ELSEVIER

Image and Vision Computing 14 (1996) 47–57

image AND VISION COMPUTING

Skeletonization algorithm running on path-based distance maps

Gabriella Sanniti di Baja[a], Edouard Thiel[b]

[a]*Istituto di Cibernetica, CNR, Via Toiano 6, 80072 Arco Felice, Naples, Italy*
[b]*Equipe TIMC-IMAG, IAB-Domaine de la Merci, 38706 La Tronche cedex, France*

Received 1 November 1994; revised 22 March 1995

Abstract

A new skeletonization algorithm is introduced which runs on the distance map of a digital figure, computed according to any among four commonly used path-based distance functions. The skeletonization algorithm has a number of features: it is reversible, since it detects the centres of the maximal disks; it is nearly invariant under figure rotation (when the adopted distance function provides a reasonably good approximation to the Euclidean distance); it includes two steps (pruning and beautifying), which allow us to simplify skeleton structure according to the user's needs, as well as to improve skeleton aesthetics; and its computational load is limited, whatever the size of the figure to be skeletonized.

Keywords: Distance maps; Centres of maximal disks; Medial line; Skeleton; Pruning

1. Introduction

The labelled skeleton is generally recognized as a convenient representation of digital figures whose thickness is neither disregardable nor constant, and has been used as a tool for shape decomposition and description [1,2]. In fact, the labels attached to the pixels of the skeleton allow one to derive information on the local width of the figure and, accordingly, enlarge the number of classes of figures that can be analysed through the analysis of their skeleton.

Skeletonization algorithms based on the use of the distance map are available in the literature [3–7]. They are naturally inclined to produce skeletons whose pixels are correctly labelled according to their distance from the complement of the input figure. Although the idea to find a unique algorithm, running whichever distance function is used is quite natural, skeletonization algorithms are generally tailored to a specific distance case. One exception can be found in Dorst [4], where several path-based distances are considered; however, the set obtained [4] should not be referred to as the skeleton, since it is generally more than one pixel wide and is not guaranteed to include the centres of the maximal disks.

Path-based distance functions are often used on the discrete plane to compute the distance between two pixels as the length of the shortest (not necessarily unique) path linking them. The degree of approximation to the Euclidean distance depends essentially upon which are the unit moves permitted along the path, and on the weights used to measure them. City-block and chessboard distance functions [8] have been widely employed in the past, as they are a natural choice on the square grid. These functions roughly approximate the Euclidean distance. In fact, only horizontal/vertical moves, of weight $w_1 = 1$, are permitted by the city-block distance d_1; in turn, horizontal/vertical moves and diagonal moves, with equal weights $w_1 = w_2 = 1$, are permitted by the chessboard distance $d_{1,1}$. A closer approximation to the Euclidean distance is provided by the two-weight distance $d_{3,4}$, where the weights $w_1 = 3$ and $w_2 = 4$, respectively, measure the length of horizontal/vertical and diagonal moves; an even better approximation is given by the 3-weight distance $d_{5,7,11}$ where, besides the horizontal/vertical and diagonal moves, the knight move in chess is also permitted, and $w_1 = 5$, $w_2 = 7$ and $w_3 = 11$ are the three weights to be used for the permitted moves. The two-weight distance $d_{3,4}$ and the 3-weight distance $d_{5,7,11}$ have been introduced in Borgefors [9]. A comprehensive study of the path-based distance functions can be found in Thiel [10].

In this paper we provide a skeletonization algorithm based on the use of the distance map, which has two main novel features: it runs whatever distance function

SSDI 0262-8856(95)01039-4

is chosen from among d_1, $d_{1,1}$, $d_{3,4}$ and $d_{5,7,11}$; and it includes the pruning and beautifying steps. The first feature allows one to select the distance function more suited to the specific needs. In fact, different distance functions provide differently structured distance maps and, accordingly, different skeletons. The second feature plays a key role in making the skeleton an effective representation system. In fact, it allows us to simplify the skeleton structure and to improve skeleton aesthetics. In particular, it allows us to remove or shorten skeleton branches, keeping under control the corresponding reduction of the skeleton recoverability power. Although the algorithm could also be implemented on parallel architectures, its use is particularly convenient when working on sequential computers. In fact, another peculiarity of this algorithm (common to all the distance-based skeletonization algorithms) is that the skeleton is obtained within an a priori known and small number of raster scan inspections, whatever the maximal pattern thickness. This feature does not characterize the standard skeletonization algorithms (i.e. the algorithms based on iterated contour detection and removal) when they are implemented on sequential computers. In fact, the number of iterations (hence the number of required raster scans) increases with pattern thickness.

Skeletonization is accomplished in three phases: computation of the distance map; detection of the medial line (i.e. the set including all the centres of the maximal disks, the saddle points and the linking pixels); and reduction of the medial line to the unit-wide labelled skeleton. The third phase also includes the pruning and beautifying steps. As for the reversibility of the transformation, we note that the input figure can be faithfully recovered by applying the reverse distance transformation to the medial line, since this includes all the centres of the maximal disks; in turn, a few contour pixels are missed out if the reverse distance transformation is applied to the unit wide skeleton; finally, when starting from the pruned skeleton, a slightly smoothed version of the input figure is obtained, where the loss in recovery can be controlled by properly selecting the pruning threshold. The skeleton obtained is rather stable under figure rotation if $d_{3,4}$ or $d_{5,7,11}$ are used, since these distance functions are good approximations of the Euclidean distance.

2. Skeletonization algorithm

2.1. Definitions

In the following we indicate with $\mathbf{F} = \{1\}$ and $\mathbf{B} = \{0\}$ the two sets constituting a binary picture digitized on the square grid. To avoid topological paradoxes, we select the 8-metric for the figure **F** and the 4-metric for the background **B**. Without loss of generality, we suppose that **F** is constituted by a single 8-connected component. We also assume that a cleaning process is initially accomplished to fill all the noisy holes of **F**, which would otherwise force the creation of non-meaningful loops in the skeleton of **F**.

The distance map (DM) of **F** with respect to **B** is a replica of **F**, where the pixels are labelled with their distance from **B**. In the following, p will be used to indicate both the pixel and its associated label. Any pixel p of DM can be interpreted as the centre of a disk fitting **F**, the label of p being related to the radius length. A pixel is a centre of a maximal disk if the associated disk is maximal, i.e. is not included by any other signle disk of **F**. Centres of a maximal disk will be denoted *cmd* in the following. The figure can be recovered by the union of its maximal disks; the union of the maximal disks can be conveniently obtained by applying the reverse distance transformation to the *cmd*s.

The skeleton **S** of **F** is a subset of **F**. Accordingly, the 8-metric is selected for **S** (while the 4-metric holds for the complement of **S**), whatever distance function is used for the DM from which **S** is extracted. The skeleton is characterized by the following properties:

1. **S** has the same number of 8-connected components as **F**, and each component of **S** has the same number of 4-connected holes as the corresponding component of **F**.
2. **S** is centred within **F**.
3. **S** is the unit-wide union of simple 8-arcs and 8-curves.
4. The pixels of **S** are labelled with their distance from **B**.
5. **S** includes almost all the *cmd*s of **F** (complete inclusion is not compatible with fulfilment of property 3).

The pixels of **S** are called *end points*, *normal points* and *branch points*, respectively, depending on the number of neighbours in **S**. End points have only one neighbour in **S**; normal points have two neighbours, and branch points have more than two neighbours in **S**.

Generally, the neighbourhood $N(p)$ of a pixel p includes the neighbours of p, i.e. the pixels that can be reached with a unit move from p. Accordingly, $N(p)$ should include from a minimum of 4 pixels, when d_1 is used, up to 16 pixels, when $d_{5,7,11}$ is used. In this paper, $N(p)$ always includes the 16 pixels surrounding p, as shown in Fig. 1. The $n_i(p)$ are also called *horizontal/vertical neighbours*, for $i = \{1, 3, 5, 7\}$, *diagonal neighbours*, for $i = \{2, 4, 6, 8\}$, and *knight neighbours*, for $i = \{9, 10, \ldots, 16\}$. The neighbours of $n_i(p)$ are indicated as $n_k(n_i)$ $(k = 1, 2, \ldots, 16)$.

2.2. Phase 1: Distance map computation

The distance map can be interpreted as the result of a propagation process: a wave front originates on the contour of **F** at a given instant of time and propagates towards the inside of **F**. Every pixel p of **F** receives distance information from pixels of **F** which are closer to **B**

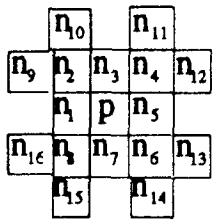

Fig. 1. The set $N(p)$.

than p, and propagates distance information to pixels of **F** which are further from **B** than p. In particular, distance information can be propagated from any p to all the pixels that can be reached from p with unit moves, so that local operations are sufficient to build the DM.

For a given distance function, the DM can be computed in two raster scans by using a well known sequential algorithm [8,9]. The algorithm is here suitably modified so that it can be used to compute the DM, whatever distance is used among d_1, $d_{1,1}$, $d_{3,4}$ and $d_{5,7,11}$. Horizontal/vertical, diagonal and knight neighbours are all taken into account, but a sufficiently high value is ascribed to the weights corresponding to the moves not permitted by the selected distance function.

The two sequential operations $f_1(p)$ and $f_2(p)$ given below are applied to every p in **F**, during a forward and a backward raster inspection of the picture, respectively:

Forward scan:

$$f_1(p) = \min\left\{ \min_{i=\{1,3\}}[n_i(p) + w_1],\ \min_{i=\{2,4\}}[n_i(p) + w_2],\ \min_{i=\{9,10,11,12\}}[n_i(p) + w_3] \right\}$$

Backward scan:

$$f_2(p) = \min\left\{ \min_{i=\{5,7\}}[n_i(p) + w_1],\ \min_{i=\{6,8\}}[n_i(p) + w_2],\ \min_{i=\{13,14,15,16\}}[n_i(p) + w_3] \right\}$$

The values to be used for the weights w_1, w_2 and w_3 in the four different cases are given in Table 1, where ∞ stands for a sufficiently high value.

The four distance maps of a small pattern, used as a running example through this paper, can be seen in Fig. 2a.

Table 1
Weight values

	w_1	w_2	w_3
d_1-DM	1	∞	∞
$d_{1,1}$-DM	1	1	∞
$d_{3,4}$-DM	3	4	∞
$d_{5,7,11}$-DM	5	7	11

2.3. Phase 2: Medial line extraction

Medial line extraction is accomplished by marking on the DM the pixels identified as centres of maximal disks, saddle points and linking pixels. Centres of maximal disks and saddle points are identified within one inspection of the DM. As soon as a pixel is marked, the inspection of the DM is temporarily interrupted, and a path growing process is done to search and mark on the DM the linking pixels. After the medial line has been extracted, a few pixels are identified and marked on the DM, so as to guarantee that the skeleton has a number of loops equal to the number of holes of the input figure. In fact, one or two pixel-wide spurious holes are possibly created in the medial line by the convergence of paths of linking pixels. Filling the spurious holes requires one raster scan inspection.

2.3.1. Centre of maximal disk detection

To check whether a pixel p is a *cmd*, the computationally expensive comparison between the disk centred on p and the disk centred on each of its neighbours is not indispensable. In fact, the structure of the path-based DM is such that the centres of the maximal disks can be generally identified by suitably comparing the label of p with that of its neighbours. In general, a pixel p is a *cmd* if for all its neighbours it results $n_i(p) < p + w_k$, where the value of w_k depends upon the relative position of p and $n_i(p)$. While this rule correctly applies on the d_1-DM and the $d_{1,1}$-DM, it may also cause on the $d_{3,4}$-DM and $d_{5,7,11}$-DM the detection of spurious centres of maximal disks, i.e. pixels whose associated disks are not maximal. These pixels, if marked, would make the skeleton structure more complex, by originating skeleton branches rid of perceptual meaning. To avoid spurious centre detection, suitable criteria have been used [6,11], respectively tailored to $d_{3,4}$-DM and $d_{5,7,11}$-DM. Although spurious centres may have only a few possible labels, a *cmd* detection method valid whichever DM is used has not yet been found.

Alternatively, the *cmd*s can be identified by means of look-up tables giving, for each value p occurring in the DM (i.e. for each radius of length p), the radii of the smallest disks which, if centred on the neighbours of p, would cover the disk associated with p. This criterion has been used in Borgefors [12], with reference to $d_{5,7,11}$-DM. Since the process necessary to build the look-up table by actually checking overlapping between disks is time consuming, one should compute in advance and memorize all the look-up-tables to be used during skeletonization. In our case, this would imply memorizing four look-up-tables.

Here we take advantage of both the above detection methods to limit memory occupation and save computation time. To limit memory occupation, we build, during

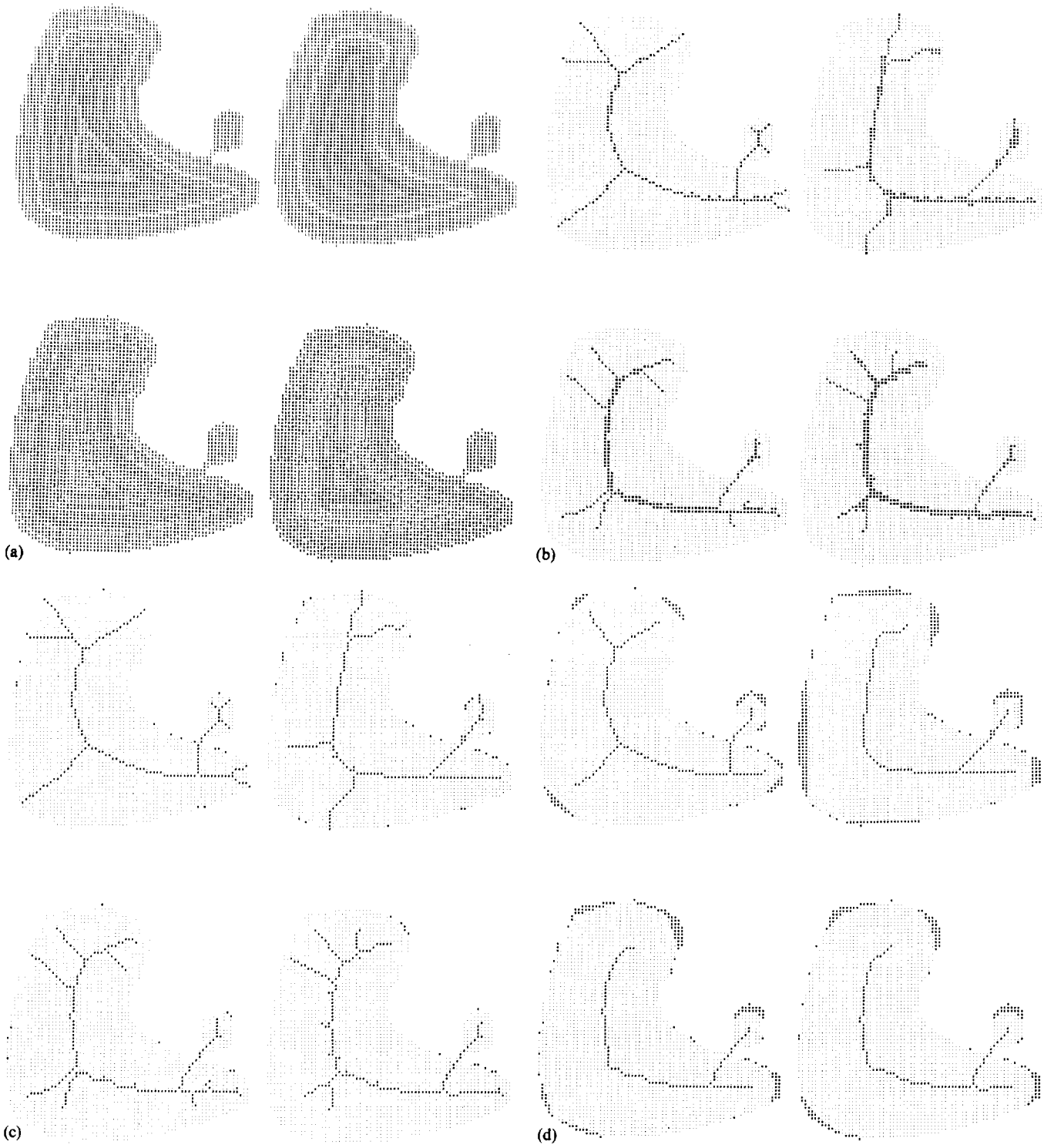

Fig. 2. Each frame refers, from top left to bottom right, to d_1, $d_{1,1}$, $d_{3,4}$ and $d_{5,7,11}$. (a) The distance map; (b) the medial line, where pixels labelled M, X, L and O, respectively, identify the *cmd*s, the saddle points which are not also centres of maximal disks, the linking pixels and the spurious holes; (c) the unit wide skeleton; S and U, respectively, identify the skeletal pixels and the pixels of the input figure which are not recovered starting from the skeleton; (d) the pruned skeleton; S and U have the same meaning as in (c).

the DM computation, only the look-up table that we actually need. The number of records of the look-up table is limited by the maximal distance value in the DM. Each record contains three fields, since at most three different types of unit move are permitted. To save computation time, we fill the fields of any record p by the values $n_i(p) = p + w_k$ (where the w_k are as in Table 1). However, since this would lead to detecting

Table 2
($d_{3,4}$-DM case)

Label	h/v	d	k
3	4	6	
6	8	9	

Table 3
($d_{5,7,11}$-DM case)

Label	h/v	d	k
5	7	10	14
7	11		
10	14	15	20
14	18	20	
16		22	
18	22		28
20		26	30
21		27	
25	28	30	35
27		33	
29	33		
31		37	
32		38	
35	39	41	45
38		44	
39		45	
40	44		
42		48	
46		52	
49		55	
53		59	
60		66	

as spurious centres a few labels in $d_{3,4}$-DM and $d_{5,7,11}$-DM, the records corresponding to these labels have to be memorized. The records to be memorized for $d_{3,4}$-DM and $d_{5,7,11}$-DM cases are, respectively, given in Tables 2 and 3, where h/v, d and k refer to the horizontal/vertical, diagonal and knight neighbours. Blank cells in these tables correspond to fields where the value $p + w_k$ is stored.

Marking of the *cmd*s on the DM is accomplished within a raster scan inspection. A pixel p is marked as a centre of a maximal disk if its horizontal/vertical, diagonal and knight neighbours are labelled less than the values orderly stored in the first, second and third fields of the record p in the look-up table. The *cmd*s found in the DM of the test pattern are denoted by M in Fig. 2b.

Inclusion of the *cmd*s in the medial line allows one to have a reversible transformation, but it is not enough to guarantee skeleton stability under figure rotation. In fact, the disk provided by discrete metrics are polygons whose sides have a fixed orientation. Thus the number of disks necessary to recover a figure subset depends upon geometry and orientation of the contour of the figure subset. Skeleton stability increases with the number of sides characterizing the disk. In this respect, disks built according to $d_{3,4}$ or $d_{5,7,11}$, respectively having 8 and 16 sides, are preferable to disks built according to d_1 or $d_{1,1}$, having only four sides.

2.3.2. Saddle point detection

The DM can be interpreted as a landscape, where the label of every pixel indicates the height of the region which is the continuous counterpart of the pixel itself. The saddle points are pixels of a ridge connecting two higher elevations. Most of the saddle points are also centres of maximal disks, so that their identification automatically occurs when the *cmd*s are searched in the distance map. The remaining saddle points are identified by using local checks that reflect the saddle point neighbourhood configuration. Detection is accomplished during the same raster scan devoted to *cmd* detection. A pixel p is marked as a saddle point if at least one of the following conditions is satisfied:

1. The set $\{n_1(p), n_2(p), \ldots, n_8(p)\}$ includes two (4-connected) components of pixels having label smaller than p.
2. The set $\{n_1(p), n_2(p), \ldots, n_8(p)\}$ includes two (8-connected) components of pixels having label larger than p.
3. The pixels of one of the triples $(n_1(p), n_2(p), n_3(p))$, $(n_3(p), n_4(p), n_5(p))$, $(n_5(p), n_6(p), n_7(p))$, $(n_7(p), n_8(p), n_1(p))$ are all labelled as p.

To count the number of 4- and 8-connected components in conditions (1) and (2), the crossing number [13] and connectivity number [14] are respectively used. Condition (3) has to be checked for all the pixels of DM when $w_1 = 1$ (i.e. in the d_1-DM and $d_{1,1}$-DM cases), only for the pixels labelled w_1, otherwise. The saddle points found in the test pattern are denoted by X in Fig. 2b.

2.3.3. Linking pixel detection

Generally, the set including the *cmd*s and the saddle points has more than one 8-connected component. Since the skeleton of a connected figure is a connected set, further pixels (called the *linking* pixels) are identified by growing paths through the ascending gradient in the DM.

As soon as a pixel p is marked during the raster scan of the DM devoted to the identification of the *cmd*s and of the saddle points, the scan is temporarily interrupted, and $N(p)$ is inspected. $N(p)$ includes at most two 8-connected components of neighbours labelled more than p, from which an ascending path can be started. For each component, the neighbour $n_i(p)$ in correspondence of which the gradient assumes the maximal value is identified (and marked) as the first linking pixel in the connecting path. Then the $n_k(n_i)$ are inspected to identify the next linking pixel in the possibly existing, unique, component of pixels labelled more than $n_i(p)$. Path growing proceeds, through the ascending gradient, as

Table 4
Weight values

	v_1	v_2	v_3
d_1-DM	2	3	−1
$d_{1,1}$-DM	2	3	−1
$d_{3,4}$-DM	3	4	−1
$d_{5,7,11}$-DM	5	7	11

far as pixels providing a positive gradient are found. The raster scan inspection of the DM is then resumed.

For any neighbour $n_i(p)$ of p, labelled more than p, the gradient is computed as:

$$\text{grad}_i = [n_i(p) - p]/v_k$$

where the values to be used for the weight v_k are given in Table 4.

Negative values are used for the weights corresponding to unit moves not permitted in the selected DM. In the d_1-DM and $d_{1,1}$-DM cases, the values of the positive v_k are different from the values of the w_k used to compute the DM; this is done both to force the creation of 8-connected paths, and to avoid path thickening and fan creation (see Fig. 3).

In $d_{3,4}$-DM, with the above values for the v_k, at most a pair of 4-adjacent $n_i(p)$ (for $i = 1, 2, \ldots, 8$) equally maximize the gradient. When this is the case, only the $n_i(p)$ with i odd is accepted in the path, to avoid path thickening.

In the $d_{5,7,11}$-DM case, three successive neighbours of p could equally maximize the gradient: a horizontal/vertical neighbour, one of the two diagonal neighbours 4-adjacent to it, and the knight neighbour which is connected to both the previous neighbours (e.g. $n_1(p)$, $n_2(p)$ and $n_9(p)$). To avoid path thickening, one should accept as linking pixel only the knight neighbour. Since this would not produce an 8-connected path, also the horizontal/vertical neighbour is accepted. For the same reason, if only a knight neighbour maximizes the gradient, the corresponding horizontal/vertical and diagonal neighbours are checked, and the pixel, out of these two, providing the largest gradient is also accepted in the path. Note that whenever the knight neighbour is accepted as a linking pixel, path growing continues only from it.

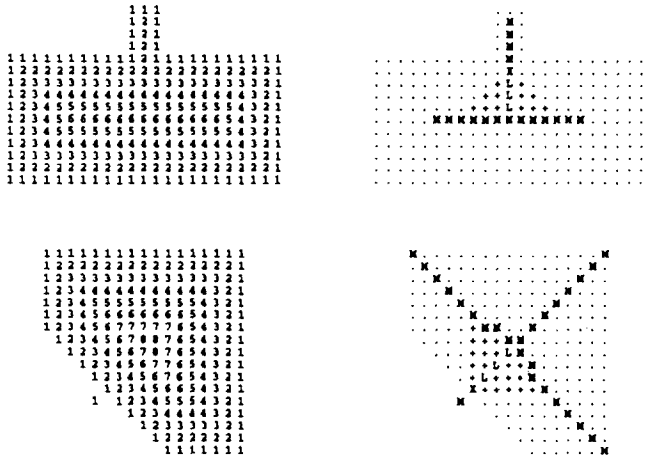

Fig. 3. Top line and bottom line refer to $d_{1,1}$-DM and d_1-DM, respectively. M, X and L identify the *cmd*s, the saddle points which are not centres of maximal disks, and the linking pixels detected when $v_1 = 2$, $v_2 = 3$ and $v_3 = -1$. Pixels denoted by '+' would maximize the gradient if $v_1 = w_1 = 1$, $v_2 = w_2 = 1$, and $v_3 = w_3 = \infty$ (in the $d_{1,1}$-DM) and $v_1 = w_1 = 1$, $v_2 = w_2 = \infty$ and $v_3 = w_3 = \infty$ (in the d_1-DM) were selected. Accepting these pixels as linking pixels leads to fan creation.

The linking pixels found in the running example are denoted by L in Fig. 2b.

2.3.4. Spurious loop filling

A few spurious skeleton loops (i.e. loops which do not surround holes of **F**) may be created by the path growing process. The spurious loops have to be filled in (i.e. the pixels inside the loops have to be marked as skeletal pixels), to obtain a topologically correct skeleton. Filling is accomplished after the *cmd*s, the saddle points and the linking pixels have been marked within a raster scan inspection of the DM. During this scan, the coordinates of all the marked pixels are stored in a list. This will allow us to directly access the marked pixels during the third phase of the skeletonization algorithm.

Spurious loops are created when linking paths, aligned along parallel directions, are so close to each other that pixels in a path are diagonal neighbours of pixels in the other path. In the d_1-DM, $d_{1,1}$-DM and $d_{3,4}$-DM, the only possibility to create spurious holes, one-pixel in size, occurs in presence of pairs of diagonally oriented paths (see Fig. 4a). In $d_{5,7,11}$-DM, directions constituted by runs of two horizontal/vertical pixels are created during path growing whenever the knight neighbour is accepted as a linking pixel. If two paths of this type are very close to each other, one more possibility exists to create spurious loops. In this case, the spurious holes are two-pixel in size (see Fig. 4b).

A pixel p is identified as belonging to a spurious hole if any of the following conditions is satisfied, during a forward raster scan inspection of the DM:

1. p is not a skeletal pixel, while all the horizontal/vertical $n_i(p)$ are skeletal pixels.
2. p and $n_5(p)$ are not skeletal pixels, while $n_1(p)$, $n_3(p)$, $n_4(p)$, $n_6(p)$, $n_7(p)$ and $n_5(n_5)$ are skeletal pixels.
3. p and $n_7(p)$ are not skeletal pixels, while $n_1(p)$, $n_3(p)$, $n_5(p)$, $n_6(p)$, $n_8(p)$ and $n_7(n_7)$ are skeletal pixels.

Loop filling does not create excessive thickening of the set of the skeletal pixels, since only a few sparse spurious holes generally exist. Pixels marked to fill spurious loops are denoted by O in the running example shown in Fig. 2b.

2.4. Phase 3: Skeleton extraction

2.4.1. Reduction to unit width

Reduction of the medial line to unit width is obtained

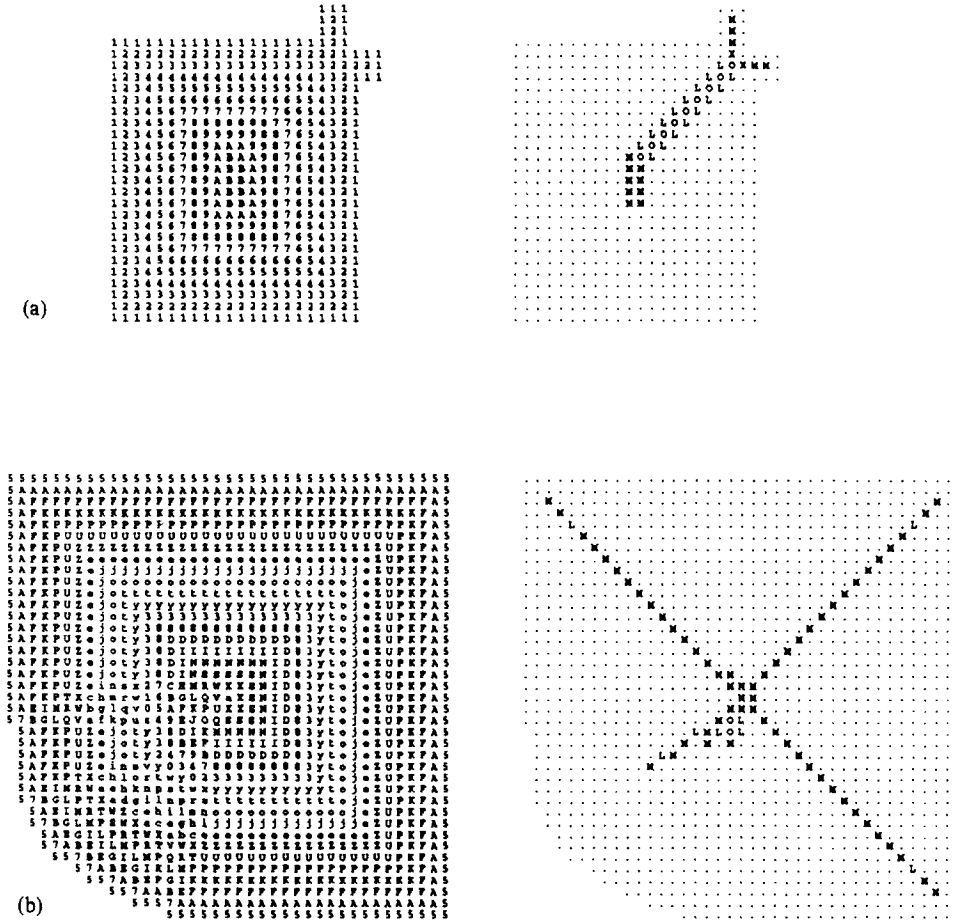

Fig. 4. Letters M, X and L identify the *cmds*, the saddle points which are not centres of maximal disks, and the linking pixels found on the $d_{1,1}$-DM (a) and on the $d_{5,7,11}$-DM (b). Letter O identifies pixels added to the medial line to fill spurious loops.

by applying topology preserving removal operations, designed in such a way to avoid an excessive shortening of the skeleton branches. Reduction to unit width is here referred to as an 'unmarking' process since, rather than removing the superfluous pixels from the medial line, we remove their marker, which was previously ascribed to distinguish them from the remaining pixels of the DM.

The medial line is likely to include 4-internal pixels and a few sparse 8-internal pixels. To favour skeleton centrality within the figure, unmarking is accomplished in two steps. The marked pixels are directly accessed by using the list of coordinates built during loop filling and the pixels which result to be 4-internal in the medial line are flagged (see Fig. 5a, where the 4-internal pixels are labelled 2). Only the non flagged pixels are possibly unmarked during the first unmarking step. All the pixels still marked undergo the unmarking process during the second step. The list of the marked pixels is suitably updated so as to keep track only of the pixels remaining in the skeleton.

A pixel p is unmarked during the first (the second) inspection of the medial line if it satisfies the following Condition (1) (conditions (1) and (2)):

1. At least a triple of neighbours $n_i(p)\, n_{i+2}(p)$, $n_{i+5}(p)$ exists ($i = 1, 3, 5, 7$, addition modulo 8), such that $n_i(p)$ and $n_{i+2}(p)$ are marked, while $n_{i+5}(p)$ is non marked.
2. At least one horizontal/vertical neighbour of p is not marked.

Condition (1) prevents both altering the connectedness and shortening skeleton branches, while Condition (2) prevents the creation of holes in the set of the skeletal pixels. Condition (2) does not need to be checked during the first inspection of the medial line, since only pixels which are not 4-internal in the initial medial line undergo the unmarking process.

Since unmarking is effective on a number of *cmds*, the input figure cannot be completely recovered by applying the reverse distance transformation to the obtained unit wide skeleton. However, since the maximal disks partially overlap each other, loss in recovery is limited to (a few) pixels of the contour of the input figure, and the skeleton can still be considered as adequate to represent **F**. The performance of the unmarking process on the

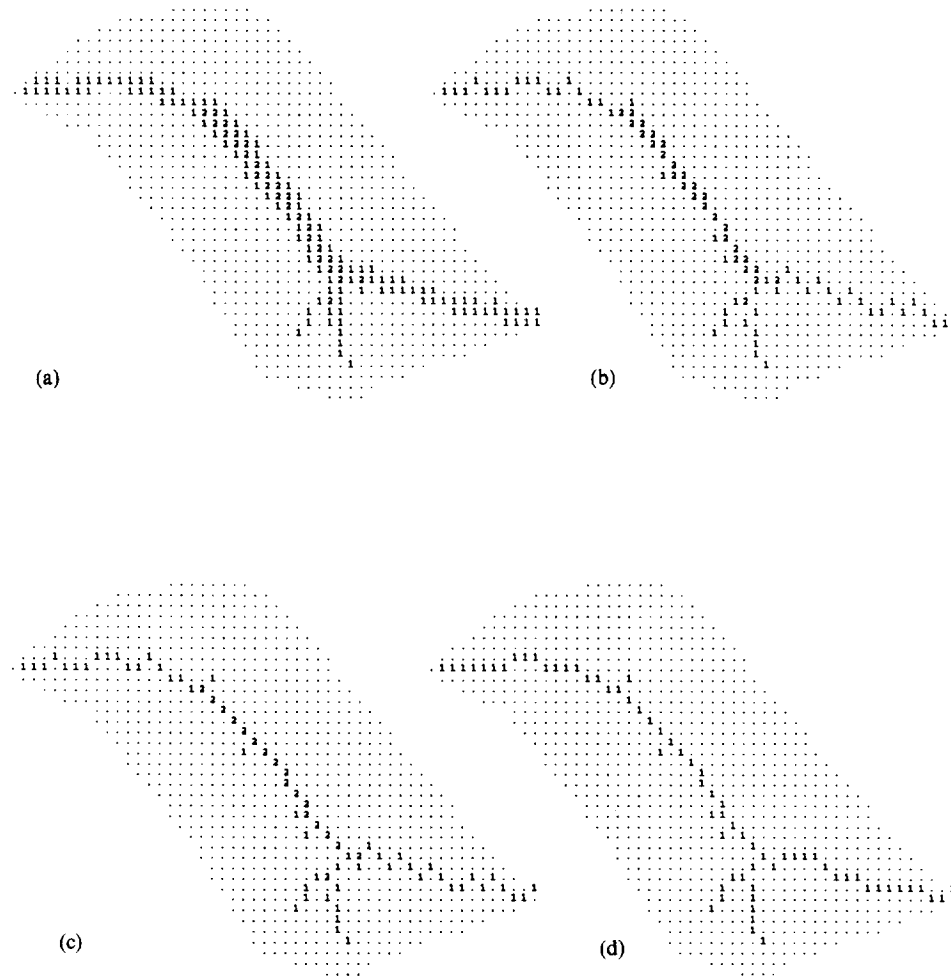

Fig. 5. (a) In the medial line found on $d_{5,7,11}$-DM, labels 1 and 2 (respectively) denote the non 4-internal and the 4-internal pixels. (b) Set resulting after the first, (c) after the second unmarking inspection of the medial line; (d) set resulting after beautifying the unit wide skeleton.

test pattern can be appreciated in Fig. 2c, where the pixels of **F** that cannot be recovered starting from the unit wide skeleton are denoted by U.

2.4.2. Pruning

Pruning is accomplished to simplify skeleton structure by deleting peripheral branches (i.e. branches delimited by end points) that do not correspond to figure protrusions, significant in the problem domain. These branches include, but are not limited to, branches originated from noisy contour protrusions and spurious branches created during the reduction to unit width of the medial line. Pruning is also effective to reduce skeleton sensitivity to figure rotation, translation and scaling.

Pruning a (portion of a) skeleton branch implies deletion of the *cmd*s placed therein. To keep under control the loss of information caused by branch deletion, a suitable criterion based on the relevance of the protrusion associated with the skeleton branch should be used. Generally, a branch can be safely pruned if a negligible difference exists between the two sets, respectively recovered by applying the reverse distance transformation to the skeleton and to the pruned skeleton.

Recently, a pruning method based on a measure of protrusion elongation has been proposed [7]. Let p and q be the end point delimiting a peripheral skeleton branch and a more internal element, along the same skeleton branch. Let d be the distance between the two elements p and q. The quantity $(p - q + d)$ measures the distance between the contour of the two regions, respectively associated with the entire branch, and with the branch pruned up to q (q excluded). This value can be compared with a threshold ϑ, whose value depends upon the accepted tolerance in figure recovery. Pruning can be done up to the most internal element q such that $(p - q + d) \leq \vartheta$. The pixel q can be identified, while tracing the skeleton branch starting from p.

To use the previous pruning criterion, whichever distance function is considered, we express the distance d between two pixels p and q in terms of the number H of horizontal/vertical unit moves, and the number D of diagonal unit moves along a minimal length 8-connected path linking p to q. This can be trivially done for a path-based distance where the permitted moves are through

horizontal/vertical and diagonal neighbours, namely for the chessboard distance ($d_{1,1} = H + D$) and the two-weight distance ($d_{3,4} = 3H + 4D$). For the city block distance only horizontal/vertical moves are permitted and the distance between p and q is the length of a 4-connected minimal path linking p to q. This path differs from a minimal length 8-connected path, since it employs a pair of successive moves (one horizontal move and one vertical move) in place of any diagonal move present in the 8-connected path. Thus it is $d_1 = H + 2D$. Finally, when the three-weight distanced is used, in the minimal length path connecting p and q knight moves are also permitted. On this path, a knight move is taken in place of any pair of successive moves (a horizontal/vertical move and a diagonal move) present on the minimal length 8-connected path connecting p and q and it results in $d_{5,7,11} = 5H + 7D + (11 - 7 - 5) \times \min(H, D)$. Summarizing, the distance can be expressed as follows:

$$d = w_1 H + w_2 D + \text{Flag} \times \min(H, D)$$

where Flag = 0 when using d_1, $d_{1,1}$ and $d_{3,4}$, while Flag = −1 when using $d_{5,7,11}$. Moreover, H and D can be computed in terms of the coordinates of p and q, since it is $H = (2d_{1,1} - d_1)$ and $D = (d_1 - d_{1,1})$.

The pruning threshold ϑ can be set in such a way as to be adequate in the four different distances cases. We fix $\vartheta = k \times w_1$, where k is the maximum number of peripheral rows and/or columns one accepts to lose in figure recovery. The value of $k = 2$ is adequate for the removal of noisy branches.

We do not limit pruning to the branches which are peripheral in the initial skeleton. In fact, an insignificant protrusion can be mapped into a skeleton subset including peripheral and non-peripheral branches. If all the peripheral branches sharing a branch point are pruned, the branch point is transformed into an end point in the modified skeleton. The peripheral branch originating from it can still be representative of an insignificant protrusion and, as such, should be pruned. To avoid a summation effect in the loss of figure recovery, the information on the planar coordinates and label of the starting point(s) of any branch(es) is propagated through the branch(es) while performing pruning. In this way, it is possible to evaluate the relevance of the entire protrusion, mapped in the union of the current peripheral skeleton branch with the neighbouring, already pruned, skeleton branches.

The effect of pruning on the test pattern can be seen in Fig. 2d, where it is $\vartheta = 2 \times w_1$. This should guarantee a loss in recovery of at most two rows and/or columns. Indeed, some more pixels are lost due to the reduction of the medial line to unit width, and to our decision to also remove from the skeleton all the one-pixel short branches possibly remaining after pruning. To point out that the representative power of the skeleton is not biased much by the unmarking process, the sets recovered starting from the skeleton and the pruned skeleton are shown.

2.4.3. *Beautifying*

To have a skeleton whose shape is more appealing, the zigzags due to the presence of noise on the contour of **F** and/or to the process carried out to obtain a unit wide skeleton should be straightened. A beautifying process is accomplished by shifting the marker from suitable skeletal pixels on unmarked neighbours.

A marked pixel p having in the skeleton just two neighbours $n_i(p)$ and $n_{i+2}(p)$ ($i = 2, 4, 6, 8$; addition modulo 8) is unmarked. At the same time, its neighbour $n_{i+1}(p)$ is marked as a skeletal pixel in place of p. If desired, one could keep track of all the pixels that have been unmarked during reduction to unit width, so as to allow shifting the marker from p to $n_{i+1}(p)$, only when $n_{i+1}(p)$ is one of the recorded pixels. This neighbour, in fact, is a skeletal pixel by full right.

Shifting the marker from p to $n_{i+1}(p)$ may cause either of $n_i(p)$ and $n_{i+2}(p)$ to prevent the unit thickness of the skeleton. Thus, if shifting is accomplished, $n_i(p)$ and $n_{i+2}(p)$ are checked and possibly removed.

An elucidating example is shown in Fig. 5d.

2.5. *Algorithm*

The skeletonization algorithm has been implemented on a SUN Sparcstation 2 and checked on a number of 512 × 512 binary pictures, where it runs in a few seconds. About half of the computation time is spent in the DM computation, which requires two raster scans. The remaining two phases of the process are very fast. Medial line extraction is accomplished within one raster scan, interleaved with the path growing process. One more raster scan is then necessary to fill the spurious loops and record in a list the coordinates of all the pixels marked on the DM. The list is used to directly access the skeletal pixels during the third phase of the process, devoted to reduction to unit width, pruning and beautifying.

The algorithm can be summarized as follows:

Step 1. Compute the distance map DM, and build the look-up table (two raster scans)

Step 2. Mark on the DM the skeletal pixels, i.e. the centres of the maximal disks, the saddle points and the linking pixels (one raster scan interleaved with a path growing process)

Step 3. Mark on the DM the pixels inside the spurious loops and record the coordinates of all the marked pixels (one raster scan)

Step 4. Unmark on the DM suitable skeletal pixels, so as to obtain the unit wide skeleton (the skeletal pixels are directly accessed).

Step 5. Unmark on the DM suitable skeletal pixels,

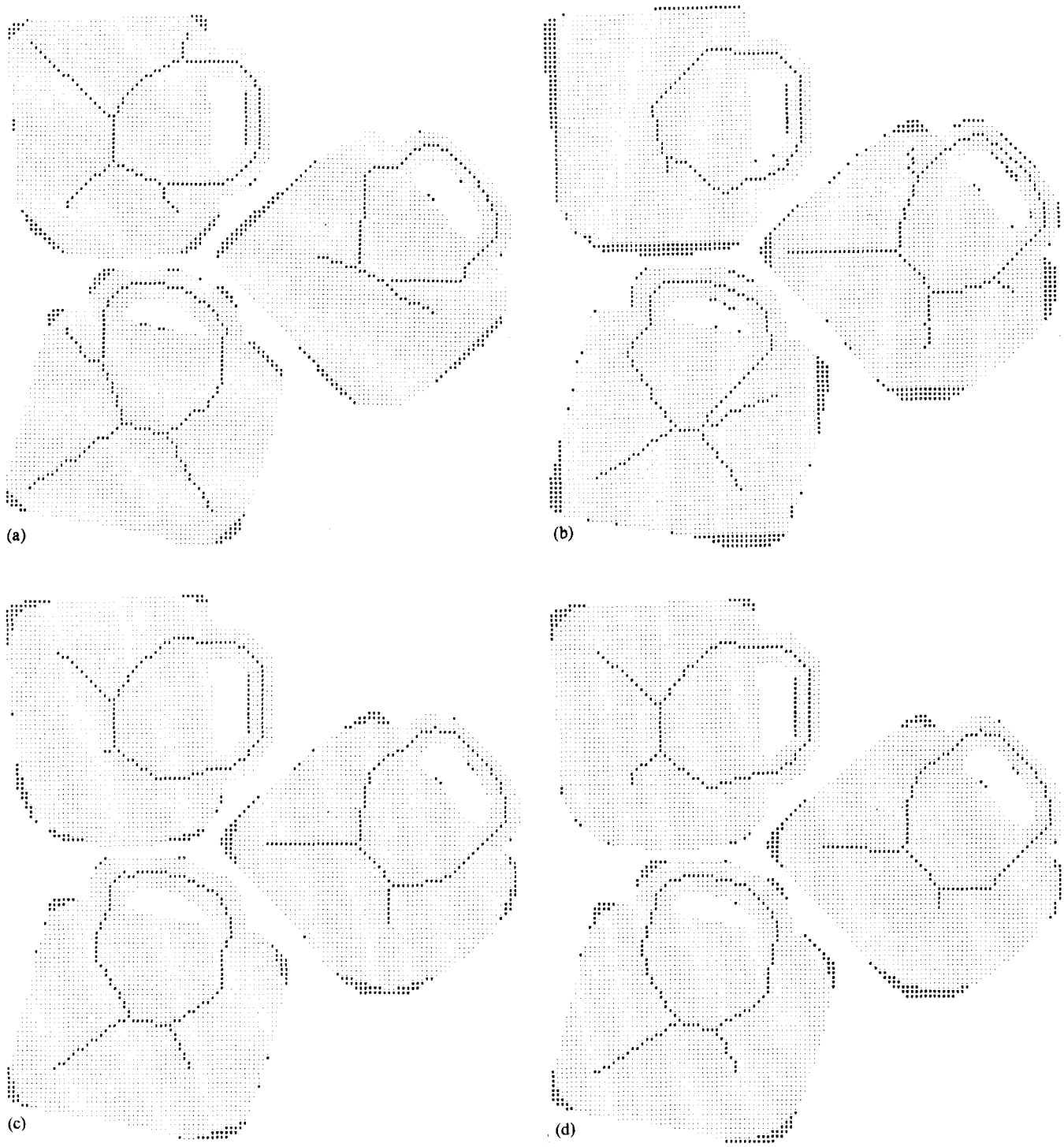

Fig. 6. A mug digitized in three different orientations. The skeletons are respectively extracted from d_1-DM (a) $d_{1,1}$-DM (b), $d_{3,4}$-DM (c) and $d_{5,7,11}$-DM (d). Only the $d_{3,4}$-skeleton and $d_{5,7,11}$-skeleton have an equal number of branches in the three orientations.

so as to perform pruning and beautifying (the skeleton is traced starting from the end points).

3. Concluding remarks

In this paper, we have introduced an algorithm to extract the labelled skeleton of a digital figure from the distance map. The algorithm runs whatever path-based distance function is used among the city-block distance d_1, the chessboard distance $d_{1,1}$, the 2-weight distance $d_{3,4}$, and 3-weight distance $d_{5,7,11}$. The algorithm can be easily extended to treat DM computed by using other

2- and 3-weight distance functions. To this aim, it is enough to build and memorize the subsets of the look-up tables necessary to correctly identify the *cmd*s. Extension of the algorithm to the case of path-based distance functions with n-weights ($n > 3$) is also possible, with some further modification. Investigation in this respect is currently in progress.

The computational cost of the algorithm is modest since only four raster scan inspections are necessary, whatever the thickness of the pattern to be skeletonized. On the contrary, the number of raster scans necessary to implement a skeletonization algorithm based on iterated contour peeling on a sequential computer depends upon pattern thickness, and the cost of skeletonization may become prohibitive for large size images. The computation time required by our algorithm slightly depends upon the ratio between the number of pixels constituting **F** and **B**, and on the adopted distance function. The d_1-skeleton is the less computationally expensive one, but is not invariant under figure rotation. Also, the $d_{1,1}$-skeleton is sensitive to figure rotation; however, its use is advisable due to the facility in deriving from it geometric features of the represented figure [15], provided that the figures to be skeletonized have a fixed orientation. A reasonable compromise between computation time and skeleton stability under pattern rotation, translation or scaling is obtained in the $d_{3,4}$-DM case. The $d_{5,7,11}$-skeleton has to be preferred when a considerable accuracy is required, since its geometrical features very closely reflect pattern features; the slightly higher cost necessary for its computation is repaid by the higher stability degree under isometric transformations of the pattern.

Finally, Fig. 6 illustrates the effect of figure rotation on the skeleton. Only in the $d_{3,4}$-DM and $d_{5,7,11}$-DM cases does the skeleton have the same number of branches in the three different pattern orientations. This was expected, since these distance functions more closely approximate the Euclidean distance.

References

[1] H. Blum and R.N. Nagel, Shape description using weighted symmetric axis features, Patt. Recogn., 10 (1978) 167–180.

[2] G. Sanniti di Baja and E. Thiel, (3,4)-weighted skeleton decomposition for pattern representation and description, Patt. Recogn., 27 (1994) 1039–1049.

[3] C. Arcelli and G. Sanniti di Baja, A width-independent fast thinning algorithm, IEEE Trans PAMI, 7 (1985) 463–474.

[4] L. Dorst, Pseudo-Euclidean skeletons, Proc. 8th Int. Conf. Pattern Recognition, Paris, France (1986) 286–288.

[5] C. Arcelli and G. Sanniti di Baja, A one-pass two-operations process to detect the skeletal pixels on the 4-distance transform, IEEE Trans. PAMI, 11 (1989) 411–414.

[6] C. Arcelli and M. Frucci, Reversible skeletonization by (5,7,11)-erosion, in C. Arcelli et al. (eds), Visual Form Analysis and Recognition, Plenum, New York (1992) 21–28.

[7] G. Sanniti di Baja, Well-shaped, stable and reversible skeletons from the (3,4)-distance transform, J. Visual Comm. Image Representation, 5 (1994) 107–115.

[8] A. Rosenfeld and J.L. Pfaltz, Distance functions on digital pictures, Patt. Recogn., 1 (1968) 33–61.

[9] G. Borgefors, Distance transformations in digital images, Comput. Vision, Graph. Image Process., 34 (1986) 344–371.

[10] E. Thiel, Les distances de chanfrein en analyse d'images: fondaments et applications, PhD Thesis, University J. Fourier—Grenoble I (1994).

[11] C. Arcelli and G. Sanniti di Baja, Weighted distance transforms: a characterization, in V. Cantoni et al. (eds), Image Analysis and Processing II, Plenum, New York (1988) 205–211.

[12] G. Borgefors, Centres of maximal discs in the 5-7-11 distance transform, Proc. 8th Scandinavian Conf. Image Analysis (1993) 105–111.

[13] D. Rutovitz, Pattern recognition, J. Roy. Statist. Soc., 129, Series A (1966) 504–530.

[14] S. Yokoi, J.I. Toriwaki and T. Fukumura, An analysis of topological properties of digitized binary pictures using local features, Comput. Graph. Image Process., 4 (1975) 63–73.

[15] L.P. Cordella and G. Sanniti di Baja, Geometric properties of the union of maximal neighborhoods, IEEE Trans. PAMI, 11 (1989) 214.