.b and 1.c to increase image contrast. Image 1.a was retrieved from USC-SIPI image database [5].

2 Quasi distance transform

QDT is derived from the ultimate erosion operator, which is by itself an extension of grayscale morphological erosion. For a given image f, all image pixel positions $p = (x,y)$ are contained in subset E; $p \in E$, $E \subset \mathbb{Z}^2$ and $E = \{0,1,\ldots,X\} \times \{0,1,\ldots,Y\}$. Grayscale erosion of image f with window w at each given pixel p was defined by P. Soille [6] as:

$$\big[f \ominus w\big](p) = \min_{\forall b \in w} \big(f(p+b)\big). \tag{1}$$

Thus, erosion assigns to each pixel the lowest intensity within its neighborhood as defined by window w. Ultimate erosion expands on this definition by gradually increasing the observed neighborhood w and outputting two images:

- $r(f)$ is named the transformation function and contains the maximum differences between two successive erosions registered at each pixel,
- $s(f)$ is named the associated function, which contains window sizes at which the maximum difference is registered.

For image f, the maximum image side is denoted as κ, $\kappa = \max(X,Y)$. Let w_i, $i \in [0,\kappa]$, be a square-shaped structuring element, i.e. a window of size $(2i+1) \times (2i+1)$. Ultimate erosion functions $r(f)$ and $s(f)$ are at pixel p formally defined as:

$$r(f,p) = \max_{0 \le i \le \kappa} \Big([f \ominus w_i](p) - [f \ominus w_{i+1}](p) \Big), \tag{2}$$

$$s(f,p) = \operatorname*{argmax}_{0 \le i \le \kappa} \Big([f \ominus w_i](p) - [f \ominus w_{i+1}](p) \Big). \tag{3}$$

QDT is derived from $s(f)$ by a correction of artefacts (see Figure 1.b) that appear due to the masking of features by outer, more contrast, regions [7]. This common weakness of ultimate operators prevents $s(f)$ from being Lipschitz continuous, whereas regular distance transforms with L_1, L_2, L_∞ metrics' are. During the correction step, 1-Lipschitzian continuity of $s(f)$ is forced by four passes (one from each side of the image) comparing each visited pixel with its neighboring values. If any value difference is higher than 1, a correction is applied by setting the visited pixel value to be one higher than the smallest neighboring pixel value (see Equations (4) and (5)). When all four passes are complete, the maximum differences are all at most 1 making the function 1-Lipschitzian continuous and completing the QDT (see Figure 1.c).

$$\delta(p) = f(p) - \min_{\substack{-1 \le i \le 1 \\ -1 \le j \le 1}} \big(f\big(p+(i,j)\big)\big) \tag{4}$$

$$f(p) = f(p) + \begin{cases} 1-\delta(p) & \text{if } \delta(p) > 1, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

3 Computational complexity of the naive method

A naive implementation of QDT is achieved by iteratively performing morphological erosion with all possible window sizes at each pixel. Thus, if the naive method processes a square image f (where $\kappa = X = Y$), the total number of pixels is $N = \kappa^2$. For erosion, square window w_i is used, where $i \in [0,\kappa]$. The computational complexity of processing a squared image f with this method can be summed up as a series:

$$\begin{aligned} \sum_1^{\kappa^2} \Big[\sum_{i=0}^{\kappa} (2i+1)^2 \Big] &= \sum_1^{\kappa^2} \Big[\sum_{i=0}^{\kappa} (4i^2+4i+1) \Big] \\ &= \kappa^2 \Big[\frac{1}{3} (4\kappa^3 + 12\kappa^2 + 11\kappa) \Big] \\ &= \frac{1}{3} (4\kappa^5 + 12\kappa^4 + 11\kappa^3), \end{aligned} \tag{6}$$

giving $O(\kappa^5) = O(N^{2.5})$.

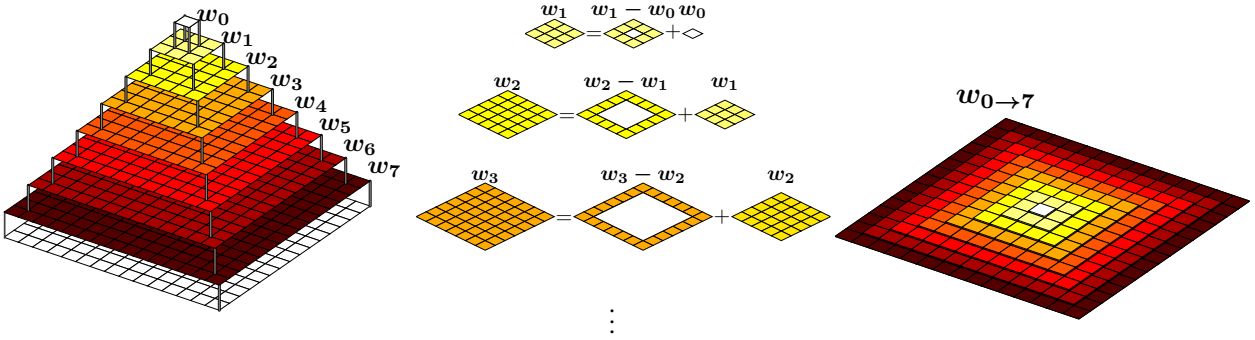


Figure 2: Using existing minima values to reduce search space

4 Carrying minima forward on window increments

An initial optimization of QDT computations can be achieved by passing the minima found within window w_i to the increment w_{i+1} at each given pixel. Thus, only the new border pixels of w_{i+1} have to be checked for potentially new minima (see Equation (7) and Figure 2).

$$\big[f \ominus w_{i+1}\big](p) = \min\Big(\big[f \ominus w_i\big](p),\ \big[f \ominus (w_{i+1} - w_i)\big](p)\Big) \tag{7}$$

This reduces the search space to searching only a single window of size κ, denoted as $w_{0\to\kappa}$. Although the optimization significantly reduces the total search-space, it still needs to be performed independently for each pixel. The computational complexity of this approach is thus a series sum of:

$$\sum_{1}^{\kappa^2}\left[\sum_{i=0}^{\kappa}\left((2i+3)^2-(2i+1)^2\right)\right] = \sum_{1}^{\kappa^2}\left[\sum_{i=0}^{\kappa} 8i+8\right]$$
$$= \kappa^2\Big[4\kappa^2+12\kappa+8\Big] = 4\kappa^4+12\kappa^3+8\kappa^2, \tag{8}$$

giving $O(\kappa^4) = O(N^2)$.

5 Extending minima-sharing to neighboring windows

To further improve the computational complexity, the search-space has to be further reduced. The per pixel computation of the method described in Section 4 is $\mathrm{O}\big(N\big)$. Thus, the per pixel computational complexity should be improved by reducing the dependency of window w from image f. The proposed method achieves this by sharing minima between the windows of neighboring pixels to decrease image lookups. There are four movement directions assuming a 4-connected image, i.e. four potential neighbors with whom information sharing can be done. The information sharing approach can be generalized by describing it for only one of the four neighbors. The remaining three are obtained by rotating the image by 90/180/270 degrees. The rotation transforms the respected movements to the generalized one as can be seen in Figure 3. The proposed method works on 4-connected images, though it can be extended to a higher connectivity.

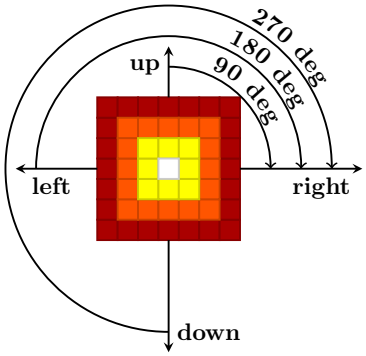


Figure 3: Generalizing window moves

6 Decomposition of nested windows and examination

In order to provide a clearer description of the information sharing process, the incremental window $w_{0\to\kappa}$ from Section 4 is further decomposed into eight segments at each scale: diagonal segments $S_i^{\{1,3,5,7\}}$ and the segments $S_i^{\{2,4,6,8\}}$ between the diagonals (see Figure 4). The structure is referred to as the clover structure, due to its resemblance to four-leaf clovers. Two windows are used by the proposed method, a constructed window w^p and a neighboring window w^{p+1} still in construction (positions p, p_{+1} are (x,y), $(x+1,y)$, respectively). Since windows w^p and w^{p+1} and image f all contain N elements, the space complexity of the method is $\mathrm{O}\big(3N\big)$. However, this allows for three unique information sharing processes, when w^p and w^{p+1} are overlapped as in Figure 5. These processes are

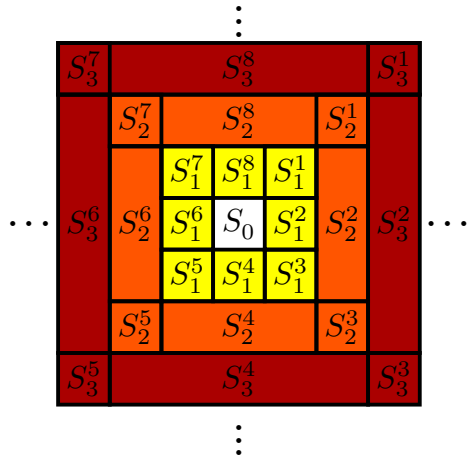


Figure 4: Clover structure

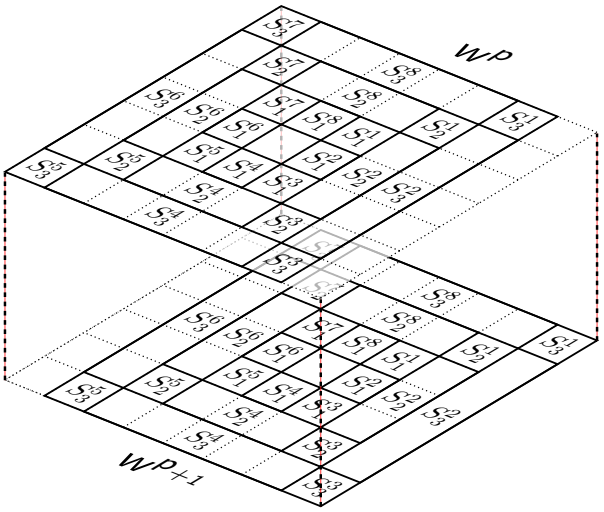


Figure 5: Overlapping of w^p and w^{p+1}

distinguished by the sections of clover structure they operate on. The sharing processes are:

- Back-sharing; processes trailing segments $S_i^6(w^{p+1})$, i.e. the left segments of the clover structure. The new segment minima are given as the minima from segments $S_{i-1}^{\{5,6,7\}}(w^p)$. In this case, a new segment minima is always found in the constant time.
- Side-sharing; processes the top and bottom segments $S_i^{4/8}(w^{p+1})$ of the clover structure. The new segment minima are given as the minima from segments $S_i^{\{4,3\}/\{8,1\}}(w^p)$. When the new minima $S_i^{4/8}(w^{p+1})$ is equal to diagonal minima $S_i^{5/7}(w^{p+1})$ the segment has to be rescanned to verify, whether the found minima is correct. If different minima is found in rescan, it replaces minima of $S_i^{4/8}(w^{p+1})$.
- Front-sharing; processes the right segments $S_i^2(w^{p+1})$ of the clover structure. The new segment minima is given as the minima from segment $S_{i+1}^2(w^p)$. When the new minima $S_i^2(w^{p+1})$ is equal to diagonal minima $S_i^1(w^{p+1})$ or $S_i^3(w^{p+1})$, the segment has to be rescanned and minima verified (as in side-sharing).

The segments $S_i^{\{1,3,5,7\}}(w^{p+1})$ are used to assist minima-sharing in segments $S_i^{\{2,4,6,8\}}(w^{p+1})$, since segments $S_i^{\{1,3,5,7\}}(w^{p+1})$ are always invalidated due to only one contained element. The new diagonal values are trivially found by $\mathrm{O}(1)$ lookup of pixel values. There is a total 4κ diagonal elements in a decomposed $w_{0\to\kappa}$, making the diagonal construction $\mathrm{O}(4\kappa) = \mathrm{O}(\kappa)$. The back-sharing is the smallest minima of the three existing segment minima: $S_i^6(w^{p+1}) = \min\left(S_{i-1}^{\{5,6,7\}}(w^p)\right)$. The series sum for computation complexity is thus:

$$\sum_1^\kappa 2 = 2\kappa = \mathrm{O}(\kappa). \tag{9}$$

The side-sharing is the selection of the smallest minima of two segments from w^p, if the found minima is not contained in the tail, i.e. the element of the segment in w^p, which is a diagonal segment in w^{p+1}.

$$S_i^4(w^{p+1}) = \begin{cases} min\left(S_i^{\{4,3\}}(w^p)\right) & \text{if } min\left(S_i^{\{4,3\}}(w^p)\right) \neq \\ & \quad S_i^5(w^{p+1}), \\ \text{rescan subspace} & \text{otherwise} \end{cases} \tag{10}$$

$$S_i^8(w^{p+1}) = \begin{cases} min\left(S_i^{\{8,1\}}(w^p)\right) & \text{if } min\left(S_i^{\{8,1\}}(w^p)\right) \neq \\ & \quad S_i^7(w^{p+1}), \\ \text{rescan subspace} & \text{otherwise} \end{cases} \tag{11}$$

A subspace scan is a minima search though all the elements, which segment $S_i^d(w^{p+1})$ represents.

The front-sharing preserves the minima, if it is not contained within the new diagonal elements:

$$S_i^2(w^{p+1}) = \begin{cases} S_{i+1}^2(w^p) & \text{if } S_{i+1}^2(w^p) \notin \\ & \quad S_i^{\{1,3\}}(w^{p+1}), \\ \text{rescan subspace} & \text{otherwise} \end{cases} \tag{12}$$

For the front-sharing, the largest segment $S_i^2(w^{p+1})$ is always rescanned, since it does not overlap with any segment in w^p. The proposed method thus avoids unnecessary subspace scans by sharing and combining minima, where possible. Each time a subspace needs to be rescanned, the advantage of the proposed method is reduced. Thus, in order to determine the computational complexity of the proposed method, the probability of rescanning needs to be determined. The probability for a segment S_i^d is denoted as $\Pr(S_i^d)$. For side-sharing, the chance of the minima being in the element removed from the segment, i.e. the tail, is inversely-proportional to the number of elements in the

segment. The number of elements in a segment S_i^d is denoted by $\|S_i^d\|$. Thus:

$$\Pr(S_i^{4,8}) = \|S_i^{4,8}\|^{-1}. \quad (13)$$

For example, using Equation (13), for S_1^8, S_2^8, S_3^8 the probabilities are $1^{-1}(100\%)$, $3^{-1}(33.\bar{3}\%)$, $5^{-1}(20\%)$, respectively. The average computational complexity for all segments $S_i^{4,8}$ in the structure is obtained as:

$$2\sum_{i=1}^{\kappa}\left[(2i+1)(2i+1)^{-1}\right] = 2\sum_{1}^{\kappa} 1 = 2\kappa = \mathrm{O}(\kappa). \quad (14)$$

For the front-sharing, the probability is also inversely-proportional to the number of elements, making it:

$$\Pr(S_i^2) = \frac{\|S_{i+1}^2\| - \|S_i^2\|}{\|S_{i+1}^2\|} = \frac{2}{\|S_{i+1}^2\|}. \quad (15)$$

For example, using Equation (15) for S_1^2, S_2^2, S_3^2, the probabilities are $2*3^{-1}$ $(66.\bar{6}\%)$, $2*5^{-1}$ (40%), $2*7^{-1}$ (28.57%), respectively. Adding the probability to the computational complexity, the series sum for the number of visited elements is:

$$\sum_{i=1}^{\kappa}\left[\frac{(2i+3)-(2i+1)}{2i+3}(2i+1)\right] = \sum_{i=1}^{\kappa}\frac{4i+2}{2i+3} \approx \sum_{i=1}^{\kappa}\frac{4i}{2i} = \sum_{1}^{\kappa} 2 = 2\kappa = \mathrm{O}(\kappa). \quad (16)$$

The simplification step in Equation 16 is reasonable, given $\lim_{i\to\infty}\frac{4i+2}{2i+3} = \frac{4i}{2i}$.

The common computational complexities for processing one neighboring window, i.e. pixel, is on average $\mathrm{O}(4\kappa+3*2\kappa+\kappa) = \mathrm{O}(\kappa) = \mathrm{O}(N^{0.5})$. Given an image with N pixels, the total computational complexity for the proposed method is $\mathrm{O}(N*N^{0.5}) = \mathrm{O}(N^{1.5})$.

7 Results

For testing, randomly generated grayscale images of square size were used, with $\kappa \in [1, 1250]$. The improved naive method and the proposed method were compared. The experiments were done as a single-thread computation on an Intel Q6600 computer with 8GB DDR3 RAM. The results can be seen in Figure 6 and confirm the theoretical computational complexity of the methods. More than an order of magnitude improvement can already be noticed at image size of 300×300, as seen in Table 1. For off-line image processing, the proposed method provides a considerable speed-up. To further speed-up the proposed method, a constricted window function $w_{0->\kappa}$ can be used, where $\kappa < \max(X, Y)$. When constricting the window, its new size has to cover all objects on the image, otherwise artefacts appear.

Table 1: Spent CPU time of improved naive and proposed methods

Image side (κ)	Total pixels (N)	Improved naive (s)	Proposed method (s)
33	1,089	0.003	0.003
101	10,201	0.222	0.062
317	100,489	20.272	1.942
1,001	1,002,001	1,896.95	65.145
1,245	1,550,025	4,625.21	137.781

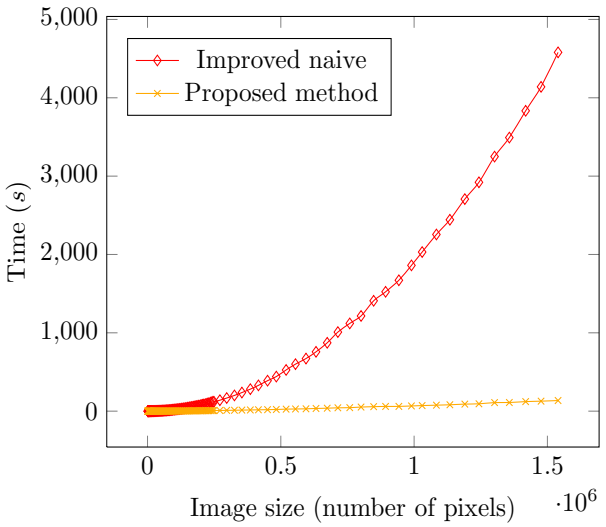


Figure 6: Spent CPU time of both methods

8 Conclusions

The proposed method, through the process of minima-sharing, preserves minima and reduces the dependency of window w on image f. Image f is thus only looked up when minima-sharing fails. In this way, $\mathrm{O}(N^{1.5})$ computational complexity has been obtained. Thus, QDT becomes suitable for processing larger images within a reasonable time.

References

[1] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing.," *Journal of the ACM*, vol. 13, pp. 471–494, 1966.

[2] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2D Euclidean distance transform algorithms: A comparative survey," *ACM Computing Surveys*, vol. 40, pp. 1–44, 2008.

[3] S. Beucher, "Numerical residues," *Image and Vision Computing*, vol. 25, pp. 405–415, 2007.

[4] A. Hanbury and B. Marcotegui, “Morphological segmentation on learned boundaries,” *Image and Vision Computing*, vol. 27, pp. 480–488, 2009.

[5] “USC-SIPI image database.” URL: http://sipi.usc.edu/database/?volume=misc&image=13.

[6] P. Soille, *Morphological Image Analysis: Principles and Applications*. Springer, 2nd ed., 2004.

[7] J. Hernandez and B. Marcotegui, “Shape ultimate attribute opening,” *Image and Vision Computing*, vol. 29, pp. 533–545, 2011.