Computer Science Department of The University of Auckland
CITR at Tamaki Campus (http://www.citr.auckland.ac.nz/)

CITR-TR-112 July 2002

Skeletons in Digital Image Processing

Gisela Klette[1]

Abstract

Skeletonization is a transformation of a component of a digital image into a subset of the original component. There are different categories of skeletonization methods: one category is based on distance transforms, and a specified subset of the transformed image is a distance skeleton. The original component can be reconstructed from the distance skeleton. Another category is defined by thinning approaches; and the result of skeletonization using thinning algorithms should be a connected set of digital curves or arcs. Motivations for interest in skeletonization algorithms are the need to compute a reduced amount of data or to simplify the shape of an object in order to find features for recognition algorithms and classifications. Additionally the transformation of a component into an image showing essential characteristics can eliminate local noise at the frontier. Thinning algorithms are a very active area of research, with a main focus on connectivity preserving methods allowing parallel implementation. There are hundreds of publications on different aspects of these transformations. This report reviews contributions in this area with respect to properties of algorithms and characterizations of simple points, and informs about a few new results.

1 Centre for Image Technology and Robotics, Tamaki Campus, Building 731,
The University of Auckland, Morrin Road, Glen Innes, Auckland 1005,
New Zealand.

Skeletons in Digital Image Processing

Gisela Klette
CITR Tamaki, University of Auckland
Tamaki Campus, Building 731, Auckland, New Zealand

July 29, 2002

Abstract

Skeletonization is a transformation of a component of a digital image into a subset of the original component. There are different categories of skeletonization methods: one category is based on distance transforms, and a specified subset of the transformed image is a distance skeleton. The original component can be reconstructed from the distance skeleton. Another category is defined by thinning approaches, and the result of skeletonization using thinning algorithms should be a connected set of digital curves or arcs. Motivations for interest in skeletonization algorithms are the need to compute a reduced amount of data or to simplify the shape of an object in order to find features for recognition algorithms and classifications. Additionally the transformation of a component into an image showing essential characteristics can eliminate local noise at the frontier. Thinning algorithms are a very active area of research, with a main focus on connectivity preserving methods allowing parallel implementation. There are hundreds of publications on different aspects of these transformations. This report reviews contributions in this area with respect to properties of algorithms and characterizations of simple points, and informs about a few new results.

Keywords: shape simplification, skeletonization, thinning, distance transform.

1 Introduction

Many operators have been proposed for presenting a connected component in a digital image by a reduced amount of data or simplified shape. In general we have to state that the development, choice and modification of such algorithms in practical applications are domain and task dependent, and there is no "best method". However, it is interesting to note that there are several equivalences between published methods and notions, and characterizing such equivalences or differences should be useful to categorize the broad diversity of published methods for skeletonization. Discussing equivalences is a main intention of this report.

1.1 Categories of Methods

One class of shape reduction operators is based on distance transforms. A *distance skeleton* is a subset of points of a given component such that every point of this subset represents the center of a maximal disc (labeled with the radius of this disc) contained in the given component. As an example in this first class of operators, this report discusses one method for calculating a distance skeleton using the d_4 distance function which is appropriate to digitized pictures.

A second class of operators produces median or center lines of the digital object in a non-iterative way. Normally such operators locate critical points first, and calculate a specified path through the object by connecting these points.

The third class of operators is characterized by iterative thinning. Historically, Listing [10] used already in 1862 the term *linear skeleton* for the result of a continuous deformation of the frontier of a connected subset of a Euclidean space without changing the connectivity of the original set, until only a set of lines and points remains. Many algorithms in image analysis are based on this *general concept of thinning*. The goal is a calculation of characteristic properties of digital objects which are not related to size or quantity. Methods should be independent from the position of a set in the plane or space, grid resolution (for digitizing this set) or the shape complexity of the given set. In the literature the term "thinning" is not used in a unique interpretation besides that it always denotes a connectivity preserving reduction operation applied to digital images, involving iterations of transformations of specified contour points into background points. A subset $Q \subseteq I$ of object points is reduced by a defined set D in one iteration, and the result $Q' = Q \setminus D$ becomes Q for the next iteration.

Topology-preserving skeletonization is a special case of thinning resulting in a connected set of digital arcs or curves. A *digital curve* is a path $p = p_0, p_1, p_2, ..., p_n = q$ such that p_i is a neighbor of p_{i-1}, $1 \leq i \leq n$, and $p = q$. A digital curve is called *simple* if each point p_i has exactly two neighbors in this curve. A *digital arc* is a subset of a digital curve such that $p \neq q$. A point of a digital arc which has exactly one neighbor is called an *end point* of this arc.

Within this third class of operators (thinning algorithms) we may classify with respect to algorithmic strategies: individual pixels are either removed in a sequential order or in parallel. For example, the often cited algorithm by Hilditch [5] is an iterative process of testing and deleting contour pixels sequentially in standard raster scan order. Another sequential algorithm by Pavlidis [12] uses the definition of *multiple points* and proceeds by contour following. Examples of parallel algorithms in this third class are reduction operators which transform contour points into background points. Differences between these parallel algorithms are typically defined by tests implemented to ensure connectedness in a local neighborhood. The notion of a *simple point* is of basic importance for thinning and it will be shown in this report that different definitions of simple points are actually equivalent.

Several publications characterize properties of a set D of points (to be turned from object points to background points) to ensure that connectivity of object and background remain unchanged. The report discusses some of these properties in order to justify parallel thinning algorithms.

1.2 Basics

The used notation follows [17]. A *digital image* I is a function defined on a discrete set $\mathbb{C}$, which is called the carrier of the image. The elements of $\mathbb{C}$ are grid points or grid cells, and the elements $(p, I(p))$ of an image are pixels (2D case) or voxels (3D case). The range of a (scalar) image is $\{0, ...G_{max}\}$ with $G_{max} \geq 1$. The range of a binary image is $\{0, 1\}$. We only use binary images I in this report. Let $\langle I \rangle$ be the set of all pixel locations with value 1, i.e. $\langle I \rangle = I^{-1}(1)$.

The image carrier is defined on an orthogonal grid in 2D or 3D space. There are two options: using the grid cell model a 2D pixel location p is a closed square (2-cell) in the Euclidean plane and a 3D pixel location is a closed cube (3-cell) in the Euclidean space, where edges are of length 1 and parallel to the coordinate axes, and centers have integer coordinates. As a second option, using the grid point model a 2D or 3D pixel location is a grid point.

Two pixel locations p and q in the grid cell model are called *0-adjacent* iff $p \neq q$ and they share at least one vertex (which is a 0-cell). Note that this specifies 8-adjacency in 2D or 26-adjacency in 3D if the grid point model is used. Two pixel locations p and q in the grid cell model are called *1-adjacent* iff $p \neq q$ and they share at least one edge (which is a 1-cell). Note that this specifies 4-adjacency in 2D or 18-adjacency in 3D if the grid point model is used. Finally, two 3D pixel locations p and q in the grid cell model are called *2-adjacent* iff $p \neq q$ and they share at least one face (which is a 2-cell). Note that this specifies 6-adjacency if the grid point model is used. Any of these adjacency relations A_α, $\alpha \in \{0, 1, 2, 4, 6, 18, 26\}$, is irreflexive and symmetric on an image carrier $\mathbb{C}$. The *α-neighborhood* $N_\alpha(p)$ of a pixel location p includes p and its α-adjacent pixel locations.

Coordinates of 2D grid points are denoted by (i, j), with $1 \leq i \leq n$ and $1 \leq j \leq m$; i, j are integers and n, m are the numbers of rows and columns of $\mathbb{C}$. In 3D we use integer coordinates (i, j, k).

Based on neighborhood relations we define connectedness as usual: two points $p, q \in \mathbb{C}$ are *α-connected* with respect to $M \subseteq \mathbb{C}$ and neighborhood relation N_α iff there is a sequence of points $p = p_0, p_1, p_2, ..., p_n = q$ such that p_i is an α-neighbor of p_{i-1}, for $1 \leq i \leq n$, and all points on this sequence are either in M or all in the complement of M. A subset $M \subseteq \mathbb{C}$ of an image carrier is called *α-connected* iff M is not empty and all points in M are pairwise α-connected with respect to set M. An *α-component* of a subset S of $\mathbb{C}$ is a maximal α-connected subset of S. The study of connectivity in digital images has been introduced in [15].

It follows that any set $\langle I \rangle$ consists of a number of α-components. In case of the grid cell model, a component is the union of closed squares (2D case) or closed cubes (3D case). The *boundary* of a 2-cell is the union of its four edges and the boundary of a 3-cell is the union of its six faces.

For practical purposes it is easy to use neighborhood operations (called *local operations*) on a digital image I which define a value at $p \in \mathbb{C}$ in the transformed image based on pixel values in I at $p \in \mathbb{C}$ and its immediate neighbors in $N_\alpha(p)$.

2 Non-iterative Algorithms

Non-iterative algorithms deliver subsets of α-components in specified scan orders without testing connectivity preservation in a number of iterations. In this section we only use the grid point model.

2.1 "Distance Skeleton" Algorithms

Blum [3] suggested a skeleton representation by a set of symmetric points. In a closed subset of the Euclidean plane a point p is called *symmetric* iff at least 2 points exist on the boundary with equal distances to p. For every symmetric point, the associated maximal disc is the largest disc in this set. The set of symmetric points, each labeled with the radius of the associated maximal disc, constitutes the skeleton of the set.

This idea of presenting a component of a digital image as a "distance skeleton" is based on the calculation of a specified distance from each point in a connected subset $M \subseteq \mathbb{C}$ to the complement of the subset. The local maxima of the subset represent a "distance skeleton". In [15] the d_4-distance is specified as follows.

Definition 1 *The distance $d_4(p,q)$ from point p to point q, $p \neq q$, is the smallest positive integer n such that there exists a sequence of distinct grid points $p = p_0, p_1, p_2, ..., p_n = q$ with p_i is a 4-neighbor of p_{i-1}, $1 \leq i \leq n$. If $p = q$ the distance between them is defined to be zero.*

The distance $d_4(p,q)$ has all properties of a metric. Given a binary digital image. We transform this image into a new one which represents at each point $p \in \langle I \rangle$ the d_4-distance to pixels having value zero. The transformation includes two steps. We apply functions f_1 to the image I in standard scan order, producing $I^*(i,j) = f_1(i,j,I(i,j))$, and f_2 in reverse standard scan order, producing $T(i,j) = f_2(i,j,I^*(i,j))$, as follows:

$$f_1(i,j,I(i,j)) = \begin{cases} 0 & \text{if } I(i,j) = 0 \\ min\{I^*(i-1,j)+1, I^*(i,j-1)+1\} & \\ \qquad \text{if } I(i,j) = 1 \text{ and } i \neq 1 \text{ or } j \neq 1 & \\ m+n & \text{otherwise} \end{cases}$$

$$f_2(i,j,I^*(i,j)) = min\{I^*(i,j), T(i+1,j)+1, T(i,j+1)+1\}$$

The resulting image T is the *distance transform image* of I. Note that T is a set $\{[(i,j),T(i,j)] : 1 \leq i \leq n \wedge 1 \leq j \leq m\}$, and let $T^* \subset T$ such that $[(i,j),T(i,j)] \in T^*$ iff none of the four points in $A_4((i,j))$ has a value in T equal to $T(i,j)+1$. For all remaining points (i,j) let $T^*(i,j) = 0$. This image T^* is called *distance skeleton*.

Now we apply functions g_1 to the distance skeleton T^* in standard scan order, producing $T^{**}(i,j) = g_1(i,j,T^*(i,j))$, and g_2 to the result of g_1 in reverse standard scan order, producing $T^{***}(i,j) = g_2(i,j,T^{**}(i,j))$, as follows:

$$g_1(i,j,T^*(i,j)) = \max\{T^*(i,j), T^{**}(i-1,j)-1, T^{**}(i,j-1)-1\}$$
$$g_2(i,j,T^{**}(i,j)) = \max\{T^{**}(i,j), T^{***}(i+1,j)-1, T^{***}(i,j+1)-1\}$$

The result T^{***} is equal to the distance transform image T. Both functions g_1 and g_2 define an operator G, with $G(T^*) = g_2(g_1(T^*)) = T^{***}$, and we have [15]:

Theorem 1 *$G(T^*) = T$, and if T' is any subset of image T (extended to an image by having value 0 in all remaining positions) such that $G(T') = T$, then $T'(i,j) = T^*(i,j)$ at all positions of T^* with non-zero values.*

Informally, the theorem says that the distance transform image is reconstructible from the distance skeleton, and it is the smallest data set needed for such a reconstruction. The used distance d_4 differs from the Euclidean metric. For instance, this d_4-distance skeleton is not invariant under rotation. For an approximation of the Euclidean distance, some authors suggested the use of different weights for grid point neighborhoods [4]. Montanari [11] introduced a quasi-Euclidean distance.

In general, the d_4-distance skeleton is a subset of pixels $(p, T(p))$ of the transformed image, and it is not necessarily connected.

2.2 "Critical Points" Algorithms

The simplest category of these algorithms determines the midpoints of subsets of connected components in standard scan order for each row. Let l be an index for the number of connected components in one row of the original image. We define the following functions for $1 \leq i \leq n$:

$$e_i(l) = \begin{cases} j \text{ if this is the } l\text{th case } I(i,j) = 1 \wedge I(i,j-1) = 0 \\ \text{in row } i\text{, counting from the left, with } I(i,-1) = 0 \end{cases}$$
$$o_i(l) = \begin{cases} j \text{ if this is the } l\text{th case } I(i,j) = 1 \wedge I(i,j+1) = 0 \\ \text{in row } i\text{, counting from the left, with } I(i,m+1) = 0 \end{cases}$$
$$m_i(l) = int((o_i(l) - e_i(l)/2) + o_i(l)$$

The result of scanning row i is a set of coordinates $(i, m_i(l))$ of midpoints of the connected components in row i. The set of midpoints of all rows constitutes a *critical point skeleton* of an image I. This method is computationally efficient.

The results are subsets of pixels of the original objects, and these subsets are not necessarily connected. They can form "noisy branches" when object components are "nearly parallel" to image rows. They may be useful for special applications where the scanning direction is approximately perpendicular to main orientations of object components.

3 Iterative Thinning Algorithms

Thinning is an operation applied to binary images I which preserves connectivity of components of $\langle I \rangle$. It is normally implemented by an iterative process of transforming specified contour points into background points. The result of a thinning algorithm might be defined to be *ideally thin* if no point which is not an end point can be deleted without violating connectivity properties. It can happen that a thinning procedure results in digital arcs which intersect each other, generating *branching points* which have more than two neighbors. An ideally thin subset of $\langle I \rangle$ may contain branching points as in the following example:

```
1       1       1
  1     1     1
    1   1   1
1 1 1 1 1 1 1
    1   1   1
  1     1     1
1       1       1
```

Note that this is not a digital arc or curve as defined above. However, these calculated connected components can be split into a set of disjoint digital arcs. A thinning algorithm should satisfy the following conditions in order to achieve skeletonization within feasible computation time:

1. The resulting subset of the original image should be ideally thin.
2. The resulting subset must approximate the medial axis.
3. End points must be preserved.
4. Connectivity of the object and the background must be preserved.
5. The algorithm should be computationally efficient.
6. The algorithm should be robust against noise.

The location of the resulting skeleton (condition 2) critically depends upon the sequence of applications of conditions or masks. Algorithms in the literature differ in tests of conditions 3 and 4 and types of applied local operators. The choice of an operator (sequential or parallel, with a different number of subcycles) and the choice of connectivity tests have influence on conditions 1 and 2. Connectivity preservation for single points can be precisely defined in different ways and will be discussed separately in this report. Some of the connectivity preserving criterions are equivalent and algorithms differ in ways of implementation and additional tests which are designed to improve the algorithm based on above conditions. These goals are used for comparisons and evaluations of thinning algorithms apart from the fact that they are not exactly defined.

This report presents a few examples of often cited algorithms and lists some properties without extensive comparisons and evaluations.

3.1 Basic Notations

Definition 2 *A simple point p in an* α-connected *component $M \subseteq \mathbb{C}$ of a binary image I with $I(p) = 1$, is a point where a transformation of I into a binary image I' by setting $I'(p) = 0$ does not change the* α-connectivity *of $M \setminus \{p\}$ in the transformed image I'.*

This report uses the following synonyms: *object* as a synonym for an *α-connected* component M, $M \subseteq \mathbb{C}$, *background* as a synonym for an *α-connected* component B of the complement of M, $\mathbb{C} \setminus M$, *deletion* as a synonym for the operation of transforming an object point into a background point.

In a binary image I an object M consists of α-connected pixels $(p, I(p))$ with $I(p) = 1$, and a background B consists of α-connected pixels $(p, I(p))$ with $I(p) = 0$. We say a point is *α-simple* (short: simple) if the deletion of this point preserves α-connectivity. The following definitions of the *crossing number* of point p will be used for testing a point p to be simple. Let $A_8(p)$ be the 8-adjacency set of p and the elements x_i of this adjacency set are numbered in the following way:

$$\begin{array}{ccc} x_4 & x_3 & x_2 \\ x_5 & p & x_1 \\ x_6 & x_7 & x_8 \end{array}$$

Definition 3 *[18] The number of transitions from a background point to an object point or vice versa when the points of $A_8(p)$ are traversed in counter-*

clockwise order is called R-crossing number $X_R(p)$*:*

$$X_R(p) = \sum_{i=1}^{8} |x_{i+1} - x_i| \quad \textit{where } x_9 = x_1 \ .$$

Let $X_A(p)$ be the number of distinct 1-components of object points in $A_8(p)$. Then it is easy to see that $X_A(p) = X_R(p)/2$.

Hilditch [5] defined the crossing number as follows:

Definition 4 *The number of times of crossing over from a background point to an object point when the points in $A_8(p)$ are traversed in order, cutting the corner between 0- adjacent 1-neighbors, is called* H-crossing number $X_H(p)$*:*

$$X_H(p) = \sum_{i=1}^{4} b_i$$

where

$$b_i = \begin{cases} 1 & \textit{if } x_{2i-1} = 0 \ \textit{ and } \ (x_{2i} = 1 \ \textit{ or } \ x_{2i+1} = 1) \\ 0 & \textit{otherwise .} \end{cases}$$

The H-crossing number is equivalent to the number of distinct 0-components of object points in $A_8(p)$.

There are straightforward ways to compute crossing numbers $X_R(p)$ and $X_H(p)$. These definitions are frequently used for the design of thinning algorithms in order to find a decision whether a point can be deleted or not. There are thinning algorithms based on condition $X_R(p) = 2$ and others based on $X_H(p) = 1$. Algorithms with $X_R(p) = 2$ preserve 1-connected subsets of the original components and algorithms with $X_H(p) = 1$ preserve 0-connected subsets of the original components (using the grid cell model).

Non-endpoints of ideally thin curves do not satisfy pixel deletion criterions in the sense, that they are non-simple and a thinning algorithm stops if all remaining points are non-simple. This is different for endpoints. Let $B(p)$ denotes the number of object points in $A_8(p)$. There are the following definitions of an endpoint in the literature.

1. $B(p) = 1$,
2. $B(p) \in \{1, 2\}$,
3. $B(p) = 1$, or $B(p) = 2$ and $X_R(p) = 2$.

Condition 1 is not very strong. Algorithms using this condition can delete "important parts" of the skeleton.

3.2 Examples of Thinning Algorithms for 2D

An example for a sequential algorithm is given in [5]. The following criterions are tested for all points $p \in \mathbb{C}$ and pixels are marked in standard scan order from the top to the bottom, and from the left to the right.

A1: p is object pixel.

A2: p is contour pixel, that means at least one 1-neighbor is background pixel.

A3: p is not isolated or end pixel, that means $B(p) > 1$.

A4: At least one object pixel in $A_8(p)$ is unmarked.

A5: $X_H(p) = 1$.

A6: If x_3 is marked, setting $I(x_3) = 0$ does not change $X_H(p) = 1$.

A7: If x_5 is marked, setting $I(x_5) = 0$ does not change $X_H(p) = 1$.

At the beginning let Q be the set of all object pixels and B the set of all background pixels. After one iteration all marked pixels (set D) are set to background pixels. The result $Q' = Q \setminus D$ becomes Q and $B' = B \cup D$ becomes B for the next iteration and so on until no simple point is left for deletion. For an object pixel condition 5 would be sufficient to preserve connectivity. Conditions 1 and 2 are checked at the beginning to be computationally efficient and all other conditions were designed for stability of the skeleton.

[18] provides an example for a parallel algorithm where a 4×4 neighborhood is used. A pixel p is deleted iff the following conditions are true:

B1: p is object pixel.

B2: p is not isolated or end pixel, that means $B(p) > 1$.

B3: $X_R(p) = 2$.

B4: $x_1 x_3 x_5 = 0$ or $X_R(x_3) \neq 2$.

B5: $x_7 x_1 x_3 = 0$ or $X_R(x_1) \neq 2$.

In some cases this algorithm produces results with excessive erosion. For example a 2×2 component of object points will vanish completely. This

disadvantage is based on the nature of parallel algorithms that a set of simple points can be deleted at the same time in one iteration.

There are different ways to prevent erosion and to preserve connectivity in parallel thinning algorithms. One method is considering larger (larger than 3×3) neighborhoods for pixel deletion tests. The following example is an algorithm which is completely parallel [6]:

> C1: p is marked for deletion if $X_R(p) = 2$ and $3 \leq B(p) \leq 6$.
>
> C2: Retain the marked points p if they satisfy one of the following conditions: a) $x_3 = x_7 = 1$ and x_1 is marked, b) $x_1 = x_5 = 1$ and x_7 is marked, c) x_1, x_7 and x_8 are marked. Delete all the other marked points.

In step C1 potentially deletable points are identified. Step C2 preserves connectivity of certain sets of points. If the algorithm is applied completely in parallel mode we may use the following neighborhood of p:

$$\begin{array}{cccc} x & x & x & x \\ x & p & x & x \\ x & x & x & x \\ x & x & x & x \end{array}$$

Thinning operators using smaller neighborhoods (normally 3×3) are preferred for reasons of computational efficiency. The example above can also be considered as an algorithm with two subiterations. Step C1 could be done fully parallel without any deletions of points. The results of C1 could be used in a fully parallel working step C2 using a smaller neighborhood.

One standard example of using subcycles or subiterations is the four-subiteration algorithm by Rosenfeld [16]. A pixel p is deleted if

> D1: p is object pixel.
>
> D2: p is not isolated or end pixel, that means $B(p) > 1$.
>
> D3: $X_H(p) = 1$.
>
> D4: $x_{2i+1} = 0$ where $i = 1, ..., 4, 1, ...$ at successive iterations.

The first of four subiterations deletes in parallel only contour pixels where $x_3 = 0$, the second subiteration deletes only contour pixels where $x_5 = 0$ and so on. The algorithm terminates when no deletions occur during four successive iterations. Rosenfeld has shown that these four iterations preserve connectivity. It turned out that the end point condition $B(p)$ is not strong enough. Too many pixels are deleted in some cases. Therefore condition D2 should be replaced by: $B(p) = 1$, or $B(p) = 2$ and $X_R(p) = 2$.

Computationally more efficient are parallel algorithms with two or only one subcycle. A typical example for a 2-subcycle algorithm [19] is the following: The first subcycle deletes only the following pixels:

E1: p is object pixel.

E2: $1 < B(p) < 6$.

E3: $X_R(p) = 2$.

E4: $x_1 x_3 x_7 = 0$.

E5: $x_1 x_7 x_5 = 0$.

Equations E4 and E5 are equal to 0 if $x_1 = 0$ or $x_7 = 0$ or $x_3 = 0$ and $x_5 = 0$. Therefore in the first subcycle only 4-simple points on the south and east borders as well as north-west pixels can be deleted. The north and west borders and the south-east pixels can be deleted in the second subcycle. That means the last 2 conditions are replaced by equations $x_3 x_5 x_7 = 0$ and $x_1 x_3 x_5 = 0$. The algorithm terminates when no deletions occur at two successive subiterations.

Another group of thinning operators are the window matching algorithms. For instance in [1] points are deleted in parallel if the following masks match the pixel and its adjacency set.

The next iteration deletes pixels which match to the 90 degree rotations of these two masks and so on.

The following example [12] belongs to the group of sequential algorithms. Contour pixels are traced and labeled. Only multiple pixels are retained. Pixel p is *multiple* if at least one of the following conditions is true:

F1: p is traversed more than once during tracing ($X_H > 1$)

F2: p has no 4-neighbors in the interior.

F3: p has at least one 4-neighbor that belongs to the contour but it is not traced immediately before or after p.

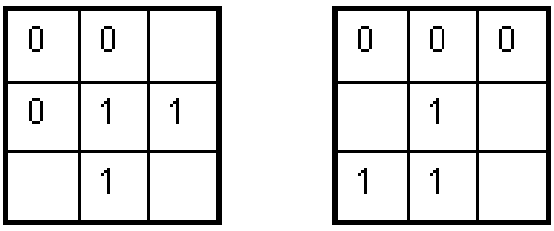


Figure 1: Basic Masks.

In order to preserve connectivity an additional condition is required: In one iteration a pixel p is not deleted if p is multiple and p is adjacent to a labeled pixel of the iteration before.

3.3 Equivalent characterizations of simple points

Let p be a grid square (pixel location) with $I(p) = 1$ of an image I. Then the *I- attachment set of p* is the set consisting of all points on the boundary of p that also lie on the boundaries of pixel locations $q, p \neq q$ with $I(q) = 1$.

In [7] Kong showed the following equivalent characterizations of simple points.

Theorem 2 *Let p be a pixel location with $I(p) = 1$. Then p is simple in I iff both the I-attachment set of p, and the complement of that set in the boundary of p are non-empty and connected.*

For 2D images this characterization can be further simplified [7].

Theorem 3 *Let p be a pixel location with $I(p) = 1$. Then p is simple in I iff the I-attachment set of p is non-empty and connected, and is not the entire boundary of p.*

In two examples of thinning algorithms in this report conditions A5 and D3 are used to preserve connectivity. This characterization of a simple point is equivalent to Kong's' theorem in case vertices are considered as endpoints of edges in the boundary. We show:

Theorem 4 *Let p be a pixel location with $I(p) = 1$. Then p is 0-simple in P iff $X_H(p) = 1$.*

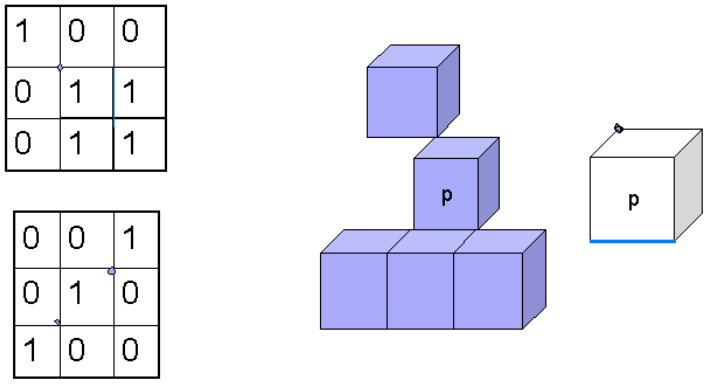


Figure 2: I-attachment sets.

Proof. Let p be 0-simple. Then we have only 4 ways (Figure 3) in which the P-attachment set is non-empty and connected and not the entire boundary, in each case $X_H(p) = 1$.

Let $X_H(p) = 1$. At least one 0-neighbor is a 1 per definition. It follows the I-attachment set is non-empty. At least one 1-neighbor is a 0 per definition, it follows the I-attachment set is not the entire boundary. It remains to show that the I-attachment set is connected. We assume the I-attachment set is not connected. Then we have 7 different ways where the I-attachment set is not connected. In each case $X_H(p) > 1$ per definition. This is a contradiction to above assumption. The I-attachment set must be connected. □

Condition B3 (or E3) is equivalent to Kong's theorem in case edges and vertices are considered as distinct elements of the boundary. We also have:

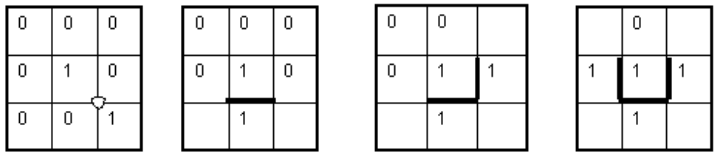
Theorem 5 *Let p be a pixel location with $I(p) = 1$ in a 2D image I. Then p is 1-simple in P iff $X_R(p) = 2$.*

Proof. The proof is analogue to the previous. Let p be 1-simple. Then we have 7 ways in which the I-attachment set is non-empty and connected and not the entire boundary. In each case $X_R(p) = 2$. Let $X_R(p) = 2$. It follows we have only one 1-connected component in set $A(P)$. It follows the I-attachment set is connected. The I- attachment set can't be non-empty or the entire boundary because in these cases $X_R(p) = 0$ what is a contradiction. □

Some authors suggest the use of masks to preserve connectivity in parallel

window matching algorithms. We show the following theorem which also justifies the choice of masks for window matching algorithms:

Theorem 6 *Let p be a pixel location with $I(p) = 1$ in a 2D image I. Then p is 0-simple in I iff the neighborhood of p matches one of the following masks (simple point masks 1 to 4, from left to right)*



or one of their 90° rotations.

Comment: Mask 1 and mask 2 with $I(x_6) = 0$ and $I(x_8) = 0$ are normally not allowed in window matching algorithms in order to avoid the deletion of end points. The first mask in the algorithm matches mask 3 in our theorem. The second mask in the algorithm offers the deletion of 6 different configurations. They all match to one of the masks in the theorem. That means the algorithm deletes only simple points but not all possible simple points are deleted. Let us consider a mask with $I(x_3) = 0$, $I(x_6) = 0$, $I(x_8) = 0$ and all the others are 1's. In this case p is 0-simple but the algorithm will not delete the pixel. The choice of masks in the algorithm avoids the deletion of end points and the creation of "caves".

In the following characterization theorem the term *non-simple* is used for object points p which have the same value $I'(p) = 1$ in the resulting image I' after thinning.

Theorem 7 *Let p be a pixel location with $I(p) = 1$ in a 2D image I and let p be a contour point and let p be not isolated. Then p is 0-non-simple iff $X_H(p) > 1$.*

Proof. Assume p is a contour point, at least one 1-neighbor has value 0, and p is not isolated. It follows per definition that $X_H(p) > 0$, p is 0-non-simple, that means $X_H(p) \neq 1$ and it follows $X_H(p) > 1$. Let $X_H(p) > 1$ then $X_H(p) \neq 1$ and p is non-simple. □

With respect to condition F1 the set of non-simple points is a subset of multiple points.

As we have seen there are several equivalent characterizations of simple points. They all can be used in thinning algorithms and the deletion of a single simple point doesn't change the connectivity of a connected component

of an image. However a parallel algorithm normally deletes a set of points at one time and even if each point of this set is simple the deletion of the whole set of simple points can destroy the connectivity. It is not enough to find characterizations for simple points in order to test connectivity preservation for a parallel algorithm. It is necessary to consider the whole set of simple points which will be deleted by a certain algorithm in one iteration.

3.4 Characterizations of simple sets

Similar to tests of single points to be simple we are interested in testing sets of points which were identified for deletion by a certain algorithm. Ronse defined in [14]:

Definition 5 *Let D be the set of object points in the original image which become background points in the resulting image after thinning, $D \subseteq I$. Let S be the set of object points after thinning, $S = I \setminus D$ and B' is the background $B' = B \cup D$ after thinning. $C_k(I)$ is the number of k-connected components of I, k is equal to 0 or 1. D is k-deletable from P iff $C_k(P) = C_k(S)$, $C'_k(B) = C'_k(B')$.*

D is strongly k-deletable from I iff a) for each k-connected component exists exactly one k-connected skeleton and vice versa b) for each k-connected background component exists exactly one k-connected background component after thinning and vice versa.

In other words a subset D of a digital image I is strongly k-deletable iff two bijective maps exist: one between k-connected components of I and k-connected skeletons and the other one between the k'-connected background components and the k'-connected background components after thinning. Kong defined [7]:

Definition 6 *A sequence $q_1, q_2, ...q_n$ of distinct object points in a digital image I is called a simple sequence of I if q_1 is simple in I, and q_i is simple in $P \setminus \{q_j, 1 \leq j < i\}$ for $1 < i \leq n$. A set D of object points in a digital image is called simple in I if D is empty or if D is finite and the elements of D can be ordered as a simple sequence of I.*

The definitions of simple sets and strongly k-deletable sets are equivalent. We show:

Theorem 8 *A set D of object points $D \subseteq I$ is simple iff D is strongly k-deletable.*

Proof. First we assume that D is strongly k-deletable. Per definition no k-connected component can vanish. For each k-connected component P' exists a proper subset D' with $D' \subset P'$, $S' = P' \setminus D'$, and D is the union of all these subsets D'. I is a digital image, D can only be finite or empty. In case D is empty the k-connected components in the original image are equal to the k- connected components after thinning. In case D is finite all subsets D' are finite or empty. Because for each k-connected component in the original image exists exactly one k- connected component after thinning there must exist exactly one set D' for each k-connected component which is empty or finite. Each non- empty D' has at least one element what must be simple otherwise the deletion would split a k- connected component into two k-connected components. In case D' has more than one element all these elements can be ordered in a simple sequence. All these sequences can be ordered in one sequence one after another and this new sequence is simple. That means D is simple.

Now we assume D is simple. Per definition D must be empty or finite. In case D is empty then the k-connected components of I stay exactly the same and all properties of strongly k-deletable are valid. In case D is finite the elements of D can be ordered as a simple sequence of I. Now we consider all elements of D' of the same k-connected component with $D' \subseteq D$. They constitute a simple sequence. D' can be empty or finite. That means no k- connected component can be split into more than one k-connected component or vanish because per definition after deletion of certain points the last one must be simple. But that means there is at least one more adjacent object point.

Similarly we can show that for each k-connected background component exists exactly one k-connected background component after thinning and vice versa. □

3.5 Connectivity Preserving Algorithms

A thinning algorithm preserves connectivity if it satisfies the following properties:

a: It must not split a k-connected component of I into two or more k-connected components of I.

b: It must not completely delete a k-connected component of I.

c: It must not merge a k'-connected component of the background of I.

d: It must not create a new k'-connected component of the background of I.

Obviously a thinning algorithm preserves connectivity if it deletes a specified set D of an image I which is simple in I. For practical reasons it is more interesting to find test criterions based on a local neighborhood. In order to report about conditions for testing algorithms we introduce two more definitions:

Definition 7 *A set of pixels or voxels is small if every 2 elements of the set are adjacent to each other. An 0-deletable set is a set which can be deleted while preserving 0-connectivity.*

Obviously, every small set of pixels or voxels is connected. A pair of 0-simple object points p, q is 0-deletable iff q is 0-simple after p is deleted. In case p and q are 1-neighbors the pair p, q is 0-deletable iff the number of 0-connected components in the adjacency set of p, q is one. The following examples show configurations where p and q are horizontally adjacent and the number of 0-connected components is larger than one.

0	1	0	0		1	1	0	0		0	1	1	0
0	p	q	0		1	p	q	0		0	p	q	0
0	0	1	0		0	0	1	1		0	1	1	0

Let A be a parallel thinning algorithm. The following conditions are sufficient to show that A preserves connectivity (connectivity preservation test):

a: If an object point is deleted by A then it must be simple.

b: If two 1-neighbors in I are deleted by A then they must constitute an deletable set.

c: No small set of object points is completely deleted by A.

For instance we like to prove that the 4 subcycle algorithm of Rosenfeld preserves connectivity. Condition a is valid obviously because of pixel deletion criteria D3. If a pair of 2 1- neighbors were deleted in the first iteration for instance then $x_{2i+1} = 0$ for p and $x_{2i+1} = 0$ for q because of D4. Together with D2 condition b follows. No small set can completely deleted because of D2 and D3.

4 Summary and Conclusions

One possibility to classify existing skeletonization algorithms could be the use of properties of the resulting set. In [2] six properties of the resulting subset are used to classify existing algorithms. According to the subtitles in this report thinning algorithms are classified into iterative and non-iterative methods. The main category of iterative methods is topology preserving thinning. There are 2 types of operators used in thinning. One type works sequentially in standard scan order or following the contour [5], [12]. The other type of operators works parallel. Parallel working algorithms can be divided into groups. There is the group of fully parallel operators, the group of operators which uses subcycles or neighborhoods larger than 3×3 in order to preserve connectivity and the group of subfield operators which partition the image into subsets and a parallel operator is applied to one of the subfield at each iteration, cycling through the subfields.

This report compiled algorithms of different categories. It shows that different characterizations of simple points (a basic notion for thinning algorithms) are equivalent. For parallel thinning methods it is necessary to characterize simple sets. Connectivity preservation tests are useful in order to prove connectivity preservation of specified algorithms. In this report skeletons based on thinning procedures are special cases where the resulting subset is a set of digital arcs or curves. Topology preserving thinning operators deliver quite often subsets of the original set which are not skeletons in our sense. However there are algorithms in the literature [2] solving this problem.

Further research needs to be done in order to compare and evaluate different methods.

References

[1] C. Arcelli, L. Cordella, S. Levialdi: Parallel thinning of binary pictures. *Electron. Lett.* **11**:148–149, 1975.

[2] C. Arcelli, G. Sanniti di Baja: Skeletons of planar patterns. in: *Topological Algorithms for Digital Image Processing* (T. Y. Kong, A. Rosenfeld, eds.), North-Holland, 99–143, 1996.

[3] H. Blum: A transformation for extracting new descriptors of shape. in: *Models for the Perception of Speech and Visual Form* (W. Wathen-Dunn, ed.), MIT Press, Cambridge, Mass., 362–380, 1967.

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

[5] C. J. Hilditch: Linear skeletons from square cupboards. in: *Machine Intelligence 4* (B. Meltzer, D. Mitchie, eds.), Edinburgh University Press 403–420, 1969.

[6] R.W. Hall: Fast parallel thinning algorithms: parallel speed and connectivity preservation. *Comm. ACM* **32:** 124–131, 1989.

[7] T. Y. Kong: On topology preservation in 2-D and 3-D thinning. in: *International Journal for Pattern recognition and artificial intelligence* vol. 9 No.5 (1995) 813–844.

[8] V. Kovalevsky: Algorithms and data structures for computer topology. in: *Digital and Image Geometry* (G. Bertrand, A. Imiya, R. Klette, eds.), LNCS 2243, Springer, Berlin Heidelberg (2001) 38–58.

[9] L. Lam, S. Lee, C. Suen: Thinning methodologies - a comprehensive survey. *IEEE PAMI* **14** (1992) 869–885.

[10] J. B. Listing: Der Census räumlicher Complexe oder Verallgemeinerungen des Euler'schen Satzes von den Polyëdern. Abhandlungen der Mathematischen Classe der Königlichen Gesellschaft der Wissenschaften zu Göttingen **10** (1862 and 1862) 97–182.

[11] U. Montanari: A method of obtaining skeletons using a quasi-Euclidean distance. *Comm. ACM* **15** (1968) 600–624.

[12] T. Pavlidis: A thinning algorithm for discrete binary images. in: *Computer Graphics Image Processing* **13**(1980) 142–157.

[13] J. L. Pfaltz, A. Rosenfeld: Computer representation of planar regions by their skeletons. *Comm. ACM* **10** (1967) 119–122.

[14] C. Ronse: A topological characterization of thinning. in: *Theoretical Computer Science*, **43**, (1986) 31–41.

[15] A. Rosenfeld, J. L. Pfaltz: Sequential operations in digital picture processing. *Comm. ACM* **13** (1966) 471–494.

[16] A. Rosenfeld: A Characterization of parallel thinning algorithms. *Information and Control* **29** (1975) 286–291.

[17] A. Rosenfeld, R. Klette: Topologies for binary or multi- level images - A review. Keynote, 6th Joint Conf. Inform. Sciences, March 8-13, 2002, Research Triangle Park, North Carolina, 2002 (to appear).

[18] D. Rutovitz: Pattern recognition. *J. Royal Statist. Soc.* **129** (1966) 504–530.

[19] T. Y. Zhang, C. Y. Suen: A fast parallel algorithm for thinning digital patterns. *Comm. ACM* **27** (1984) 236–239.