Watershed Segmentation with Chamfer Metric

Vasily Goncharenko[1] and Alexander Tuzikov[2]

[1] National Center of Information Resources and Technologies, National Academy of Sciences of Belarus, Akademicheskaja 25, 220072 Minsk, Belarus
vasily@mpen.bas-net.by
[2] United Institute of Informatics Problems, National Academy of Sciences of Belarus, Surganova 6, 220012 Minsk, Belarus
tuzikov@newman.bas-net.by

Abstract. Watershed transformation is introduced as a computation in image graph of a path forest with minimal modified topographic distance in $(\mathbb{R}^+)^2$. Two algorithms are presented for image segmentation that use a metric defined by a unit neighborhood as well as a chamfer (a, b)-metric. The algorithms use ordered queues to propagate over image pixels simulating the process of flooding. Presented algorithms can be applied to gray-scale images where objects have noticeable boundaries.

1 Introduction

Watershed transformation of gray-scale images often results in better segmentation and contour detection outcomes in comparison with other methods. Image segmentation by watershed transformation belongs to the region growing methods that combine pixels according to similarity of their properties relative to the properties of their local neighbors. This method works good for images with objects characterized by brightness or color characteristics rather than texture features.

Watershed transformation of gray-scale images was firstly described by S. Beucher and C. Lantuéjoul [1] in their paper devoted to contour detection of objects on metallographic pictures. The authors adopted geographic terminology for describing the contour detection process, based on the disclosure of the areas with the greatest absolute gradient values. The image was presented as a topographic surface the drops of water fall on, stream down and come into local minima. Each local minimum has its own set of points of the surface named **catchment basin**. If a drop falls on a point belonging to a catchment basin, it will come down into a corresponding local minimum. Several catchment basins may intersect - their common points form watersheds. Formal construction of watershed points was based on the detection of points equidistant from different catchment basins lying on a given level λ.

In a later work S. Beucher [2] considered two groups of watershed transformation algorithms. The first group contained algorithms which simulate the flooding process (or immersion into water). An algorithm based on morphological operations was taken as an example. The second group was made of procedures detecting watershed points directly.

A. Kuba, L.G. Nyúl, and K. Palágyi (Eds.): DGCI 2006, LNCS 4245, pp. 518–529, 2006.

L. Vincent and P. Soille in their work [3] presented an algorithm of watershed transformation based on immersion process described in terms of morphological operations and graph theory. The immersion process used FIFO queues to propagate over the pixels. Their algorithm was faster and more exact than any other algorithm presented at the moment of their publication.

In addition to morphological approach to calculation of watershed transformation of discrete images there is another way based on presentation of image in the form of a graph and calculation a shortest path forest for given local minima. S. Beucher and F. Meyer have developed an algorithm using a queue for finding the shortest path between arbitrary nodes of the graph in the process of building a tree of shortest paths [4]. However, if the image has "flat" regions (plateau), such an algorithm will result in errors at the regions. To use the algorithm in the presence of plateaus, the image should be corrected to remove them. A. Lotufo and R. Falcão proposed an algorithm using the ordered queue which allows to solve the problems with plateaus [5]. However, their algorithm has drawbacks: one pixel might be pushed into the queue more than once.

There is a number of other approaches to calculation of watershed transformation. J. Roerdink and A. Meijster in their review article [6] have discussed two groups of watershed transformation algorithms: based on immersion, and on topographical distance calculation. They also discussed problems of parallelization of watershed transform calculation and concluded that it is hard to parallelize because of its inherently sequential nature.

Another approach to watershed transform calculation is the topological one [7]. Topological watershed transform uses a graph representation of the image and is based on a notion of "simple" topology. Such a transformation of gray-scale image results in another gray-scale image preserving topological information. Topological transform calculation is based on detection of "simple" points on each cross section of gray-scale image.

In this paper an algorithm of watershed transform calculation is based on detection of minimal cost paths. The path cost consists of two components. The first characterizes maximal intensity of pixels along the path. The second component of the cost gives the length of the "flat" path part and is determined using a chosen metric. This allowed to extend the algorithm to the case of chamfer metric.

2 Basic Notions and Definitions

Given a set of pixels $X \subseteq Z^n$, where n is image dimension, let Y be a set of intensity values of the image. **Digital n-dimension gray-scale image** is a function $f : X \to Y$ where the value $f(p) \in Y$ defines intensity of pixel $p \in X$. We will consider a case when function f is discrete. Watershed transformation is interpreted as a result of immersion of surface f into water. It is assumed that at selected local minima $m_i \in M$ of function f the holes are pierced such that water will be gradually filling up "cavities" starting from the minima.

Graph of gray-scale image f is a weighted graph $G = (X, \Gamma, f, d)$, with function f defined on the set of graph nodes. Arc (p, q) with weight $d(p, q)$ between nodes $p, q \in X$ exists iff q belongs to the neighborhood $\Gamma(p)$ of p. Hence, image graph contains information about intensity of pixels and their adjacency. For each node p of graph G the value $\eta = |\Gamma(p)|$ is called **the connectivity** of node p. For our task the connectivities of all graph G nodes are the same: for 2D images given on rectangular lattice $\eta = 4$ or $\eta = 8$, for 3D images $\eta = 6, \eta = 14$, or $\eta = 26$.

Path $\pi = (p_0, p_1, \ldots, p_k)$ from node p_0 to node p_k of graph G is an ordered sequence of pixels $(p_0, p_1, \ldots, p_k)$, such that $p_{i+1} \in \Gamma(p_i)$ and $p_i \neq p_j$ for $j \neq i,\ 0 \leq i \leq k-1$. Path $(p_0, p_1, \ldots, p_k)$ is denoted by π_k or simply π and its part $(p_i, p_{i+1}, \ldots, p_j),\ 0 \leq i \leq j \leq k$ from node p_i to node p_j in graph G is denoted by $\pi_{i,j}$. If $i = 0$, then the first index in the designation of path is omitted.

Let Π be a set of all paths in graph G. Function $\rho : \Pi \to \left(\mathbb{R}^+\right)^2$ is called **path function** of graph G if for any π_i, π_j such that π_i is a part of π_j (denoted as $\pi_i \subset \pi_j$), the following inequality is always true:

$$\rho(\pi_i) < \rho(\pi_j),$$

with $\rho(\pi_0) = (0, 0)$. In other words, appending at least one pixel to any path results in increasing its path function. Increasing path function is treated in the sense of **lexicographic ordering** in $\left(\mathbb{R}^+\right)^2$.

The value $C(p_i) = \rho(\pi_i), i \leqslant k, \pi_i \subseteq \pi_k$ is called **the cost of pixel** p_i on the path π_k. The value $C(p_i)$ is a vector in $\left(\mathbb{R}^+\right)^2$. The first and second coordinates of vector $C(p_i)$ are denoted by $C_x(p_i)$ and $C_y(p_i)$ respectively.

The distance $\delta(p, q)$ between any pixels $p, q \in X$ is defined as follows:

$$\delta(p, q) = \min_{\pi \in \Pi_{p,q}} \left\{\rho(\pi)\right\},$$

where $\Pi_{p,q}$ is a set of all paths between pixels p and q in graph G.

Let M be a set of selected pixels (markers) of graph G, $|M| > 1$, and I_M is a set of marker indices. **Catchment basin** $CB(m_i)$ of marker $m_i \in M, i \in I_M$ is defined as a set of points $x \in X$ which satisfy the following condition:

$$CB(m_i) = \left\{p \in X | \delta(m_i, p) < \delta(m_j, p), i \neq j\right\}.$$

Watershed $W(G)$ of graph G is a set of points not belonging to any catchment basin:

$$W(F) = X \setminus \left(\sum_{m_i \in M} CB(m_i) \right).$$

Catchment basins and watershed points define a segmentation of image. For simplicity it is possible to include watershed pixels into the relevant catchment basins in order to get segmentation just by catchment basins. Usually markers $m_i \in M$ are chosen as representative pixels of image objects as well as the background (i.e. not belonging to any object).

It is clear that catchment basins are Voronoi cells of the selected local minima. Therefore standard algorithms developed for Voronoi cells construction can be used for building catchment basins.

Let G be a graph of image f with selected markers $m_i \in M, i \in I_M$ and ω is an arbitrary label, $\omega \notin I_M$. **Watershed transformation** of image f is a mapping $\lambda : X \to I_M \cup \{\omega\}$ such that $\lambda(p) = i$ if $p \in CB(m_i)$, and $\lambda(p) = \omega$ if $p \in W(f)$.

3 Path Function

Choosing a path function having maxima at edges between different source image objects is an important task. It is based on the assumption that all objects of source gray-scale image f differ in intensity level, and there is a noticeable leap of intensity at edges between different objects.

As a simplest example of path function one can take a function $l(\pi)$ defined as follows:

$$l(\pi) = \left(0, \sum_{i=0}^{k-1} d(p_i, p_{i+1})\right),$$

where $p_{i+1} \in \Gamma(p_i)$. The function $l(\pi)$ is called **the length of flat path** π.

A drawback of the function $l(\pi)$ is that it does not depend on intensity values of pixels p_i lying at the path π and considers image f as being flat. Therefore, function $l(\pi)$ can not be used as a criterion to join pixels into one region based on their intensities.

Topographical distance takes care of relief of image [8]. Since path function should have maxima for pixels lying on edges between objects, **a gradient image** $|\nabla f|$ is often used instead of original one. Every pixel of the gradient image has intensity value equal to the absolute gradient value of the source image in this pixel. The value of absolute gradient of each pixel can be calculated based on discrete approximation. Later on the gradient image obtained from the source image f will be denoted by ψ.

The absolute gradient value can be approximated as follows:

$$\psi(p) = \max_{\gamma_p \in \Gamma(p)} \left\{ \left| f(p) - f(\gamma_p) \right| \right\}.$$

Watershed transformation of source image f can be computed in one of the following ways:

1. Transform source image f into gradient image ψ, and then make the watershed transform on gradient image ψ.
2. Make watershed transform directly on the source image f. The approximation of absolute gradient value $\psi(p)$ is calculated for each pixel p in the process of watershed transformation.

Regardless of the selected way the value $\psi(p)$ can be treated as an intensity of pixel p of the gradient image. Therefore one can build graph G of the image

ψ. All definitions that use image f are correct for gradient image ψ too. Taking this into account, we assume that graph G is built for image ψ unless otherwise specified.

Topographical length $l_\psi(\pi_k)$ of path π in G is defined as follows:

$$l_\psi(\pi_k) = \begin{cases} (0,0), & \text{if } k = 0, \\ \left(\max_{i=0,\ldots,k}\left\{\psi(p_i)\right\}, \sum_{i=0}^{k-1} d(p_i, p_{i+1})\right), & \text{if } k > 0. \end{cases} \quad (1)$$

For $k > 0$ the expression (1) can be formulated as follows:

$$l_\psi(\pi_k) = \left(\max_{i=0,\ldots,k}\left\{\psi(\pi_i)\right\}, 0\right) + l(\pi_k).$$

Gradient image usually has local maxima on object edges. At areas where maximal pixel intensity is constant, topographical length increases due to the second coordinate.

Topographical distance $T_\psi(p,q)$ between pixels $p, q \in X$ is equal to topographical length of the shortest path π from p to q in X. The lengths are compared in lexicographic sense.

The definition of topographical distance as a vector value allows to calculate correctly the distance between points in images with arbitrary intensities. The original definition of the topographical distance as a scalar value introduced in [8] was restricted to a class of images with non-marked pixels having at least one neighbor with lower intensity value. Such images do not contain non-minimal flat regions (plateaus).

As it was stated above, the watershed points are located on the same topographical distance from markers of at least two different objects. If topographical length l_ψ is used as path function, the watershed points are not always located on the place with maximal absolute gradient value indicating the edge of objects. Therefore we introduce a notion of a modified topographical distance which allows to create a set of watershed points located on the areas with maximal absolute gradient value.

Modified topographical length $l'_\psi(\pi_k)$ of path π_k is a path function defined as follows:

$$l'_\psi(\pi_0) = (0,0);$$
$$l'_\psi(\pi_1) = \begin{cases} \big(\psi(p_1), 0\big), & \text{if } \psi(p_1) > \psi(p_0); \\ \big(\psi(p_0), d(p_0,p_1)\big), & \text{if } \psi(p_1) \leqslant \psi(p_0). \end{cases}$$

And for $1 < i \leqslant k$:

$$l'_\psi(\pi_i) = \begin{cases} \big(\psi(p_i), 0\big), & \text{if } \psi(p_i) > \max_{j=0,\ldots,i-1}\left\{\psi(p_j)\right\}; \\ l'_\psi(\pi_{i-1}) + \big(0, d(p_{i-1}, p_i)\big), & \text{if } \psi(p_i) \leqslant \max_{j=0,\ldots,i-1}\left\{\psi(p_j)\right\}. \end{cases}$$

The expression for $l'_\psi(\pi_i), 0 < i \leqslant k$ can be rewritten in the following form:

$$l'_\psi(\pi_i) = \left(\max_{j=0,\ldots,i}\left\{\psi(p_j)\right\}, 0\right) + l(p_i, p_{i-1}, \ldots, p_{i-n}).$$

Here n is calculated as follows:

$$n = \max\left\{ t \mid 0 \leqslant t \leqslant i, \psi(p_{i-t}) = \max_{s=0,\ldots,i} \{\psi(p_s)\} \right\}.$$

Hence, p_{i-t} is the first pixel of the path π_k which has maximal intensity value along the whole path.

For $0 < i \leqslant k$ the following relation is true:

$$C(p_i) = l'_\psi(\pi_i) = \begin{cases} \big(\psi(p_i), 0\big), & \text{if } \psi(p_i) > C_x(p_{i-1}), \\ C(p_{i-1}) + \big(0, d(p_{i-1}, p_i)\big), & \text{if } \psi(p_i) \leqslant C_x(p_{i-1}). \end{cases}$$

Modified topographical distance $T'_\psi(p, q)$ between arbitrary pixels p and q of graph G is equal to the modified topographical length of the shortest path π from p to q in G. Here like above the shortest path is treated in the lexicographic sense.

4 Algorithms Implementing Watershed Transformation

The algorithms implementing watershed transformation (see algorithms 1 and 2) define the shortest paths between arbitrary nodes of image graph using ordered queues [9, 10]. Now let us define operations with ordered queues. The operation $Enqueue\big(p, C(p)\big)$ pushes pixel p into the tail of queue with priority $C(p)$. The operation $DequeueMin()$ pops one pixel from the head of the first non-empty queue with minimal priority.

Let us discuss the algorithm of gray-scale image segmentation using a simple metric (the distances to adjacent pixels equal to 1). Priorities of the queues in this algorithm are scalars because ordered queues automatically sort pixels lying along every path by the second coordinate of their costs: the closer (in the sense of flat path) a pixel is from the object marker, the sooner it will be pushed into a queue with priority equal to the first coordinate of its cost, and the earlier it will be popped from the queue as compared to other pixels with the same priority.

The algorithm starts from the initialization step. At this step each marked pixel $p \in M$ gets a unique label $\lambda(p) > 0$ of the object it belongs to, and the cost $C_x(p) = 0$. After that the pixel is pushed into a queue with priority $C_x(p)$ (see lines 9-13 of algorithm 1). Each not marked pixel $p \notin M$ gets cost $C_x(p) = \infty$ and label $\lambda(p) = 0$. Not marked pixels are not pushed into queue (see lines 5-8). At the initialization step all pixels of graph G are considered to be non-examined, so they have flag $TEMP$ (see lines 2-4).

At the second step (propagation) one pixel p is popped from the head of the first non-empty queue with minimal priority by $DequeueMin()$ operation (see line 16). If it has flag $TEMP$ (not examined before), then it already has the minimal cost at the path from the marker, so it is now considered to be examined and gets flag $DONE$ (see lines 17-18). One distinction of the proposed algorithm from the Lotufo-Falcão algorithm [5] is that in the algorithm 1 each pixel being popped from the queue is checked to have flag $TEMP$. Therefore a not examined

Algorithm 1. Gray-scale image segmentation by watreshed transformation (using a metric defined by a unit neighborhood)

Require: Graph G of gradient image ψ; a set of markers M with indices in I_M. The priorities are scalars.

```
 1: {Initialization}
 2: for all p ∈ X do
 3:    flag(p) ⇐ TEMP;
 4: end for
 5: for all p ∉ M do
 6:    C_x(p) ⇐ ∞;
 7:    λ(p) ⇐ 0;
 8: end for
 9: for all p ∈ M do
10:    C_x(p) ⇐ 0;
11:    Enqueue(p, C_x(p));
12:    λ(p_i) ⇐ i, i ∈ I_M, i > 0;
13: end for
14: {Propagation}
15: while Queue not empty do
16:    p ⇐ DequeueMin();
17:    if flag(p) == TEMP then
18:       flag(p) ⇐ DONE;
19:       if λ(p) ≠ w then
20:          for all neighbor q of p and flag(q) == TEMP do
21:             C'_x(q) ⇐ max{C_x(p), ψ(q)}
22:             if C'_x(q) < C_x(q) then
23:                C_x(q) ⇐ C'_x(q);
24:                λ(q) ⇐ λ(p);
25:                Enqueue(q, C_x(q));
26:             else
27:                if C'_x(q) == C_x(q) and λ(q) ≠ λ(p) then
28:                   λ(q) ⇐ w, w ∉ I_M;
29:                end if
30:             end if
31:          end for
32:       end if
33:    end if
34: end while
```

pixel may have several "copies" in the queue. A pixel popped in that way will always have the minimal cost and, consequently, there is no necessity to check the queue for the presence of its "copy" each time it is pushed into the queue, as it is done in Lotufo-Falcão algorithm. If pixel p does not belong to a set of watershed points, then each its neighbor $q \in X$ having flag $TEMP$ is determined (see lines 19,20). Then the cost of each pixel q is calculated relative to the path passing through pixel p: $C'_x(q) = \max\{C_x(p), \psi(q)\}$ and if pixel q had cost $C'_x(q) < C_x(q)$, it gets a new cost $C_x(q) = C'_x(q)$ and label $\lambda(p) \in I_M$ showing that pixel q now belongs to the same object as pixel p. Besides, pixel q is pushed

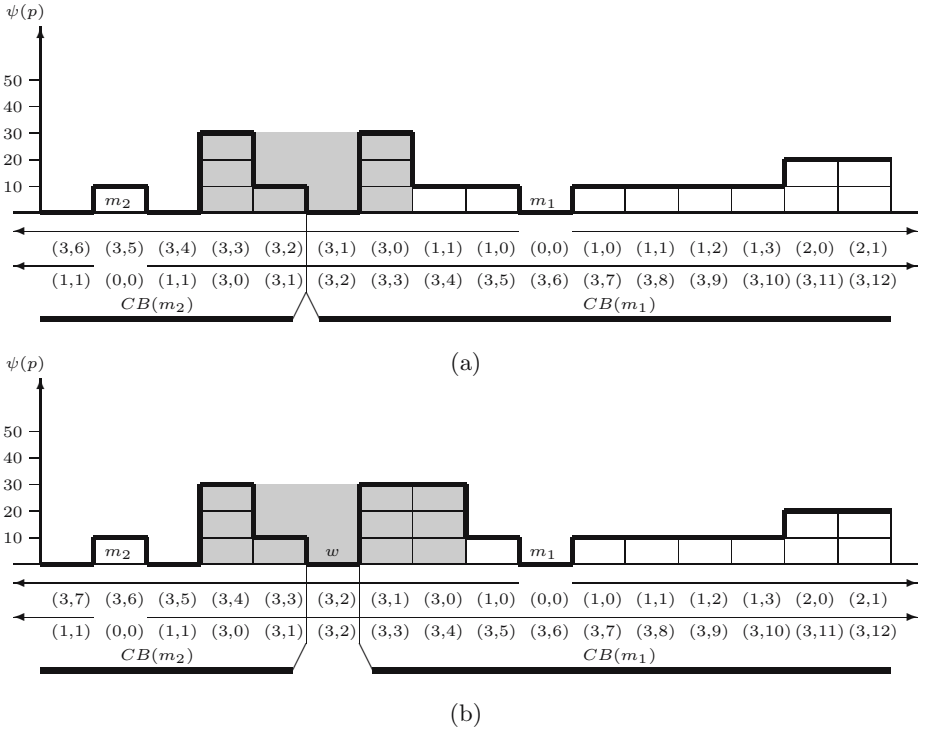


Fig. 1. Catchment basins for two selected markers using modified topographical distance T'_ψ. Gradient image ψ is shown by bold line. Modified topographical distance from marker m_1 to other pixels (top row) and from marker m_2 to other pixels (lower row) is shown in the brackets.

into the tail of queue with priority $C'_x(q)$ (see lines 21-25). If pixel q has been already pushed, the first "copy" of this pixel being popped from the queue has the least priority (and therefore the least cost). All other "copies" of this pixel having greater priority when popped from the queue will already have the flag $DONE$ and will not be examined (see line 17).

If pixel q is located on the same distance from markers of different catchment basins and has cost $C'_x(q) = C_x(q)$ and label $\lambda(p) \neq \lambda(q)$, then this means that the pixel belongs to a set of watershed points and gets new label $w \notin I_M$ (see lines 26-29). The algorithm finishes when the queue is empty. Each pixel p having label $\lambda(p) \in I_M$ belongs to the object corresponding to this label. If graph G is connected, all pixels not belonging to the watershed should have the label $\lambda(p) \in I_M$. If pixel p has label $\lambda(p) = w \notin I_M$, it belongs to the set of watershed points.

The set of watershed points, that is built by the proposed algorithm, not necessarily forms a closed contour lying at object edges (see fig. 1). If width of the segment of pixels with maximal cost (emphasized by gray color at the fig.1) is even, then there is no watershed point, since no pixels of the path lie on the

Algorithm 2. Gray-scale image segmentation by watreshed transformation (using chamfer (a, b)-metric)

Require: Graph G of gradient image ψ; a set of markers M with indices in I_M. The priorities are 2D vectors.

```
{Initialization}
for all p ∈ X do
    flag(p) ⇐ TEMP;
end for
for all p ∉ M do
    C(p) ⇐ (∞, ∞);
    λ(p) ⇐ 0;
end for
for all p ∈ M do
    C(p) ⇐ (0, 0);
    Enqueue(p, C(p));
    λ(p_i) ⇐ i, i ∈ I_M, i > 0;
end for
{Propagation}
while Queue not empty do
    p ⇐ DequeueMin();
    if flag(p) == TEMP then
        flag(p) ⇐ DONE;
        if λ(p) ≠ w then
            for all neighbor q of p and flag(q) == TEMP do
                if ψ(q) > C(p) then
                    C'(q) ⇐ (ψ(q), 0);
                else
                    C'(q) ⇐ C(p) + (0, d(p, q));
                end if
                if C'(q) < C(q) then
                    C(q) ⇐ C'(q);
                    λ(q) ⇐ λ(p);
                    Enqueue(q, C(q));
                else
                    if C'(q) == C(q) and λ(q) ≠ λ(p) then
                        λ(q) ⇐ w, w ∉ I_M;
                    end if
                end if
            end for
        end if
    end if
end while
```

same distance from both markers m_1 and m_2 (see fig.1a). The path illustrated in the fig. 1b includes the watershed point lying on the same distance from the both markers. This feature of the algorithm makes information about watershed points almost useless for extraction of object contours. Watershed points are extracted merely to show that they may belong to any adjacent object. Therefore, user can

put them to any object at his discretion using some extra information contained in the image. If watershed points are unnecessary, the algorithm can be modified not to extract them, but put them to any adjacent catchment basin. It can be done by just removing the lines 19, 26-29, 32 of the algorithm 1 and substituting "$<$" by "$\leqslant$" in the line 22.

Let us consider the algorithm of gray-scale image segmentation using chamfer (a, b)-metric (see fig. 2). In this case the priorities are vectors, because the distances to adjacent pixels are different and, therefore, different pixels get different second coordinate of the cost value. The segmentation algorithm (see algorithm 2) is similar to the algorithm 1. The only distinction is in using vector costs of pixels and increasing the second coordinate of the cost in accordance with chamfer (a, b)-metric.

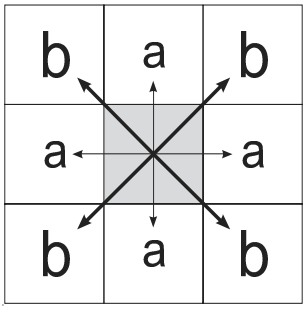


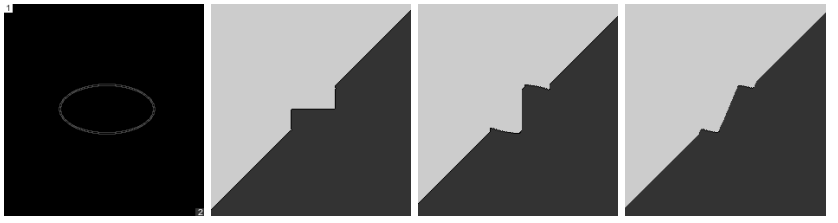
Fig. 2. The illustration of chamfer (a, b)-metric for two-dimensional rectangular lattice. Central pixel is emphasized with gray color. There are two groups of neighbor pixels: the distance to one group is a, and that to another group is b.

If watershed points are unnecessary, the algorithm 2 can be modified not to extract them, but put them to any adjacent catchment basin. It can be done by removing the lines 19, 30-33, 36 of the algorithm 2 and substituting "$<$" by "$\leqslant$" in the line 26.

5 Algorithm Discussion

Fig. 3 shows the results of a synthetic image segmentation containing an ellipsis against a homohenious background. This figure illustrates "the reaction" of algorithms to "the noise". One can see that chamfer metric gives better results as compared with the metrics defined by a singular vicinity. Ideally the image should have been divided into halves.

Fig. 4 shows the result of segmentation of a 3D tomographic image of pelvis given by algorithm 1. Markers are selected automatically where intensities exceed some threshold value. The time of segmentation of the image with 512*512*136



(a) (b) (c) (d)

Fig. 3. The result of segmentation by the proposed algorithms
(a) Gradient image to be segmented. Markers are noted with numbers at image corners.
(b) Result of segmentation by algorithm 1 using "city block" metric.
(c) Result of segmentation by algorithm 1 using "chessboard" metric.
(d) Result of segmentation by algorithm 2 using chamfer (a, b)-metric with $a = 3$, $b = 4$.

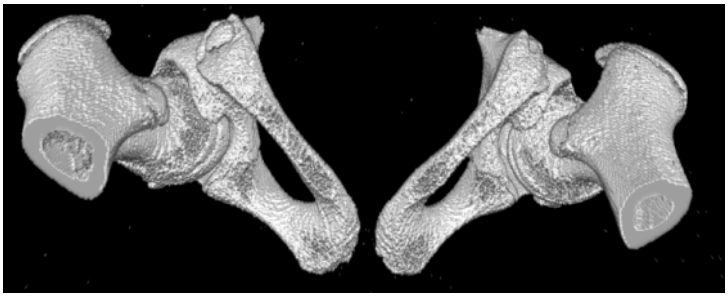


Fig. 4. CT image segmentation by a watershed algorithm

voxels, 16 bits per voxel by algorithm 1 without extracting watershed points is about 800ms. Not optimized algorithm 2 does it in about 4.5 seconds.

6 Conclusion

The paper formulates the notion of watershed transformation in terms of graph theory. Presented graph approach abstracts from specific metric and topology of the image, allowing to be applied to various data sets. The chamfer metric allows approximate more precisely Euclidean metric when calculating path lengths. Two algorithms of segmentation are presented: using a metric defined by singular neighborhood, and using chamfer (a, b)-metric. These algorithms use ordered queues permitting effective implementation. Unlike the similar algorithm proposed by Lotufo and Falcão, the presented in this paper algorithm gives more accurate result and allows extending for the Euclidean metric.

The work was partially carried out in the framework of INTAS N 04-77-7003 project.

References

1. Beucher, S., Lantuéjoul, C.: Use of Watersheds in Contour Detection. International Workshop on Image Processing, CCETT/IRISA (1979) 2.1–2.12
2. Beucher, S.: The Watershed Transformation Applied to Image Segmentation. Conference on Signal and Image Processing in Microscopy and Microanalysis (1991) 299–314
3. Vincent, L., Soille, P.: Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations. IEEE Trans. Pattern Analysis and Machine Intelligence, **13** (1991) 583–598
4. Beucher, S., Meyer, F., Dougherty, E. R.(ed.): The Morphological Approach to Segmentation: the Watershed Transformation. Mathematical Morphology in Image Processing. Marcel Dekker, New York (1993) 433–481
5. Lotufo, R., Falcao, A.: The Ordered Queue and the Optimality of the Watershed Approaches. In: Goutsias, J., Vincent, L., Bloomberg, D.S.(eds.): Mathematical Morphology and Its Applications to Image and Signal Processing, Vol. 18. Kluwer Academic Publishers, Boston Dordrecht London (2000)
6. Roerdink, J. B. T. M., Meijster, A.: The Watershed Transform: Definitions, Algorithms and Parallelization Strategies. Fundamenta Informaticae, **41** (2000) 187–228
7. Couprie, M., Bertrand, G.: Topological Grayscale Watershed Transformation. SPIE Vision Geometry V Proceedings, Vol. 3168 (1997) 136–146
8. Meyer, F.: Topographic Distance and Watershed Lines. Signal Processing, **38** (1994) 113–125
9. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2001)
10. Moore, E.: The Shortest Path Through a Maze. Proceedings of an International Symposium on the Theory of Switching, Part II. Harvard University Press, Cambridge (1959) 285–292