DISTANCE TRANSFORMS
METRICS, ALGORITHMS AND APPLICATIONS
DISTANCE TRANSFORMS
METRICS, ALGORITHMS AND APPLICATIONS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus, prof. drs. P.A. Schenck,
in het openbaar te verdedigen ten overstaan van een commissie
aangewezen door het College van Dekanen
op 13 juni 1991 te 16.00 uur

door

Bernard Johanna Hendrikus Verwer,
geboren te Leidschendam,
natuurkundig ingenieur.
Dit proefschrift is goedgekeurd door de promotor prof. dr. I.T. Young.

Dr. P.W. Verbeek heeft als toegevoegd promotor in hoge mate bijgedragen aan het totstand komen van dit proefschrift.
Voor Liesa, Falke en Lara
Voor mijn ouders
CONTENTS

1. Introduction ......................................................................................................................... 9
2. An efficient uniform cost algorithm applied to distance transforms .................................... 13
3. Local distances for distance transformations in two and three dimensions .......................... 21
4. A contour processing method for fast binary neighbourhood operations ............................. 35
5. Improved metrics in image processing applied to the Hilditch skeleton .............................. 47
6. 2-D adaptive smoothing by 3-D distance transformation ...................................................... 59
7. Shading from shape, the eikonal equation solved by grey-weighted distance transform ....... 75
8. Heuristic search in robot configuration space using variable metric ................................... 91
9. A multiresolution work space, multiresolution configuration space approach to solve the path planning problem ......................................................... 105
10. Conclusions ....................................................................................................................... 113

References .................................................................................................................................. 117
Summary ..................................................................................................................................... 123
Dankwoord ................................................................................................................................. 125
Curriculum vitae ......................................................................................................................... 127
OVERVIEW OF THIS THESIS

This thesis is organized around a number of papers on distance transforms. Some chapters have an epilogue in which unpublished results are added. The order is chosen as to facilitate reading.

Distance transforms were introduced in 1966 by Rosenfeld & Pfaltz. A distance transform defines for each point in an image the distance to a reference point or a set of reference points. Figure 1a, 1b and 1c show in three different formats the distances inside two gold particles embedded in glass to the glass-gold contact.

Algorithms to calculate distances come in three variants. Parallel, sequential and ordered. Parallel algorithms are only of interest when parallel hardware is available. We will not discuss them in this thesis.

Sequential algorithms are all derived from the original Rosenfeld & Pfaltz algorithm (1966). They serve as reference for a new class of algorithms, the ordered algorithms, which will be discussed in chapter 2 (Verwer et al. 1989).

Distances are defined by metrics. In our daily world, the Euclidean metric holds. In a digital world we often approximate it. Figure 2 shows a few approximations: the well known city-block and chess-board metrics, the Euclidean metric and two "new" metrics from the class of chamfer metrics (Borgefors 1984). Chapter 3 summarizes the literature on optimal approximations for chamfer metrics, derives new results in three dimensions and updates existing results in two dimensions.

Since the distance transform was born in image processing, most of its applications are in image processing. Faster algorithms for the ancient erosion, dilation, propagation and skeletonization are introduced in chapter 4, based on distance transform techniques (Van Vliet & Verwer 1988). Chapter 5 introduced a more accurate (approximating the Euclidean distance better) skeletonization. Figure 3 shows the improvement (Verwer 1988). The epilogue of chapter 5 treats the generalization to three dimensions.

---

Figure 1. Three graphical representations of a distance transform result: on the left hand side in contour lines (points at the same distance to the contour are connected by a line); on top as landscape (the larger the distance, the higher the landscape); on the right hand side as density plot (as the distance increases, the density increases).

---

V Which may include exploiting the parallelism in conventional processors (Van den Boomgaard, 1988, personal communication)
Chapter 6 treats an adaptive way of smoothing 2-D images using a 3-D distance transform. It uses a parabolic metric to achieve scale invariance (Verbeek & Verwer 1989).

A further extension is a nonhomogeneous metric. A new algorithm for the grey weighted distance transform was successfully applied in the design of a 50 guilder coin, see figure 4 (Verbeek & Verwer, 1990). The algorithm is described in chapter 7, its applicability in seismics in the epilogue.

Last but not least, chapter 8 and 9 describe some of the research done in robot path planning. The main problem in path planning is the computational complexity of collision detection. Many robot positions have to be verified during a path planning phase. Chapter 8 shows some of the early work, when we still tried to map the whole configuration space. In chapter 9 the concept of bubble hierarchies is introduced (Featherstone 1989), see figure 5. Distance transforms are again used, both in the design of bubble hierarchies and in the detection of bubble collisions (Verwer 1990).

![Figure 2. 'Circles' for different metrics - from chapter 3.](image)

![City-block skeleton and Hexadecagonal skeleton](image)

![Figure 3. A binary object plus two skeletons. - from chapter 5.](image)

![Figure 4. The Dutch 50 4-queens coin - from chapter 7.](image)
B.J.H. Verwer, P.W. Verbeck and S.T. Dekker

AN EFFICIENT UNIFORM COST ALGORITHM APPLIED TO DISTANCE TRANSFORMS


Any new idea, Mahound is asked two questions. The first is asked when it’s weak: WHAT KIND OF AN IDEA ARE YOU? Are you the kind that compromises, does deals, accommodates itself to society, ramrod-backed type of damnfool notion that would rather break than sway with the breeze? – The kind that will almost certainly, ninety-nine times out of a hundred, be smashed to bits; but, the hundredth time, will change the world.

‘What’s the second question?’ Gibreel asked aloud.
Answer the first one first.

Salmon Rushdie, The Satanic Verses
An Efficient Uniform Cost Algorithm Applied to Distance Transforms

BEN J. H. VERWER, PIET W. VERBEEK, AND SIMON T. DEKKER

Abstract—In artificial intelligence, a number of search algorithms is available for finding shortest paths in graphs. The uniform cost algorithm is a special case of one of those algorithms, the A*-algorithm. In the uniform cost algorithm, nodes are expanded in order of increasing cost. We have developed an efficient version of this algorithm for integer cost values. Nodes are sorted by storing them at predefined places (bucket sort), keeping the overhead low. The algorithm is applied to general distance transformation. A constrained distance transform is an operation which calculates at each pixel of an image the distance to the nearest pixel of a reference set, distance being defined as minimum path length. The uniform cost algorithm, in the constrained case, proves to be the best solution for distance transformation. It is fast, the processing time is independent of the complexity of the image and memory requirements are moderate.

Index Terms—A*-algorithm, bucket sort, distance transform, uniform cost propagation.

I. INTRODUCTION

Heuristic search is widely used in optimization theory, problem solving, games theory, and path finding. A* is a basic algorithm in heuristic search.

A* finds the cheapest path in a graph from a start node to the closest goal node [1]. Nodes are processed in a specific order, determined by an heuristic evaluation function \( f(n) \), defined as an estimate of the cost of the path from the start node to the closest goal node constrained to go through node \( n \). The function \( f(n) \) is the sum of two functions \( g(n) \) and \( h(n) \). \( g(n) \) representing the costs from the start node to node \( n \) and \( h(n) \) an estimate of the costs from node \( n \) to the goal node. In \( A^* \), \( g(n) \) is calculated recursively during the search process; if \( n' \) is a successor node of \( n \), then \( g(n') \) is the sum of \( g(n) \) and the cost \( c(n, n') \) associated with the arc between \( n \) and \( n' \). The estimate \( h(n) \) should be based on the location of the goal and characteristics of the problem domain.

A* can be described in four steps [1].

1) Mark the start node 'open,' calculate \( f(s) \).

2) Select the open node \( n \) whose \( f \) value is smallest. If ties occur, resolve in favor of a goal node.

3) If \( n \) is a goal node, mark \( n \) 'closed' and terminate the algorithm.

4) Otherwise, mark \( n \) 'closed,' calculate \( f \) for each successor node of \( n \) and mark 'open' each successor not already marked 'closed.' Remark as 'open' any closed node \( n \), which is a successor of \( n \) for which \( f(n) \) is smaller now than it was when \( n \) was marked 'closed.' Goto step 2.

A special case of \( A^* \) is the uniform cost algorithm where heuristic information is not used: \( h(n) \) is put to 0. We propose an efficient method to implement this algorithm for integer cost values. Our approach is useful if goal-oriented behavior is not of interest or if the loss of time due to the calculation of \( h(n) \) exceeds the gain introduced by the number of which need not be processed.

II. EFFICIENT UNIFORM COST PROPAGATION

We propose to use a bucket structure to implement the node labeling. If a node is labeled 'open' it is stored in a bucket. Each bucket has associated with it a particular cost value. In bucket \( i \) all nodes with cost value \( g(n) = i \) are stored. A cumbersome sorting procedure to find the node with the lowest cost value is avoided by emptying the buckets in order of increasing cost.

The efficient uniform cost propagation we propose starts by assigning the cost 0 to the start nodes and the value infinite to the other nodes (or a cost larger than the maximum cost to occur). So, the start nodes are stored in bucket 0.

The buckets are then emptied in order of increasing cost, starting with bucket 0. A node retrieved from a bucket checks whether it can propagate costs to adjacent nodes. An adjacent node receives the cost of the generating node plus the cost of the arc between them, if the adjacent node's original cost was higher than the newly calculated cost. In that case, the node is stored in the corresponding bucket to be retrieved at the proper time. After having checked all
adjacent nodes, the generating node is removed from the bucket (labeled "closed"). In [1] it is proven that the costs of all closed nodes are correct if \( h(n) \) has a property called consistency. An equivalent condition is monotonicity [2]. An heuristic function is called monotone if \( h(n) \leq c(n, n') + h(n') \). If no heuristic information is used \((h(n) = 0)\) and if negative arc costs are not allowed \((c(n, n') \geq 0)\) then the heuristic function \( h(n) \) is certainly monotone and then the cost of a generating node will always be correct.

The ordering of nodes inside a bucket is not relevant. Moreover, the buckets can circulate: if a bucket has been emptied, it can be reassigned to the next cost value that could be required. At any time, a limited number of buckets, equal to the maximum arc cost plus 1, is sufficient (nodes in the current bucket can only generate costs between the current cost and the current cost plus the maximum arc cost; buckets for the other cost values are superfluous). The buckets are thus addressed modulo the maximum arc cost plus 1.

The speed of the algorithm is due to the simple administration involved. A sorting of "open" nodes as in the more general A*-algorithm is not required because nodes are stored at a predefined place. This "sorting" method is generally known under the name bucket sort. It is the only "sorting" method with a time complexity of \( O(n) \) where \( n \) is the number of elements to be sorted.

The cost of a generating node is always correct (see above), the cost of a generated node not necessarily. If the cost of a generated node is not correct, it will be corrected before it propagates itself. At the time it is corrected, the node will remain in a wrong bucket. To prevent the node from being processed twice and to circumvent a cumbersome removal procedure, a comparison is included. Upon each retrieval of a node, the value of the node and the index of the bucket from which the node was retrieved, should match. If they do not match, the node has already been processed and is therefore treated as "closed."

III. DISTANCE TRANSFORMS

A. Definitions

Distance transforms are a tool in image processing, with applications in skeletonization, medial axis transformation, convex hull extraction, clustering, matching, and robot path finding. A constrained distance transform is a transform of two binary images \( S \) and \( R \) to a gray-value image \( G \). A gray value in \( G \) represents the minimum distance of an object pixel in \( S \) to the closest pixel in \( R \), the reference image. The background pixels of \( S \) form the constraint image. Examples are given in Figs. 1, 2, and 3. Object pixels are shown white, background pixels black.

The traditional distance transform [3] is a special case of the constrained distance transform, in which the reference image \( R \) is the constraint image of \( S \). The algorithm we present is only faster than existing distance transforms in the constrained case, although in some applications of the normal distance transform its usefulness could also lie in the processing of pixels in order of increasing distance (e.g., skeletonization [4]).

B. Metrics

The distance \( d \) between two pixels is defined as the minimum path length of all the paths, entirely contained in \( S \), between the two pixels. A path is defined as a sequence of basic vectors \( b_i \), a path length as the sum of the distance values \( d_i \) associated with the basic vectors \( b_i \) and \( d_i \) are called chamfer distances [5].

In \( N \) dimensions the elementary basic vectors are \( (x_1, x_2, \ldots, x_N)^T \) with \( x_i \in \{0, \pm 1\} \), \( i = 1, 2, \ldots, N \) (see Fig. 4 for the 2-D elementary basic vectors). Nonelementary basic vectors have components outside the set \( \{0, \pm 1\} \) [6]. In [7] it is proven that the extension to nonelementary basic vectors guarantees that a specified maximum relative error is reached. From hereon we will focus on the elementary basic vectors only, although the algorithm presented is equally applicable for nonelementary basic vectors. In two dimensions the maximum relative error can be minimized to 3.96 percent. In three dimensions to 1.36 percent [7]. For the associated (real valued) chamfer distances, we refer to Table I.

Two remarks:

- There is no limit on the dimensionality of the space. Robot path finding is performed in a six- or seven-dimensional state space [8].
- Other error measures than the maximum relative error are the maximum absolute error [5] and the mean square error [9]. For these error measures a general result as in [7] has not been established yet.

Integer approximations are also shown in Table I, from [10]. In integer approximations only the ratio is optimized, after which a scalar correction is sufficient to obtain absolute distances. The first integer approximation in Table I is the chessboard metric (e.g., Fig. 1), the second approximation the city-block distance (e.g., Fig. 2). As can be seen, these are not optimal metrics with respect to the Euclidean metric. The general 2-D coefficient pairs \( (d_x, d_y) \) will be called an octagonal distance measure because the points at equal distance to one reference pixel form octagonals (e.g., Fig. 3).
TABLE I
CHAMFER DISTANCES OF THE ELEMENTARY BASIC VECTORS IN 2 AND 3
DIMENSIONS, PLUS THE REQUIRED SCALING FACTOR AND THE MAXIMUM
RELATIVE ERROR [10]

<table>
<thead>
<tr>
<th></th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>scaling error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-D best set (real valued)</td>
<td>0.9604</td>
<td>1.3583</td>
<td>1.00</td>
<td>3.96</td>
</tr>
<tr>
<td>2-D integer approximations</td>
<td>1</td>
<td>1</td>
<td>0.85</td>
<td>17.16</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1.21</td>
<td>17.16</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>2.12</td>
<td>5.57</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>7</td>
<td>5.17</td>
<td>4.21</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>20</td>
<td>12.50</td>
<td>4.00</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>41</td>
<td>30.19</td>
<td>3.96</td>
</tr>
<tr>
<td>3-D best set (real valued)</td>
<td>0.9398</td>
<td>1.3291</td>
<td>1.4278</td>
<td>1.00</td>
</tr>
<tr>
<td>3-D integer approximations</td>
<td>1</td>
<td>1</td>
<td>0.79</td>
<td>26.80</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1.27</td>
<td>26.80</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>3.07</td>
<td>7.95</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6</td>
<td>4.29</td>
<td>6.79</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>10</td>
<td>7.40</td>
<td>6.39</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>16</td>
<td>11.71</td>
<td>6.32</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>17</td>
<td>12.80</td>
<td>6.26</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>26</td>
<td>20.24</td>
<td>6.11</td>
</tr>
</tbody>
</table>

3): We propose to use the same name in higher dimension, although the points at equal distance then form other shapes.

C. Algorithms

Borgefors [5, 6] uses a filter approach to calculate at each position in an object the distance to its contour. Two passes suffice. Dorst and Verbeek [11] use the same approach for the constrained case. The image now has to be processed until convergence. Piper and Granum [12] tested several combinations of scanning methods and masks. They propose some improved sequential transforms, but also show that for some applications an ordered propagation algorithm is better suited. They propagate distances from the reference points. The adopted connectivity determines the order in which pixels are processed. Pixels are thus processed in order of increasing chessboard distance, while the distances calculated are octagonal. This technique is called breadth-first in heuristic search. In some cases (see Fig. 5) this leads to major recalculation of distance values. Therefore, Piper and Granum decide to change the order of calculation in such cases. An ad hoc solution. We believe that the uniform cost algorithm more closely corresponds to the concept of propagating distances.

We will not compare our algorithm to parallel algorithms. The parallel distance transforms will have to do as many iterations as the largest distance in the image, while our algorithm needs to process each pixel only once. Any software implementation will therefore be in favor of our algorithms. For special purpose hardware, the same holds. Only if one has general purpose parallel hardware at one's disposal, which does not allow the manipulation of addresses, the balance could shift in favor of the parallel algorithms or in favor of the sequential local transforms. Any comparison in that case is useless.

IV. THE UNIFORM COST ALGORITHM APPLIED TO DISTANCE TRANSFORMS

Applied to distance transforms, the uniform cost algorithm operates in an image. The nodes are the pixels. The arcs are determined by the basic vectors of the metric. Constraint pixels are labeled "write-protected" and do not participate in the process. (If nonelementary basic vectors are used, care has to be taken that basic vectors do not cross constraints. In that case the pixels crossed by the nonelementary basic vector have to be nonconstraint pixels as well.) The algorithm described in Section III can be implemented straightforward. The chamfer distances $d_i$ satisfy the condition that $c(n, n') \geq 0$. After having generated distances a pixel is labelled "closed." Each "closed" pixel has the correct distance value and is processed only once.

In the Sections VI and VII, we will compare the performance of our algorithm to the traditional algorithm. First we will, however, discuss a difference between the approach of the $A^\star$-algorithm and the approach of distance transform algorithms.

V. WRITE VERSUS READ FORMALISM

There is a subtle difference in the approach of our algorithm versus other distance transforms. In Borgefors [5, 6], Dorst and Verbeek [11], and Piper and Granum [12], at each pixel a new value was calculated by

$$p_0 = \min \{ p_i + c_i \} \quad \text{"read formalism"}$$

in which $p_0$ is the value of the central pixel, in which the $p_i$ are the values of the neighboring pixels, including the central pixel, and in which the $c_i$ are the corresponding chamfer distances. This approach clearly shows the influence of image processing. The operation mimics a convolution. Coefficients are added to instead of multiplied with the pixel values and a minimization replaces the summation.

In our algorithm the minimization disappears and individual comparisons takeover

$$p_i = \begin{cases} p_0 + c_i & \text{if } (p_0 + c_i < p_i) \\ p_0 + c_i & \text{"write formalism."} \end{cases}$$

(2)

Of course the computational complexity is the same, the comparisons in formula (2) correspond to the intermediate calculations in evaluating formula (1).

In distance transforms both formalism are allowed (an example of a uniform cost algorithm which uses the read-formalism is given in [4]).

VI. PROCESSING TIMES

The algorithms have been tested on a number of $256 \times 256$ test images. 3-D algorithms are available as well, but the images are more difficult to display here. The basic differences between the algorithms are the same.

The constraint images were respectively an empty image and the images as shown in Fig. 6(a)-(d). The reference image contained one pixel located at position $(1, 1)$. The output of the distance
transforms, identical for all algorithms, is shown in Fig. 7(a)-(d). The distance values are displayed modulo 256 to show the pixels at equal distance more clearly. The metric used was the smallest integer approximation of the octagonal metric, the coefficient pair \( \{2, 3\} \). This is to avoid overflow in the image memory.

The processing times are printed in Table II. The experiments were performed on a 10 MHz 68000 processor running under UNIX. The algorithms were all written in C. As can be concluded from the table the processing times for empty constraint images are comparable. In that case, our algorithm does not perform better (even \( \pm 20 \) percent worse than the sequential local distance transform).

As soon as obstacles appear, the sequential local transform of Dorst and Verbeeck [5] takes much more time, linearly dependent on the required number of passes through the image. As shown in [12] the number of passes is very data-dependent. The performance can only deteriorate from the nonconstrained case.

The processing time of the ordered propagation distance transform [12] can increase or decrease with respect to the nonconstrained case. For the triangle of Fig. 6(a), the processing time jumps up due to the difference between the octagonal metric and the chessboard connectivity (as explained in Section III). In the spiral and the maze, this difference is not relevant and the processing times decrease because there are less distance values to be calculated.

The uniform cost algorithm accelerates if more obstacle pixels are present. The gain is not linearly proportional to the number of pixels which are obstacle pixels (see, e.g., the processing time of the maze) because some obstacle pixels are still addressed before it is noted that they need not be processed. The processing time is therefore dependent on the data, but obstacles can not slow the algorithm down!

Concluding, we believe the uniform cost algorithm is the most elegant solution for distance transforms in the constrained case. Other algorithms can be satisfactory in specific cases, but the uniform cost algorithm has a good performance at all times.

**VII. MEMORY REQUIREMENTS**

An upper limit for the size of one bucket is the size of the image, if the whole image consists of reference points. In practical situations a smaller size will suffice. The order of the number of pixels which have to be stored is, except for in pathological cases [12], one order smaller than the order of the total number of pixels which are present.

If the dynamical allocation scheme is used, the size of the buckets is determined automatically. In a dynamical allocation scheme the buckets are built of chunks, each of which can contain a fixed number of pixels. If a chunk is filled during the calculations, the next free chunk is used or a new one is allocated. In Fig. 8, the allocated memory is shown as a function of the number of points (4 bytes) per chunk. The allocated memory shows in all cases a decaying exponential, followed by a sawtooth and finally by a linear function. The decaying exponential is due to overhead over the linked list structure, being huge for small chunksizes. The sawtooth is due to the quantization of the chunksize, one pointer extra per chunk can make a set of chunks superfluous. And finally, the linear function arises when the minimum number of chunks, equal to the number of buckets, is reached. A larger chunksize will not decrease the number of chunks, but only increase the allocated memory.

Fig. 9 shows the influence of the chunksize on the processing time. The asymptotic behavior quickly stabilizes after \( \pm 10 \). The time involved to switch from one chunk to the next is than negligible. It can be concluded that a reasonable chunksize is not data-dependent. Any number between 15 and 100 is satisfactory.

**VIII. CONCLUSION**

Our algorithm for uniform cost transforms efficiently sorts "open" nodes by putting them at a previously defined place. It can be used as a heuristic search technique in those cases where the use of heuristic information is not possible or time consuming.

Applied as distance transform the algorithm is as fast as other known algorithms. Applied as constrained distance transform the algorithm is generally faster than sequential local transforms. Memory requirements are moderate. We believe our algorithm is a more elegant solution and more easily programmable than the propagated distance transform. Each pixel is processed only once. The simple approach offers perspectives for hardware implementations.

Our algorithm can be used as a substitute for a filtering technique in image processing and as a simple heuristic search technique for artificial intelligence purposes.
REFERENCES


EPILOGUE TO CHAPTER 2

Below I will prove that the chamfer distance is a metric. A metric on a set A is a function $d$ from $A \times A$ to $R$. The function has to satisfy three conditions:

1. $d(\vec{u}, \vec{v}) = 0$, if and only if $\vec{u} = \vec{v}$
2. $d(\vec{u}, \vec{v}) = d(\vec{v}, \vec{u})$;
3. $d(\vec{u}, \vec{v}) \leq d(\vec{u}, \vec{w}) + d(\vec{w}, \vec{v})$.

Given a square orthogonal grid in $N$ dimensional space, a local approach to implement metrics is (chapter 3):

- Define a set $P$ of prime vectors on the grid (in the paper these were called basic vectors, but prime vectors, Ragnemalm, 1990, is probably better)
  
  $P = \{ \vec{p}_1, ..., \vec{p}_l \}$

- Define a path $Q$ as a sequence of prime vectors:
  
  $Q = (\vec{p}_{i_1}, ..., \vec{p}_{i_m})$, $1 \leq i_k \leq l$

- Assign local distances (in the paper called chamfer distances) $d_i$ to the prime vectors $\vec{p}_i$ and define the length $L$ of the path $Q$ as:
  
  $L(Q) = \frac{1}{s} \sum_{k=1}^{m} d_{i_k}$

where $s$ is a constant, real-valued scale factor, useful if we constrain the local distances $d_i$ to be integer valued.

- Let the length of the empty sequence be zero:
  
  $L(\epsilon) = 0$

- Define the chamfer distance $d_e$ between two points $\vec{u}$ and $\vec{v}$ on the grid as the minimum path length of all paths between $\vec{u}$ and $\vec{v}$ (figure 1):
  
  $d_e(\vec{u}, \vec{v}) = \min_Q (L(Q))$

The local distances $d_i$ are generally chosen to minimize the difference between $d_e(\vec{u}, \vec{v})$ and the Euclidean distance $d_e(\vec{u}, \vec{v})$:

$d_e(\vec{u}, \vec{v}) = \sqrt{ \sum_{j=1}^{N} (u_j - v_j)^2 }$

where $u_j$ and $v_j$ are the coordinates of $\vec{u}$ and $\vec{v}$ in the space with dimension $N$.

Let $\vec{p}_i$ be a prime vector connecting point $\vec{a}$ with point $\vec{b}$. Then $\vec{p}_i'$, with local distance $d_i'$, is defined as the prime vector connecting $\vec{b}$ with $\vec{a}$.

\[ \vec{p}_i \]

\[ \vec{p}_i' \]

\[ \vec{u} \]

\[ \vec{v} \]

Figure 1. The distance between points $\vec{u}$ and $\vec{v}$ is defined as the minimum path length between $\vec{u}$ and $\vec{v}$, using a limited number of prime vectors. Note that more than one minimum path can exist between two points.

**Theorem:** If $d_i > 0$ and $d_i' = d_i$, then $d_e(\vec{u}, \vec{v})$ is a metric.

**Proof condition (1).**

If $\vec{u} = \vec{v}$, $d_e(\vec{u}, \vec{v}) = 0$ by definition (7) and (8), if $\vec{u} \neq \vec{v}$, the path from $\vec{u}$ to $\vec{v} \neq \emptyset$ and $d_e(\vec{u}, \vec{v}) > 0$ since $\forall i d_i > 0$.

**Proof condition (2).**

If $Q_1 = (\vec{p}_{i_1}, ..., \vec{p}_{i_m})$, $1 \leq i_k \leq l$ is a minimum path from $\vec{u}$ to $\vec{v}$, then we can construct a path $Q_2$ from $\vec{v}$ to $\vec{u}$: $Q_2 = (\vec{p}_{i_m}, ..., \vec{p}_{i_1})$, $1 \leq i_k \leq l$. If $Q_2$ is a minimum path, then $d_e(\vec{u}, \vec{v}) = d_e(\vec{v}, \vec{u})$.

Suppose $Q_2$ is not a minimum path but $Q_3 = (\vec{p}_{j_1}, ..., \vec{p}_{j_n})$, $1 \leq j_k \leq l$, from $\vec{v}$ to $\vec{u}$ is. Then we can construct a path $Q_4 = (\vec{p}_{i_n}, ..., \vec{p}_{i_1})$, $1 \leq j_k \leq l$, from $\vec{u}$ to $\vec{v}$. $L(Q_4) < L(Q_2) = L(Q_1)$. Contradiction since $Q_1$ was a minimum path.

**Proof condition (3).**

Let $Q_1$ be the minimum path from $\vec{u}$ to $\vec{v}$. If $Q_2 = (\vec{p}_{i_1}, ..., \vec{p}_{i_m})$, $1 \leq i_k \leq l$ is a minimum path from $\vec{u}$ to $\vec{w}$, and $Q_3 = (\vec{p}_{j_1}, ..., \vec{p}_{j_n})$, $1 \leq i_k \leq l$ is a minimum path from $\vec{w}$ to $\vec{v}$, then the concatenation $Q_4 = Q_2 Q_3 = (\vec{p}_{i_1}, ..., \vec{p}_{i_m}, \vec{p}_{j_1}, ..., \vec{p}_{j_n})$, has a length $d_e(\vec{u}, \vec{w}) + d_e(\vec{w}, \vec{v})$.

Suppose $d_e(\vec{u}, \vec{v}) > d_e(\vec{u}, \vec{w}) + d_e(\vec{w}, \vec{v})$, then $L(Q_1) > L(Q_4)$, but $Q_1$ was a minimum path. Contradiction.

So all chamfer distances built by symmetric neighbourhoods form a proper metric. The parabolic distances defined in chapter 6 are therefore not a proper metric, but still useful. On the other hand, the knight's move metric, (Das and Chatterji, 1988), is a proper metric, but, although interesting, probably less applicable in image processing. In the next chapter we will focus on local distances leading to an approximation of the Euclidean metric.
B.J.H. Verwer

LOCAL DISTANCES FOR DISTANCE TRANSFORMATIONS
IN TWO AND THREE DIMENSIONS

Submitted for publication

3

‘What do you mean you’ve never been to Alpha Centauri? For heaven’s sake, mankind, it’s only four light years away you know. I’m sorry, but if you can’t be bothered to take an interest in local affairs that’s your own lookout.’

‘Energize the demolition beams.’

LOCAL DISTANCES FOR DISTANCE TRANSFORMATIONS IN TWO AND THREE DIMENSIONS

Ben J.H. Verwer

Pattern Recognition Group
Faculty of Applied Physics
Delft University of Technology
Lorentzweg 1, 2628 CJ Delft, The Netherlands

Abstract: A unified treatment is presented for determining the distance between two points in a two- or three-dimensional digitized space. Instead of a global Euclidean distance, a distance transformation based upon a sequence of optimal local distances is used. The optimal distance is derived in the context of minimizing the maximum error and the unbiased mean square error. Integer approximations for the local distances are developed for neighborhood sizes of three and five. Minimization is performed over circles and spheres to preserve the symmetries of the neighborhoods. In two dimensions the differences with previously published results are small. The results in three dimensions are new.

Key words: Distance transform, local distances, chamfer distances, maximum error, mean square error, bias, isotropy.

1. Introduction

Since 1984 [1] the chamfer distance \( d_c \) has been used to estimate the Euclidean distance \( d_e \) between points on a square grid in a variety of two and three dimensional applications [2-6]. Parameters have been derived on the basis of theoretical considerations which minimize the maximum error \( |d_e - d_c| \) of estimated distances [7] or which minimize the mean square error under the constraint that the bias is zero [8-10]. Neither the maximum error [7], nor the mean square error [8-10] have been normalized by length, so they are absolute, not relative measures. This is important in view of the set of distances over which the parameters are optimized. Both [7] and [8-10] considered a set of distances between the origin and the points on a line \( x=\)constant. Since the distance to points at 45° is greater than to those at 0°, a certain relative error in the diagonal direction improperly dominates the parameter optimization process compared to the same relative error in another direction. To preserve isotropy, this paper will take the set of distances from the origin to a Euclidean circle in the limit of infinite radius (as in [11]).

Section 2 defines the chamfer metric, section 3 describes the transition from the discrete domain to the continuous domain. Section 4 derives results concerning the maximum error, section 5 concerning the unbiased mean square error.

2. Chamfer metric

Given a square orthogonal grid in \( N \) dimensional space, a local approach to implement metrics is [12-14]:

- Define a set \( \mathcal{P} \) of prime vectors on the grid (not necessarily independent):
  \[ \mathcal{P} = \{ \vec{p}_1, \ldots, \vec{p}_l \} \]  

We call these vectors prime vectors, after [15], since we take care (from a practical point of view) not to include any vector in the set \( \mathcal{P} \) which is an integer multiple of any of the other prime vectors.

- Define a path \( Q \) as a sequence of prime vectors:
  \[ Q = (\vec{p}_{i_1}, \ldots, \vec{p}_{i_m}), 1 \leq i_k \leq l \]  

- Assign local distances \( d_i \) to the prime vectors \( \vec{p}_i \) and define the length \( L \) of the path \( Q \) as:
  \[ L(Q) = \frac{1}{s} \sum_{k=1}^{s} d_{i_k} \]  

23
where \( s \) is a constant, real-valued scale factor that is useful if we constrain the local distances \( d_i \) to be integer valued.

We can rewrite \( L(Q) \) as a sum over \( \mathcal{P} \) instead of over \( Q \):

\[
L(Q) = \frac{1}{s} \sum_{i=1}^{l} n_i d_i
\]

where \( n_i \) is the number of times the prime vector \( \vec{p}_i \) occurs in the path \( Q \).

- Define the gridded chamfer distance \( d_{gc} \) between two points \( \vec{u} \) and \( \vec{v} \) on the grid as the minimum path length of all paths between \( \vec{u} \) and \( \vec{v} \):

\[
d_{gc}(\vec{u}, \vec{v}) = \min_Q \{ L(Q) \}
\]

The aim of this paper is to find the local distances \( d_i \) which minimize the difference between \( d_{gc}(\vec{u}, \vec{v}) \) and the Euclidean distance \( d_e(\vec{u}, \vec{v}) \):

\[
d_e(\vec{u}, \vec{v}) = \sqrt{\sum_{j=1}^{N} (u_j-v_j)^2}
\]

where \( u_j \) and \( v_j \) are the coordinates of \( \vec{u} \) and \( \vec{v} \) in the space with dimension \( N \).

The set \( \mathcal{P} \) at least contains the prime vectors which link a point with its nearest neighbors. If more directions are employed, the resulting metric can better approximate the Euclidean metric. As an example consider that in two dimensions in the 3*3 neighborhood of a point there are eight prime vectors. Four of them are from the central pixel to the north, east, south, and west neighbors, with local distance \( d_{10} \), and the remaining four to the diagonal neighbors with local distance \( d_{11} \). In the 5*5 neighborhood eight "knight's moves" are added, all of length \( d_{21} \).

This is illustrated in figure 1. The other transitions in the 5*5 neighborhood can be written as two cascaded transitions of length \( d_{10} \) or \( d_{11} \) and are therefore not prime vectors.\(^1\)

The traditional city-block metric can be obtained by associating the value 1 with both \( d_{10} \) and \( d_{11} \). The 'circles' in these metrics, that is the sets of points which lie at an equal distance to one point, are, respectively, a diamond and a square. A better approximation to the Euclidean distance can be obtained by letting the ratio of \( d_{10} \) and \( d_{11} \) approximate the real-valued optimal ratio (see sections 4 and 5) 1:\( \sqrt{2} \) (for example by using \( d_{10}=5, d_{11}=7 \)). Such a metric is called octagonal (see figure 2). According to the same argument, a metric obtained by using the 5 by 5 neighborhood is called hexadecagonal. Montanari [14] introduced the general name quasi-Euclidean for these metrics since the error with respect to Euclidean distances can be made arbitrarily small by increasing the neighborhood size [16]. We, however, shall call these metrics chamfer metrics, following Borgefors [1].

The chamfer distance transform propagates distances through a local technique. This is a different approach than the Euclidean distance transform as proposed by

---

\(^1\) \( d_{xy} \) denotes the local distance associated with the vector (x,y) and its symmetric partners, the vectors (x,-y), (-x,y), (-x,-y), (y,x), (y,-x), (-y,x), (-y,-x).
Danielsson [17], which propagates both the x and y distance and takes the square root of the sum of squares at the end. Although for the Euclidean distance transform equation 6 (above) is difficult to implement completely, the method is considerably more accurate than the chamfer distance transform. In two dimensions the absolute error never exceeds 0.29 pixels which in the asymptotic case (as the radius of the Euclidean circle goes to infinity) is negligible. The chamfer distance transform, however, offers more flexibility than the Euclidean distance transform; the constrained case [3] (constrained by obstacles) is a trivial extension for the chamfer distance transform, complicated for the Euclidean distance transform [15]. The grey-weighted distance transform [5] can only be implemented with a chamfer distance transform. In addition, the chamfer distance transform uses less computation time and less memory.

3. From $Z^N$ to $R^N$ and back

In equation 4 the chamfer distance $d_{gc}(\vec{u}, \vec{v})$ is defined as a minimum path length. Because of translation invariance we can rewrite $d_{gc}(\vec{u}, \vec{v})$ as $d_{gc}(0, \vec{v}-\vec{u}) = d_{gc}(0, \vec{a})$ with $\vec{a} = \vec{v}-\vec{u}$. To analyze the behavior of $d_{gc}$ on $Z^N$ we will first derive an analytical counterpart in $R^N$.

Let $B = \{b_i, i=1,...,N\}$ be a set of $N$ linearly independent vectors in $R^N$. Then $B$ is a basis of $R^N$. This means that every vector in $R^N$ can be written as a linear combination of $B$:

$$\vec{a} = \sum_{i=1}^{N} x_i b_i$$

The $x_i$ are the coordinates of the point $\vec{a}$ in the vector space spanned by $B$. In order to reach a continuous equivalent of equation 4, we pick $B$ from the set of prime vectors $P$ (see eq. 1) and constrain the coordinates $x_i$ to be non-negative integers (cf. $n_i \geq 0$). The continuous chamfer distance becomes:

$$d_{cc}(0, \vec{a}) = \frac{1}{s} \sum_{i=1}^{N} x_i d_i$$

The main problem now is to find the set $B$. Or, given a point $\vec{a}$, which terms in equation 4 correspond to the $N$ terms in equation 8?

The solution for $B$ depends, of course, upon the local distances $d_i$. In the next sections we will try to find optimal values for the $d_i$ and this will require the set $B$. We will avoid this “chicken and egg problem” by assuming that the optimal values are close—in ratio—to the Euclidean distances of the prime vectors.

In two dimensions $B$ consists of pairs of adjacent prime vectors. Call the set of points $\vec{a}$ that share a set $B$ a region $R_B$. We say that the vectors of $B$ span $R_B$. For example the prime vectors (1,0), (1,1), and (2,1) and their symmetric partners span 16 different regions (figure 2d). These regions can also be found by complete enumeration. This is illustrated in figure 3 where we first connect all pairs of intersections of the prime vectors with the unit circle. The lines on the outside, closest to the unit circle, then determine the regions. In three dimensions the same procedure can be applied (see section 5.3 and 5.4), each region is spanned by three basic vectors.

![Figure 3. The prime vectors spanning the grey region are the arrowed vectors since the line connecting the intersections of these vectors with the unit circle lies closer to the unit circle than the other lines.](image)

On grid points in the discrete, two-dimensional world $d_{cc}(0, \vec{a}) = d_{gc}(0, \vec{a})$ [14]. Another way to express the same idea is as follows: In two dimensions the grid generated by two prime vectors is identical to the square grid inside the region spanned by those two prime vectors.

---

2The grid generated by $N$ prime vectors in $N$ dimensions is the set of points reached by all integer combinations of the prime vectors.
Unfortunately this is not the case in three dimensions. One example suffices. Take as the set of prime vectors the vectors \((1,0,0), (1,1,0), (1,1,1), (2,1,0), (2,1,1)\) and \((2,2,1)\) plus their symmetric partners. The point \((3,2,1)\) is not part of the grid generated by the prime vectors \((2,1,0), (2,1,1)\) and \((2,2,1)\) although it lies inside the region spanned by these prime vectors.

Thus, given a point on the grid, equation 7 does not necessarily give integer \(x_i\)'s. In a distance transform implementation, other prime vectors may have to be used than the prime vectors which define the region inside which the point lies. This will result in a larger distance than one would expect from equation 8. This difference becomes negligible for large \(x_i\) (that is, \(\Sigma x_i >> 1\)) since the density of the missing grid points is constant as the radius of the Euclidean circle goes to infinity.

There are two other sources of errors which occur in the discrete world:

- The continuous points between which the distance has to be measured do not necessarily coincide with two grid points. For the simple case of figure 4a the actual length is somewhere between \(\sqrt{10}\) and \(\sqrt{34}\). This effect is negligible for large \(x_i\).

- For small \(x_i\) only a limited number of orientations is possible (figure 4b). An optimization over all directions between 0 and \(2\pi\) is therefore only valid for large \(x_i\).

Figure 4. (a) The distance between two points inside the boxes can be any number between \(\sqrt{10}\) and \(\sqrt{34}\). (b) The line connecting two grid points close together can only make a limited number of angles with the horizontal.

It would be tempting to derive expressions for \(d_{ge}(0, \vec{a})\) on regular grids for small \(x_i\), and to derive optimal local distances, say, as a function of the largest \(d_e(0, \vec{a})\) occurring, but I will only pursue the case for large \(x_i\). For large \(x_i\), \(d_{gc}(0, \vec{a})\) is equal to \(d_{cc}(0, \vec{a})\). In the continuous world there is no need to keep the \(x_i\) integer valued, and since we are interested in relative errors only, we can analyze \(d_{cc}(0, \vec{a})\) on the unit circle instead of on a circle in the limit of infinite radius.

4. The maximum error

In this section I will describe a method to find the maximum error for arbitrary neighborhoods in any number of dimensions. The development largely follows [18] but is applicable to any set of prime vectors. The maximum error is sometimes called the maximum absolute error; in both cases it is the absolute value of the maximum difference between the chamfer distance and the Euclidean distance.

4.1. The minimum and maximum distance

To analyze the maximum error we study the minimum and maximum of \(d_{cc}(0, \vec{a})\) for \(|\vec{a}| = 1\), that is, \(\vec{a}\) lies on the unit circle: \(d_e(0, \vec{a}) = 1\).

The minima of \(d_{cc}(0, \vec{a})\) (see equation 8) are found along the prime vectors \((x_i = 1, \vec{p}_k; x_i = 0\) for \(i = 1, \ldots, N, i \neq k\)). This follows directly from the triangular property. The global minimum is then the minimum of all the local minima:

\[
d_{\text{min}} = \frac{1}{s} \min_{i} \left\{ \frac{d_i}{|\vec{p}_i|} \right\}
\]

(9)

The maximum for a set of prime vectors \(\mathcal{B}\) is found using the method of Lagrange multipliers. Equations 10 and 11 have to be solved for \(x_i\). If any of the \(x_i\) in the solution is negative, the solution is not relevant since we are looking for the maximum in \(R_x\), the region determined by all \(x_i\) positive. The maximum is then located on one of the borders of \(R_x\), which form sub-spaces of \(R_x\) spanned by \(N-1\) prime vectors. In these subspaces the same procedure can be applied.

\[
\frac{\partial d_{cc}(0, \vec{a})}{\partial x_i} - \lambda \frac{\partial \Phi}{\partial x_i} = 0, \quad i = 1, \ldots, N
\]

(10)
\[ \Phi = \sum_{i=1}^{N} x_i^2 - 1 = 0 \]  

(11)

\( \Phi \) describes the constraint \( l \hat{d} = 1 \). Equations 10 and 11 have to be solved for all sets \( \mathcal{B} \). The global maximum \( d_{\text{max}} \) is the maximum of all these solutions.

\( d_{\text{min}} \) and \( d_{\text{max}} \) are both proportional to \( 1/s \). We can now, for a given set of \( d_i \), minimize the maximum error by choosing \( s \) such that maximum positive error \( 1-d_{\text{min}} \) (\( d_e > d_c \)) equals the absolute value of the maximum negative error \( 1-d_{\text{max}} \) (\( d_e < d_c \)):

\[ s = \frac{d_{\text{min}} + d_{\text{max}}}{2} \]  

(12)

leading to a maximum error of:

\[ e_{\text{max}} = 1 - d_{\text{min}} = d_{\text{max}} - 1 \]  

(13)

In some applications (for example, collision avoidance [19]) it would be disastrous to overestimate the distance. In that case \( s \) can be set equal to \( d_{\text{min}} \).

4.2. Optimal local distances

The optimal local distances with respect to the maximum error are easy to find: the minimum of the chamfer distance is determined by equation 9. Since any \( d_i \) higher than \( d_{\text{min}} \) can only increase chamfer distances and thus \( d_{\text{max}} \), the set with all \( d_i \) equal to \( |\mathbf{p}_i| \) is a best set:

\[ \forall i: d_i = |\mathbf{p}_i| \]  

(14)

The scale factor, \( s \), still has to be calculated using equation 12.

This approach is independent of dimensionality. For this paper, however, I will restrict myself to two and three dimensions.

In table 1 the maximum error is shown for neighborhood sizes up to 25. In two dimensions we can get a compact expression for the maximum error. We are given two prime vectors \( \mathbf{p}_1 \) and \( \mathbf{p}_2 \), \( \psi \) radians apart, with optimal local distances \( |\mathbf{p}_1| \) and \( |\mathbf{p}_2| \). Solving equations 10 and 11, we find for the maximum chamfer distance:

\[ d_{\text{max}} = \frac{1}{\cos^2 \frac{\psi}{2}} \]  

(15)

In an \( m \times m \) neighborhood the angle \( \psi \) between the vectors (1,0) and \( \left( \frac{1}{2}(m-1), 1 \right) \) is the largest and thus gives the largest \( d_{\text{max}} \). The error, therefore, is solely determined by \( \psi \).

\[ e_{\text{max}} = \frac{1-\cos \frac{\psi}{2}}{1+\cos \frac{\psi}{2}} \]  

with \( \psi = \tan \frac{2}{m-1} \)  

(16)

Previously published optimal values in two dimensions for the \( 3 \times 3 \) neighborhood were \( d_{10} = 0.95509 \) and \( d_{11} = 1.36930 \) [1]. The difference with \( d_{10} = 1/1.0412 = 0.9604 \) and \( d_{11} = \sqrt{2}/1.0412 = 1.3583 \) cf. equation 14 is only 0.56% in value. For the larger neighborhoods no optimal results with respect to the maximum error have been previously published. In three dimensions only suboptimal values have been given with \( d_{100} \) fixed at 1 [1]. In [20] results are given for \( N \) dimensions with neighbourhood sizes \( 3^N \), but all local distances are constrained to 1 or \( \infty \). Furthermore the maximum errors are, as in [1], not normalized by length.

| Table 1. Maximum Absolute Error Versus Neighbourhood Size in Two and Three Dimensions. The Optimal Local Distances are \( d_i = |\mathbf{p}_i| \) with \( s = 1/(1-e_{\text{max}}) \) |
|---|---|---|---|
| size | \( e_{\text{max}} \) 2d | \( e_{\text{max}} \) 3d |
| 3  | 3.9566 | 6.0191 |
| 5  | 1.3557 | 2.4113 |
| 7  | 0.6498 | 1.2234 |
| 9  | 0.3760 | 0.7254 |
| 11 | 0.2439 | 0.4764 |
| 13 | 0.1707 | 0.3357 |
| 15 | 0.1259 | 0.2488 |
| 17 | 0.0967 | 0.1916 |
| 19 | 0.0766 | 0.1520 |
| 21 | 0.0621 | 0.1235 |
| 23 | 0.0514 | 0.1023 |
| 25 | 0.0432 | 0.0861 |

4.3. Integer approximations

Because of speed and/or memory constraints, integer arithmetic can be imperative in image processing. A simple search procedure can determine good integer approximations for the \( 3 \times 3 \) and \( 5 \times 5 \) neighborhoods. Table 2 lists these values for two dimensions and table 3 for three dimensions. Integer approximations in larger neighborhoods are not presented since they are probably less useful although improvement in the form of lower \( e_{\text{max}} \) can still be made. In the \( 7 \times 7 \) neighborhood, for example, the combination \( d_{10} = 12 \), \( d_{11} = 17 \), \( d_{21} = 27 \),
Table 2. Maximum absolute error for integer local distances in two dimensions, 3*3 and 5*5 neighborhoods. The square roots are the optimal non-integer parameters.

<table>
<thead>
<tr>
<th>d10</th>
<th>d11</th>
<th>s</th>
<th>emax %</th>
</tr>
</thead>
</table>
| 0.8536 | 17.16
| 1.2071 | 17.16
| 2.1180 | 5.57
| 5.1675 | 4.21
| 12.5000 | 4.00
| 1.0412 | 3.96
| 4.1213 | 2.94
| 5.0052 | 1.79
| 9.0819 | 1.52
| 17.2174 | 1.43
| 1.0137 | 1.36

Table 3. Maximum absolute error for integer local distances in three dimensions, 3*3*3 and 5*5*5 neighborhoods. The square roots are the optimal non-integer parameters.

<table>
<thead>
<tr>
<th>d100</th>
<th>d110</th>
<th>d111</th>
<th>d210</th>
<th>d211</th>
<th>d221</th>
<th>s</th>
<th>emax %</th>
</tr>
</thead>
</table>
| 0.7887 | 26.79
| 1.3660 | 26.79
| 2.2247 | 10.10
| 3.0725 | 7.94
| 4.2913 | 6.79
| 7.4011 | 6.39
| 19.2355 | 6.11
| 1.0641 | 6.02
| 4.12132 | 2.94
| 9.18913 | 2.66
| 20.5000 | 2.44
| 1.02471 | 2.41

5. The unbiased mean square error

5.1. Two dimensions, the 3*3 neighborhood

For examining the mean square error, equation 8 is once again the point of departure. The eight vectors of the 3*3 neighborhood span eight equivalent regions. Due to symmetry, it suffices to analyze the region spanned by the vectors (1,0) and (1,1), but we need to change to spherical coordinates, \( \theta = (\cos \phi, \sin \phi) \). We are still analyzing \( d_{cc}(0, \theta) \) on the unit circle and in spherical coordinates:

\[
  d_{cc}(0, \theta) = \frac{1}{s} (d_{10}(\cos \phi - \sin \phi) + d_{11} \sin \phi)
\]

As in [8-10] we demand an unbiased estimator of the distance function:

\[
  \beta = \frac{\pi}{4} \int_0^\pi (d_{cc}(0, \theta) - 1) \ p(\phi) \ d\phi = 0
\]

where \( p(\phi) \) is the probability density function of the line \((0, \theta)\). In [21] it was shown that \( p(\phi) \) equals 1. The mean square error, therefore, becomes:

\[
  \sigma^2 = \frac{\pi}{4} \int_0^\pi (d_{cc}(0, \theta) - 1)^2 \ d\phi
\]

(An extra factor of 1/cos\( \phi \) was present in [8-10] because of the extra distance to the line \( x=1 \).) Evaluating equations 18 and 19 yields:

\[
  \beta = (\sqrt{2} - 1)d_{10} + (1 - \frac{1}{2}\sqrt{2})d_{11} - \frac{\pi}{4} = 0
\]

\[
  \sigma^2 = \frac{\pi}{4} - \frac{1}{2}d_{10}^2 + (2 - \sqrt{2})d_{10} + (1 - \frac{\pi}{4})d_{10}d_{11} + (\sqrt{2} - 2)d_{11} + (\frac{\pi}{8} - \frac{1}{4}d_{11}^2 + \frac{\pi}{4}
\]

Equations 20 and 21 are used to solve two problems. First to find the scale factor \( s \) for a given set of integer \( d_i \). Second to determine the optimal real-valued \( d_i \).

For a given set of integer \( d_i \), equation 20 determines the scale factor. Equation 21 gives only the associated mean square error. Based on that mean square error, one can find good integer approximations. These can then be found close – in ratio – to the optimal real-valued \( d_i \).

To find the optimal real-valued \( d_i \) the scale factor is set to one. \( \sigma^2 \) is then minimized subject to the constraint that \( \beta = 0 \):

\[
  \frac{\partial \sigma^2}{\partial d_i} - \lambda \frac{\partial \beta}{\partial d_i} = 0, \quad i = 1, 2
\]
Eliminating $\lambda$ from equation 22 for $i=1$ ($d_{10}$) and $i=2$ ($d_{11}$) gives:

$$(2\pi - 4\sqrt{2}) d_{10} = (\pi\sqrt{2} - 4) d_{11}$$

(23)

Combined with equation 20 this gives as optimal values:

$$d_{10} = \frac{(1+\sqrt{2})\pi}{8} = 0.9481$$

$$d_{11} = \frac{(2+\sqrt{2})\pi}{8} = \sqrt{2} d_{10} = 1.3408$$

(24)

with a mean square error of:

$$\sigma = \sqrt{\frac{(2+\sqrt{2})\pi^2 + \left(\frac{1+\sqrt{2}}{16}\right)\pi - 1} = 2.33\%}$$

(25)

The results in equation 24 were presented earlier in [11] as a length estimator when only the scale factor was optimized.

5.2. Two dimensions: larger neighborhoods

The 3*3 neighborhood is readily generalized to larger neighborhoods. In the appendix a script in Mathematica [22] is listed which goes through the necessary calculations. For the 5*5 neighborhood the optimal values are:

$$d_{10} = 0.9801$$

$$d_{11} = 1.4060$$

$$d_{21} = 2.2044$$

(26)

Table 4 lists the optimal local distances and the associated mean square errors in two dimensions up to the 15*15 neighborhood. In neighborhoods beyond 3*3, the symmetry of the neighborhood is disturbed, the prime vectors are no longer equivalent, and the optimal values are no longer in proportion to the lengths of the prime vectors. In the 5*5 neighborhood, for example, the angle between (1,0) and (2,1) is 27°, between (2,1) and (1,1) only 18°.

Previously published values in the 5*5 neighborhood are: $d_{10} = 0.9797$, $d_{11} = 1.4058$, $d_{21} = 2.2050$, a very small difference with equation 26 (the values given here are actually recalculated from [8-10] to a higher accuracy). The improvement is, of course, only of theoretical importance. Beyond the 5*5 neighborhood, however, the results are new.

Table 5 shows integer approximations for the 3*3 and 5*5 neighborhoods. Also tabulated are the maximum errors which can occur. A reasonable integer approximation in the 7*7 neighborhood is again $d_{10} = 12$, $d_{11} = 17$, $d_{21} = 27$, $d_{31} = 38$, and $d_{32} = d_{11} + d_{21} = 44$ with an

---

3Vectors (1,0) and (1,1) are equivalent since all regions spanned are equivalent. The length of a vector itself is not important, only its length relative to its associated local distance.
r.m.s. error of 0.45%. In the 9*9 neighborhood no approximation can be found with \( d_{10} < 20 \) and an error less than the error of the 3*3 neighborhood.

**TABLE 5. UNBIASED MEAN SQUARE ERROR FOR INTEGER LOCAL DISTANCES IN TWO DIMENSIONS.**

*Also indicated are the minimum and maximum deviations. The last rows are the optimal non-integer parameters.*

<table>
<thead>
<tr>
<th>( d_{10} )</th>
<th>( d_{11} )</th>
<th>( d_{21} )</th>
<th>( s )</th>
<th>( \sigma )</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>chess b.</td>
<td>0.9003</td>
<td>9.77</td>
<td>-21.46</td>
<td>11.07</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>city b.</td>
<td>1.2732</td>
<td>9.77</td>
<td>-21.46</td>
<td>11.07</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
<td>2.1736</td>
<td>2.85</td>
<td>-7.98</td>
<td>2.88</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td></td>
<td>5.2474</td>
<td>2.35</td>
<td>-5.67</td>
<td>2.62</td>
</tr>
<tr>
<td>12</td>
<td>17</td>
<td></td>
<td>12.6684</td>
<td>2.33</td>
<td>-5.28</td>
<td>2.62</td>
</tr>
<tr>
<td>0.9481</td>
<td>1.3408</td>
<td></td>
<td>1.0000</td>
<td>2.33</td>
<td>-5.19</td>
<td>2.62</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>9</td>
<td>4.1203</td>
<td>1.55</td>
<td>-2.92</td>
<td>2.97</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>11</td>
<td>5.0206</td>
<td>1.05</td>
<td>-2.02</td>
<td>1.56</td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>20</td>
<td>9.1409</td>
<td>0.85</td>
<td>-2.15</td>
<td>0.86</td>
</tr>
<tr>
<td>16</td>
<td>23</td>
<td>36</td>
<td>16.3351</td>
<td>0.71</td>
<td>-2.05</td>
<td>0.96</td>
</tr>
<tr>
<td>0.9801</td>
<td>1.4060</td>
<td>2.2044</td>
<td>1.0000</td>
<td>0.71</td>
<td>-1.99</td>
<td>1.00</td>
</tr>
</tbody>
</table>

5.3. Three dimensions: the 3*3*3 neighborhood

In three dimensions we again need spherical coordinates \( \vec{z} = (\cos \varphi \sin \theta, \sin \varphi \sin \theta, \cos \theta) \) and we have 48 equivalent regions (see figure 5). It suffices, therefore, to analyze the region spanned by the prime vectors (1,0,0), (1,1,0) and (1,1,1):

\[
d_{cc}(0, \vec{z}) = \frac{1}{s} (d_{10}(\cos \varphi \sin \theta - \sin \varphi \sin \theta) + d_{11}(\sin \varphi \sin \theta - \cos \theta) + d_{11}(\cos \theta))
\] (27)

The expression for the bias and the mean square error are:

\[
\beta = \frac{12}{\pi} \int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{2}} (d_{cc}(0, \vec{z}) - 1) \rho(\theta, \varphi) \, d\varphi \, d\theta = 0
\] (28)

\[
\sigma^2 = \frac{12}{\pi} \int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{2}} (d_{cc}(0, \vec{z}) - 1)^2 \rho(\theta, \varphi) \, d\varphi \, d\theta
\] (29)

The probability density function \( \rho(\theta, \varphi) \) now equals \( \sin \theta \) [20]. Integration and minimization is performed numerically\(^4\) to find the \( d_i \) which minimize equation 29 constrained by 28.

The optimal local distances resulting from the procedure are:

\[
d_{100} = 0.8940
\]
\[
d_{110} = 1.3409
\]
\[
d_{111} = 1.5879
\] (30)

with an r.m.s. error of 2.33% (this r.m.s. error is remarkably close – but not equal – to the error in two dimensions). For integer approximations to the distances see table 6. These results are new.

The values are not in the proportion of 1 to \( \sqrt{2} \) to \( \sqrt{3} \) since the vectors (1,0,0), (1,1,0) and (1,1,1) are not equivalent: eight regions touch (1,0,0), four regions touch (1,1,0) and six regions touch (1,1,1).\(^5\)

![Figure 5. (a) 'Sphere' in the 3*3*3 chamfer metric.](image)

5.4. Three dimensions: the 5*5*5 neighborhood

In the 5*5*5 neighborhood four domains of integration are necessary. They are indicated in figures 6 and 7. Integrals similar to those in equations 28 and 29 can be written down, which I omit for brevity’s sake. The optimal local distances are then:

---

\(^4\)Mathematica\textsuperscript{TM} [20] built-in functions NIntegrate and FindMinimum were used. The AccuracyGoal option was set to 6 digits, meaning that the minimum \( \sigma^2 \) listed had 6 significant digits.

\(^5\)A set of equivalent prime vectors in three dimensions consists of the vectors to the vertices of an icosahedron: \( b_1 = (1, 0, 0), b_2 = (0.4472, 0.8944, 0), b_3 = (0.4472, 0.2764, 0.8507) \) (this is of no importance in practice). The optimal local distances for these prime vectors are indeed equal: 0.8428.
\[ d_{100} = 0.9556 \\
d_{110} = 1.3956 \\
d_{111} = 1.7257 \\
d_{210} = 2.1830 \\
d_{211} = 2.3885 \\
d_{221} = 2.9540 \]

with an r.m.s. error of 0.782\%. Integer coefficients are also listed in table 6.

Figure 6. ‘Sphere’ in the 5*5*5 chamfer metric. The numbered regions correspond to the regions in figure 7.

Figure 7. $\theta$ as function of $\varphi$ on the borders of the integration domains for the 5*5*5 chamfer metric. The numbered regions correspond to the regions in figure 6.

Conclusions

Optimal local distances in two and three dimensions were calculated with respect to the maximum error and the unbiased mean square error. Table 7 is a quick guide to the tables used in this paper for neighborhood sizes 3 and 5. Furthermore, table 1 lists the maximum error for larger neighborhoods in two and three dimensions. From this table the optimal local distances are simple to deduce. Table 4 lists the unbiased r.m.s. error in larger neigh-

<table>
<thead>
<tr>
<th>$d_{100}$</th>
<th>$d_{110}$</th>
<th>$d_{111}$</th>
<th>$d_{210}$</th>
<th>$d_{211}$</th>
<th>$d_{221}$</th>
<th>$s$</th>
<th>$\sigma$ %</th>
<th>min %</th>
<th>max %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8312</td>
<td>12.04</td>
<td>-30.54</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.5000</td>
<td>10.16</td>
<td>-33.33</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.3312</td>
<td>3.58</td>
<td>-14.21</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.1624</td>
<td>3.28</td>
<td>-10.56</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.4525</td>
<td>2.35</td>
<td>-10.16</td>
</tr>
<tr>
<td>14</td>
<td>21</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15.6877</td>
<td>2.33</td>
<td>-10.76</td>
</tr>
</tbody>
</table>

0.8940 1.3409 1.5879 best set = 1.0000 2.33  -10.60  2.95

| 1         | 2         | 3         | 4         | 5         | 6         | 9     | 1.8083     | 11.48 | -44.70 | 13.33 |
| 3         | 4         | 5         | 6         | 7         | 8         | 9     | 2.9247     | 2.50  | -8.22  | 5.42  |
| 4         | 6         | 7         | 8         | 9         | 10        | 11    | 4.1442     | 1.23  | -3.48  | 2.38  |
| 5         | 7         | 9         | 11        | 12        | 13        | 15    | 5.0753     | 1.16  | -3.47  | 2.38  |
| 9         | 13        | 16        | 20        | 22        | 27        |       | 9.2195     | 0.98  | -2.99  | 2.33  |
| 17        | 25        | 31        | 39        | 43        | 53        |       | 17.9161    | 0.82  | -5.11  | 1.39  |

0.9556 1.3956 1.7257 2.1830 2.3885 2.9540 best set = 1.0000 0.78  -4.44  1.45

31
borhoods in two dimensions and the associated optimal local distances; for three dimensions the amount of data in larger neighborhoods becomes too large.

<table>
<thead>
<tr>
<th>two dimensions</th>
<th>three dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>max. abs error</td>
<td>table 2</td>
</tr>
<tr>
<td>mean sq. error</td>
<td>table 5</td>
</tr>
</tbody>
</table>

In all calculations minimization has been performed over the unit circle in the limit of infinite radius thus preserving isotropy.

As a result of the theory presented here, we are now able to use the local chamfer distance transformation in three dimensions with optimal or close to optimal integer local distances. In two dimensions a small correction has been made to existing optimization methods and results.

Acknowledgements

This research was sponsored in part by the SPIN-FLAIR project DIAC (Delft Intelligent Assembly Cell) and the SPIN-3D project (Three-Dimensional Image Analysis). The author wishes to thank Sito Dekker, Bob Duin, Jim Mullikin, Piet Verbeek, Albert Vossepoel and Ted Young for their contribution to ideas expressed in this paper.

Appendix

Three Mathematica™ scripts are listed in the next column. The first gives the root mean square error in two dimensions for a given set of prime vectors and their associated lengths. The second calculates the optimal values for a set of prime vectors. The third script actually performs the integrations and is called by the first two scripts. Equation 24, for example, is found by issuing:

```mathematica
mseopt[{{1,0},{1,1}}]
```

6 Or numerically (faster):

```mathematica
FindRoot[moment[prime,leng scale,1] == 0, {scale,1}];
```

7 Add //N for quicker results
References


L.J. van Vliet and B.J.H. Verwer

A CONTOUR PROCESSING METHOD FOR FAST BINARY NEIGHBOURHOOD OPERATIONS


4

Lieve pion op a5,

Mooi klein ding, randpion ben je, niet meer dan één veldje mag je bestrijken. Je bent zo klein, bijna niets en je hebt de hele partij daar op je plaatsje gestaan, maar al die tijd was mijn hoop op jou gebouwd en al mijn angstig hunkeren was voor jou. Ik zag je wel, zoals je daar stond, kleine bengel. De mensen dachten natuurlijk dat het om de pion op d5 ging, hij trok hun aandacht, ja ze keken allemaal naar hem, maar jij en ik wisten het wel, het ging om jou, om jou en jou alleen.

Je hebt gewacht stouterd, je hebt je niet opgedrongen, want je wist dat ik al die tijd alleen maar aan jou dacht en dat je niets hoefde te doen, want dat ik vanzelf wel bij je zou komen. Kleine randpion, je ben nu vrij. Ga je gang, op a8 wacht jou en mij de onuitsprekelijke heerlijkheid. Heb mijn dank, lief klein ding. Ik heb je lief,

je Koning
Zwart geeft het op.

Jan Hein Donner, De Koning
Sweet pawn at a5,

Beautiful little thing, contour pawn that you are, not more than one field you may control. You are so small, almost nothing, and you have been at your place the whole game, but all the time my hope was built on you and all my fearfull achig was for you. I saw you, as you stood there, little beggar. They thought of course that it was about the pawn at d5, he attracted their attention, yes, they all looked at him, but you and I knew, it was about you, about you and you alone.

You waited, naughty child, you have not pressed yourself forward, because you knew that I was thinking only about you all that time and that you did not need to do anything, because I would come to you automatically. Small contour pawn, you are free now. Go ahead, at a8 the unpronouncable manor awaits you and me. Be thanked, sweet small thing. I love you, your King,
Black resigns.

Jan Hein Donner, De Koning
A contour processing method for fast binary neighbourhood operations

Lucas J. van VLIEET and Ben J.H. VERWER

Pattern Recognition group of the Faculty of Applied Physics, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands

Received 19 June 1987

Abstract: In this paper new fast algorithms for erosion dilation, propagation and skeletonization are presented. The key principle of the algorithms is to process object contours. A queue is implemented to store the contours in each iteration for the next iteration. The contours can be passed from one operation to another as well. Contour filling and object labelling become available by minor modifications of the basic operations. The time complexity of the algorithms is linear with the number of contour elements to be processed. The algorithms prove to be faster than any other known algorithm.

Key words: Binary neighbourhood operations, skeleton, contour processing algorithm.

1. Introduction

Binary neighbourhood operations are basic operations in image processing. Their simple description has long allowed special purpose hardware to be constructed. In those situations where such hardware is not available, one has to restrict oneself to algorithms suited for general purpose computers. On general purpose computers a straightforward implementation of the binary neighbourhood operations performs poorly, performance being measured as the total required processing time to transform an input image into an output image.

Erosion, dilation, propagation and skeletonization are often iterated several times or even as many times as required to achieve stabilization. We propose a method in which only the interesting pixels are processed. This is achieved by keeping a simple administration in each iteration to aid the next iteration.

We know of three other cases in the literature which deal with fast binary neighbourhood operations. We will compare the performance of our algorithms with the results of Young et al. (1981), and Groen and Foster (1984). Groen and Foster (1984) used a table-look up approach in combination with a region of interest per line. They sorted pointers in order to process connected runs of pixels. Young et al. (1981) used a run-length representation of images to achieve among others binary neighbourhood operations (erosion, dilation, propagation, medial axis transform). However, their medial axis transform is so bounded by the run-length representation that a comparison with the traditional Hilditch-skeleton (1969) is uncalled for.

We were not able to compare our results directly with the results of Piper (1985), who proposed a skeletonization method for an environment based on interval coding, but we will try to indicate how his results relate to ours.

For completeness we will give the processing times achieved by straightforward implementations of the operations as well.
2. Description of operations

We will briefly recapitulate the definitions of erosion, dilation, propagation and skeletonization. We will follow the notation of Groen and Foster (1984).

The operations treated in this paper are based on 3 × 3-neighbourhoods. Erosion, dilation and propagation work with neighbourhoods (Figure 1) in the image before a particular iteration started. We will refer to these neighbourhoods as non-recursive (Figure 2a). Skeletonization as proposed by Hilditch (1969) uses non-recursive, recursive and partially recursive neighbourhoods. Recursive neighbourhoods are neighbourhoods in the image in which the values calculated are updated immediately; partially recursive neighbourhoods consist of seven neighbours taken from a non-recursive neighbourhood and one neighbour taken from a recursive neighbourhood (Figures 2c,d,e,f).

For a 3 × 3-neighbourhood Groen and Foster define the number of 4-connected neighbours $\Phi_4$:

$$\Phi_4 = \sum_{k=0,2,4,6} b_k$$

the number of 8-connected neighbours $\Phi_8$:

$$\Phi_8 = \sum_{k=0,\ldots,7} b_k$$

and the Hilditch crossing number $\gamma_h$ (1969):

$$\gamma_h = \sum_{k=1,2,3,4} h_k$$

where $b_k$ denotes the binary pixel value at position $k$ (Figure 1) in the neighbourhood (Figure 2) and where $h_k = 1$ if $(b_{2k-2} = 0)$ and $(b_{2k-1} = 1$ or $b_{2k} = 1$) else $h_k = 0$.

We assign object pixels the value one and background pixels the value zero. Below we have listed the conditions, which, if true, cause a change in the value of the central pixel $b_8$.

**Erosion**

In each iteration those object pixels are removed, which were connected to at least one background pixel in the preceding iteration. The erosion therefore works with non-recursive neighbourhoods.

4-connected: $b_8 \rightarrow 0$ if $(b_8 = 1$ and $\Phi_4 < 4)$

8-connected: $b_8 \rightarrow 0$ if $(b_8 = 1$ and $\Phi_8 < 8)$

**Dilation**

In each iteration those pixels are added, which were connected to at least one object pixel in the preceding iteration. Again, only non-recursive neighbourhoods are used.

4-connected: $b_8 \rightarrow 1$ if $(b_8 = 0$ and $\Phi_4 > 0)$

8-connected: $b_8 \rightarrow 1$ if $(b_8 = 0$ and $\Phi_8 > 0)$

**Propagation**

Each iteration propagation consists of one iteration dilation of the input image (often called seed) followed by a logical AND with a mask image.

4-connected:

- $b_8 \rightarrow 1$ if $(b_8 = 0$ and $m_8 = 1$ and $\Phi_4 > 0)$
- $b_8 \rightarrow 0$ if $(b_8 = 1$ and $m_8 = 0)$

8-connected:

- $b_8 \rightarrow 1$ if $(b_8 = 0$ and $m_8 = 1$ and $\Phi_8 > 0)$
- $b_8 \rightarrow 0$ if $(b_8 = 1$ and $m_8 = 0)$

$m_8$ denotes the value of the pixel in the mask image which corresponds to the central pixel in the input image.

**Hilditch skeletonization**

Hilditch has developed a skeletonization method...
in which non-recursive, recursive and partially recursive neighbourhoods are used. Non-recursive
neighbourhoods are used to avoid removal of end and break pixels; recursive neighbourhoods to
avoid removal of single pixels and two pixel thick lines; partially recursive neighbourhood to avoid
erosion of two pixel thick lines from the end. Which partial recursive neighbourhoods have to be tested
depends on the scan direction (see Section 4).

Hilditch defined a crossing number ($\gamma_b$) to repre-
sent the topology. If and only if the crossing
number equals one, a pixel may be removed. If the
crossing number exceeds one, a pixel is a link or
vertex pixel. If the crossing number equals zero, the
pixel is either a single pixel or lies in the middle of
an object.

\[ b_8 \rightarrow 0 \quad \text{if} \quad (b_8 = 1 \text{ and } \phi_8 \neq 1 \text{ and } \gamma_b = 1). \]

**Anchor skeletonization**

The anchor skeleton (Verbeek and Duin, 1979) is
a variant of the skeleton in which some pixels are
per definition skeleton pixels. These pixels are spec-
ified in a separate image, called anchor image. The
anchor skeleton is useful to connect different parts
of a scene. E.g. when skeletonizing a greyvalue image
greylevel after greylevel, the skeleton pixels in
an upper level are 'anchored' in a lower level. Or
when skeletonizing layers of printed circuit boards,
connection points are anchored.

If $a_9$ denotes the value of the pixel in the anchor
image, which corresponds to the central pixel in the
input image, then the condition for change becom-
es:

\[ b_8 \rightarrow 0 \quad \text{if} \quad (b_8 = 1 \text{ and } \phi_8 \neq 1 \text{ and } a_9 = 0 \text{ and } \gamma_b = 1). \]

Note that the anchor skeleton reduces to the ordi-
nary skeleton if the anchor image is left empty.
Therefore we have only implemented anchor skele-
tonization.

3. **Principles of approach**

Traditionally, binary neighbourhood operations
have been implemented as table lookups. Each
pixel is treated alike. We propose to take advantage
of specific properties of operations to gain pro-
cessing time. To process as less pixels as possible and
to keep the calculations as simple as possible are
the principles of our approach.

The operations described in the introductory re-
marks have in common the existence of a region of
interest which consist of pixels on or connected to
object contours. We propose to use a queue to ad-
dress these pixels. In the initializing phase of the
operations pointers to contour pixels are queued.
During the iterations pixels which pop from the end
of the queue are processed and the neighbours to be
processed in the next iteration are pushed to the
front of the queue.

A queue is most easily implemented as an array
with two pointers to denote the 'front' and the 'end'
of the queue. The pointers are incremented modulo
the size of the array each time a pointer is read at
the 'front' or written at the 'end'; shifting inside the
queue is not necessary.

Robustness however demands a queue-length of
at least 8/9th of the size of the image to be processed
to be able to store the maximum number of contour
pixels which can occur in an image (see Figure 3).
Taking into account the fact that each pixel only
needs to be queued once for the operations in-
volved, moderate extra memory costs will simplify
the administration to a large extent: if one increases
the queue size to the size of the image to be pro-
cessed the modulo requirement vanishes. We have
adopted this approach for the algorithms presented
in the next section.

If memory allocation should pose problems it is
possible to stick to the 'modulo' approach. The
queue size then has to be determined heuristically.
Apart from pathological cases a queue size of about
20% of the size of the image is sufficient.

The second principle, to keep the calculations as
simple as possible, is a derivative of the first. At the
point where only contour pixels are processed the

Figure 3. Pathological image pattern in which 8/9th of the total
number of pixels belongs to the contour.
need for calculations can vanish. For example, if a contour pixel is removed in an erosion, all object neighbours are subject to removal in the next iteration and any calculation in the next iteration would be redundant. Only the skeleton requires additional computations, induced by the need to hold break and end pixels.

From hereon we will speak of 'queueing a pixel' rather than the more precise 'queueing a pointer to a pixel'.

4. Algorithms

Introduction

The algorithms start with a call to a routine in which all contour pixels are queued. The skeleton uses a special routine because candidate pixels for removal may not be set in the anchor image. After the queue initialization, the operations proceed along different but comparable paths. Either the neighbours of the queue pixels are processed and queued or the pixels on the queue are processed themselves and their neighbours are queued. Most often the first method is used to prevent pixels from being queued twice without having to use labels. In the case of skeletonization the second method is more natural because the recursive and the non-recursive image must be available.

Connectivity

In binary neighbourhood operations connectivity is crucial. Pecht (1985) has shown that the shape of structuring elements can induce redundancy after one or more iterations. For the 8-connected erosion and dilation only diagonal neighbours have to be considered after the first iteration. In the propagation redundancy is not present because of the constraints introduced by the mask image. Skeletonization erodes objects 4-connected and in the 4-connected case naturally only the horizontal and vertical neighbours have to be processed.

Queue initialization

The queue is initialized by filling it with the contour pixels of the objects in the image. A contour pixel is an object pixel which is connected to at least one background pixel.

We have implemented and compared two methods. It proved to be faster to check only object pixels in the detection of the contour if the 8-connected contour is desired or if less than fifty percent of the image is object. Otherwise it is faster to check all pixels with a scan method as proposed by Groen and Foster (1984).

The processing time of the scan method is image independent, while the processing time of the object-oriented method is image dependent. The scan method is implemented by a table-lookup. The table-entry is composed of the 3 × 3 neighbourhood. Three neighbours in the scan-direction are read for each pixel. In the entry of the previous pixel the upper three bits are stripped off, the result is shifted up three bits and the new values are added. In the detection of the 8-connected contour use of the scan-code is not advantageous.

Erosion

Our erosion starts with a call to the initialization procedure. In the first iteration all queued pixels are removed (i.e. the value of the pixels is changed from one to zero). In the next iterations the object neighbours of the pixels on the queue are removed and queued. Because of the sequential procedure each pixel will only be queued once. During the iterations queued pixels are counted in order to know how many pixels must be popped in the next iterations (the number of queued pixels is implicitly derived at the end of each iteration by taking the difference of the pointers to the 'front' and the 'end' of the queue.)

For the 4-connected erosion the horizontal and vertical neighbours are taken into account and for the 8-connected erosion the diagonal neighbours (see remarks in the section on connectivity). To prevent propagation outside the image, we have chosen to remove the edge of the image beforehand. If the edge is removed after calling the initialization routine, objects will not be eroded from the edges of the image; if it is called before, the objects are eroded from the edges as well.

A pseudo code of the algorithm can be found in the Appendix.
Dilation

The dilation also starts with a call to the initializing procedure. Contour pixels which are 4-connected to the background are queued, independent of the type of connectivity of the operation because all background pixels can be reached from these pixels.

In the first iteration all background pixels, 4- or 8-connected to the queued pixels dependent on the connectivity of the operation, are added (the value is changed from zero to one) and queued. Again each pixel will only be queued once.

The next iterations are identical to the first apart from the fact that the 8-connected dilation only accesses the diagonal neighbours as described previously.

A pseudo code of the dilation can be found in the Appendix.

Propagaton

The propagation is the same as the dilation with one extra check. In the initialization seed contour pixels are queued after which the seed image is initialized with the logical **AND** between the seed and the mask image. In the iterations added neighbours have to be set in the mask image.

The propagation is often used to remove objects connected to the edges of an image. This is easily achieved in our algorithm by adding to the initialization a processing of the edges.

Hilditch skeleton

The most complex binary neighbourhood operation is skeletonization. A skeleton algorithm which treats the objects contour by contour and which uses recursivity to its advantage is the Hilditch skeleton.

Hilditch (1969) uses a fixed scan direction (top-down, left to right) and the newly found values for the north and west neighbours are put in the neighbourhoods in the original image to avoid removal of two pixel thick lines. In our skeletonization algorithm, in which a fixed scan direction does not exist (Figure 4), all 4-connected neighbours have to be tested in partially recursive neighbourhoods (see Figure 2, Section 2).

The implementation of the Hilditch skeleton uses three binary images:
- an input image;
- an anchor image, containing the pixels which are by definition skeleton pixels;
- a change image, storing the pixels changed in a current iteration.

The Hilditch skeleton is 8-connected. Hence, the image has to be eroded 4-connected. The queue is initialized with a unanchored 8-connected contour pixels. In the first iteration each queued pixel is removed if it is not a break or end pixel in the original neighbourhood, not a break pixel in the recursive neighbourhood and not a break pixel in the four partially recursive neighbourhoods. Of course each of the latter has to be tested only if that neighbour is set in the change image.

If a pixel may be removed it is set in the change image. Detected break and end pixels are anchored because: 'once a break pixel, always a break pixel' and 'once and end pixel, always an end pixel'.

When all queued contour pixels have been processed, the input image is updated by removing the pixels set in the change image. These pixels can quickly be addressed by passing through the queue a second time. Simultaneously a new queue is built. All unanchored object pixels, 4-connected to the
changed pixels, are queued and temporarily anchored to prevent double queueing. This temporary anchor is removed during the testing phase of the next iteration which is, apart from that, the same as in the first iteration. Note that the temporary anchor can be discerned from the original anchor because the temporary anchor pixels are queued and the original anchor pixels are not. The skeletonization procedure finishes after a desired number of iterations or when stabilization occurs.

The Hilditch skeleton without end pixels proved to be relatively time consuming compared to the Hilditch skeleton with end pixels. Without endpixel condition a pixel which is a breakpixel in some iteration will be a candidate pixel for removal in all next iterations and therefore all one pixel thick lines need to be queued over and over again.

A solution is to obtain the skeleton without end pixels from the skeleton with end pixels by eroding from the end pixels until closed contours or single pixels remain. The anchor which has been filled with detected skeleton pixels has to be restored before the erosion from the end pixels can start.

If objects connected to the edge should stay connected to the edge, the edge is anchored and made object, else the edge is made background before calling the initialization procedure.

A pseudo code of the Hilditch skeletonization can be found in the Appendix.

**Combinations of operations**

Binary neighbourhood operations can be combined to compose higher level operations. Examples of higher level operations are opening (erosion followed by dilation), closing (dilation followed by erosion), separation of objects with and without holes (skeleton without end pixels followed by pepper removal and propagation) or removing small objects (erosion followed by propagation).

These higher level operation can be programmed efficiently with our algorithms by passing the region of interest from one operation to another. Between the operations the image need not be scanned for the contours.

The queue itself cannot be passed on straightforwardly. Some operations end with a queue containing the background contours, others with a queue containing the object contours. Moreover the queue has to be shifted to the start again. Interaction routines to build a queue containing object contour pixels from a queue of background contour pixels and vice versa are easy to construct. In the first case the object neighbours of the queued pixels become the new queue. In the second case the background neighbours of the queued pixels.

The processing time drops dramatically if the queue is passed on. The interaction routine is of time complexity linear in the number of contour elements, whereas a complete initialization is of time complexity linear in the number of image pixels.

**Contour filling and object labelling**

Propagation of interesting pixels (contours) can be used to fill contours and to label objects. In the case of contour filling, a pixel inside a contour is queued and subsequently the area inside the contour is filled by queueing and adding all connected background pixels inside the contour. In the case of object labelling, the image is sequentially scanned. As soon as a non-labelled object pixel is encountered, this pixel is queued and all connected object pixels are labelled subsequently as in the contour filling case. Forks, spirals, etc. do not pose any problems as opposed to a method which scan the image. The time to label an image is proportional to the number of object pixels.

**5. Results**

We have used two 256$^2$ test images to compare the speed of our algorithms, referred to as the VVV-algorithms, with the CLP-algorithms of Groen and Foster (1984) and the PXY-algorithms of Young et al. (1981). The test images were a thresholded image containing gold particles in glass (Figure 5a) and the background of this image (Figure 5b). The objects are to the shape frequently encountered in biomedical cell analysis, the background is used to observe the behaviour of the algorithms when a lot of holes are present in an object. Depending on the operation, the result on completely filled or empty images are given as well, as are the results of straightforward implementations.
The computer used in this experiment was a MicroDutch. This is a VME-bus system built around a Motorola 68020 with a clock frequency of 12.5 MHz., running under UNIX V.2. All programs have been written in C.

Piper (1985) has proposed a method to implement the Hilditch skeleton using interval coding. In his experiments he used a VAX 11/750 running under UNIX which is three to four times slower than the MicroDutch we used. Nevertheless, the performance of his method can be estimated. Assume that his system is four times slower than ours and that his 512² image (Piper, 1985, Figure 5b) is comparable to our 256² image of Figure 5b. Hence, the estimated processing time of his algorithm on our system on the image of Figure 5b is 10 seconds, twice the time of our algorithm.

PXY-algorithms

The PXY-algorithms work on run-length coded images. The run-codes are stored in PXY-tables. A PXY-table contains the addresses of both the start and the stop of each run. Processing a PXY-table means processing only object contours.

Monadic and dyadic point operations (not, and, or, xor) are the basic operations in the PXY-algorithms. A dilation is implemented by shifting the runs in a PXY-table up, down, left and right and by a logical or between the original image and the shifted image. An erosion is implemented as a dilation of the background.

The strength of the PXL-algorithms is the possibility to use the output PXL-table of one operation as input for a next operation. A disadvantage is the large amount of memory space required for complicated images.

CLP-algorithms

The CLP-package of Groen and Foster does not process whole images, but only regions of interest. During one iteration the candidate pixels for the next iteration are stored in a list. Per row the candidate pixels are sorted to obtain runs of successive pixels. In processing these runs a table-lookup is used. The table-index is filled according to the scan-code, so only three new neighbours have to be read for each pixel on a line.

The computational overhead introduced by sorting is large. Therefore a simple sub-optimal procedure often proves to be as fast: instead of building a complete candidate pixel list, only the minimum and maximum column value of the changed pixels are used in setting the region of interest.

The advantages of the methods are their general applicability for binary neighbourhood operations and their limited use of memory. A disadvantage is the time consuming sorting in the optimal CLP-procedure, as will become clear in the next section.

Measurements

The results are summarized in three tables. Table 1 contains the processing times of Figure 5a, Table 2 of Figure 5b and Table 3 of empty and filled images.

Some comments on the experiments. We have only compared the 4-connected algorithms. The results for the 8-connected case show the same trend.
The PXY-propagation was not available. The VVV-skeleton without end pixels consisted of a skeleton with end pixels followed by a pass through the image to detect all non-break pixels and subsequent erosion from these pixels. The initialization uses the first method of Section 4. The VVV-propagation from the edge used a redundant pass through the image to test for other seeds than the edge. If this pass is omitted the processing time becomes 0.5 second less. The CLP-algorithms we used were the optimal algorithms as described by Groen and Foster (1984).

The straightforward implementation (SFW) of the skeleton is a scan-code algorithm. At each pixel a table entry is composed of the neighbours by shifting and refreshing three new values. The recursive and partially recursive neighbourhoods are constructed from this entry as well. In the straightforward implementation of the other operations for each pixel the conditions as described in Section 2 are evaluated.

The VVV-algorithms calculate a skeleton 3 to 4 times faster than the CLP-algorithms. The dilations and erosions are an order of magnitude faster than the CLP-algorithms and almost 1.5 to 2 times as fast as the PXY-algorithms. The processing times of the straightforward implementations show that intelligent algorithms have their benefits, but only if the overhead is kept small.

### 6. Conclusion

We have implemented and tested algorithms for binary neighbourhood operations. The operations are erosion, dilation, propagation and skeletonization. The algorithms process only object contours by storing pointers to relevant pixels.

Our new algorithms are the fastest in all tests. Compared to the PXY-erosion and -dilation of Young et al. (1981), the gain in processing time is small, but since these algorithms do not offer a reasonable skeleton, the large difference with the next available alternative, the CLP-algorithms of Groen and Foster (1984), is more important.

From the basic operations other operations can easily be derived, such as contour filling and object labeling. If operations are combined, e.g. opening and closing, initialization is redundant and the algorithms become even faster.
Acknowledgement

We thank I.T. Young, R.C. Peverini, P.W. Verbeek, P.J. van Otterloo, F.C.A. Groen and N.J. Foster for making their software available to us, and TPD - TNO (Institute of Applied Physics Delft) for using their computer system.

References


Appendix

The appendix contains the pseudo of the erosion, dilation and the Hilditch skeleton.

Erosion 4-connected/8-connected

begin
make image edge zero
initialize the queue with pointers to pixels 4-/8-connected to the background
for all pointers on queue do
remove pixel pointed to
endfor
while (old_queue not empty and still iterations to go) do
for all pointers on old_queue do
for all horizontal & vertical/diagonal neighbours do
if neighbour = 1 then
neighbour = 0
put pointer to neighbour on new_queue
endif
endfor
endfor
old_queue = new_queue
endwhile
end

Dilation 4-connected/8-connected

begin
make image edge one
initialize the queue with pointers to pixels 4-connected to the background
for all pointers on queue do
for all 4-connected/8-connected neighbours do
  if neighbour = 0 then
    neighbour = 1
    put pointer to neighbour on new queue
  endif
endfor
endfor
while (old_queue not empty and still iterations to go) do
  for all pointers on old_queue do
    for all horizontal & vertical/diagonal neighbours do
      if neighbour = 0 then
        neighbour = 1
        put pointer to neighbour on new_queue
      endif
    endfor
  endfor
  old_queue = new_queue
endwhile
end

Hilditch skeleton
begin
  make image edge anchor
  initialize the queue with pointers to the 8-connected unanchored contour pixels
  while (old_queue not empty and still iterations to go) do
    for all pointers on old_queue do
      if ($\gamma_n = 1$ in non-recursive, recursive and partially recursive
        neighbourhoods and $\Phi_8 \neq 1$ in non-recursive neighbourhood) then
        set pixel in change image
      else
        anchor pixel
      endif
    endfor
    for all pointers on old_queue do
      if pixel pointed to is set in change image then
        remove pixels in change, mask and anchor image
      endif
      for all horizontal & vertical neighbours do
        if (neighbour = 1 and neighbour $\neq$ anchor) then
          anchor the neighbour temporary
          put pointer to neighbour on new_queue
        endif
      endfor
    endfor
    old_queue = new_queue
  endwhile
end
B.J.H. Verwer

IMPROVED METRICS IN IMAGE PROCESSING APPLIED TO THE HILDITCH SKELETON

Proceedings 9th International Conference on Pattern Recognition,
Rome, November 14-17, 1988, 137-142.

5

‘He sent it by the usual means,’ Iff shrugged. ‘A P2C2E.’
‘And what is that?’
‘Obvious,’ said the Water Genie with a wicked grim.
‘It’s a Process Too Complicated To Explain.’

Salmon Rushie, Haroun and the Sea of Stories
IMPROVED METRICS IN IMAGE PROCESSING
APPLIED TO THE HILDITCH SKELETON

Ben J.H. Verwer
Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology
Lorentzweg 1, 2628 CJ Delft, The Netherlands

ABSTRACT

An algorithm is presented in which binary object pixels are addressed and processed in order of increasing distance to the background. The distances are defined as path lengths. The metric can be chosen as to obtain arbitrary good approximations of the Euclidean metric.

The algorithm incorporates an efficient propagation method in which extensive use is made of directional information.

The algorithm is applied to the Hilditch skeleton and proves to be as fast as the best other Hilditch skeletonization algorithms available in software.

1. INTRODUCTION

Traditional metrics in image processing are the cityblock metric and the chessboard metric [1]. With respect to isotropy and accuracy, however, they do not perform satisfactorily. Other metrics can be used [2] [3], which approximate the Euclidean metric better in view of isotropy and can even approximate it to an arbitrary degree with respect to the maximum relative error [4]. Borgerstors [3] called these metrics chamfer metrics. In our research we have used reasonable integer approximations of the real-valued optimal coefficients [5] [6].

Section 2 recapitulates the idea of chamfer metrics and gives some practical values for the coefficients of the metric. Section 3 describes a new algorithm, in which pixels are accessed in order of increasing chamfer distance.

Many applications in image processing can utilize good accuracy and good isotropy. We here only mention pattern recognition applications in which the pre-processing step has to be rotation independent, e.g. straightening chromosomes in cytology and matching sensory input to model data in robotics.

Moreover, the algorithm addresses pixels directly and uses an efficient propagation method using directional information. This allows high performance in software and offers perspective for hardware [7].

Section 4 gives details for an implementation of the Hilditch skeletonization method [8], shows the improved accuracy and isotropy and compares processing times with existing implementations. Section 5 concludes the paper.

2. THE CHAMFER METRIC

In the chamfer metric [2] a distance d is defined as a minimum path length. A path is defined as a sequence of basic vectors b_i, a path length as the sum of the (chamfer) distance values d_i associated with the basic vectors b_i.

The three conditions for a metric [1] are:
1. d(p,q) = 0, if and only if p = q;
2. d(p,q) = d(q,p);
3. d(p,r) ≤ d(p,q) + d(q,r).

Condition 1 can be fulfilled if only the chamfer distance of the null-vector is zero; condition 2 if the basic vectors which have the same Euclidean distance receive the same chamfer distance. Conditions 3 follows directly from the definition of distance as minimum path length.

In two dimensions the elementary basic vectors are (x_1,x_2) with x_1, x_2 ∈ {0,±1} (see figure 1). In that case the maximum relative error is minimized by associating chamfer distance 0.9804 with those vectors b_1, in which only one of x_1, x_2 ∈ {0} and length 1.3583 with the vectors b_2 in which none of x_1, x_2 ∈ {0}. Other used error measures are the maximum absolute error [2] and the mean square error [9].

A good integer approximation is the pair of coefficients 5 and 7 (see table 1, from [5] [6]). In integer approximations only the ratio is optimized, after which a scalar correction is sufficient to obtain absolute distances.

Figure 1. The elementary basic vectors b_1 and b_2 of the 2-D rectangular grid.

Non-elementary basic vectors or basic vectors which reach outside a 3x3 neighbourhood can be used to obtain better accuracies. In [4] it is proven that the extension guarantees that a specified maximum relative error is obtained. In table 1 the errors are specified for some integer approximations of the basic vectors b_1 and b_2 plus the so-called 'knight's move' vector b_3 (see figure 2).

Figure 1. The non-elementary basic vector b_3 (knight's move).

We will call the neighbours reached from a focal pixel with such a knight's move k-connected to the focal pixel, in addition to the 4- and 8-connected neighbours. As a reasonable integer approximation we will use the distance values 5, 7 and 11.

CH2614-6/88/0000/0137$01.00 © 1988 IEEE
Table 1. Integer approximations for the distance values $d_1$, $d_2$, $d_3$, plus the required scaling factor and the maximum relative error.

<table>
<thead>
<tr>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>scaling</th>
<th>error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>2.12</td>
<td>5.57</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>5.17</td>
<td>4.21</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>17</td>
<td>12.50</td>
<td>4.00</td>
<td></td>
</tr>
<tr>
<td>best set (real valued)</td>
<td></td>
<td></td>
<td>3.96</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>4.12</td>
<td>2.94</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>5.01</td>
<td>1.79</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>9.01</td>
<td>1.52</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>24</td>
<td>17.22</td>
<td>1.43</td>
<td></td>
</tr>
<tr>
<td>best set (real valued)</td>
<td></td>
<td></td>
<td>1.36</td>
<td></td>
</tr>
</tbody>
</table>

3. THE ALGORITHM

The algorithm starts with detection of the background contour. This requires one scan through the image, in which pointers to all background pixels 4-connected to the object are stored (a pointer is a variable which contains the address of a pixel in memory). Subsequently the pixels are processed in order of increasing chamfer distance. For that we need a proper storage mechanism. We propose to use a bucket structure [10].

The bucket structure consists of a number of buckets, numbered 0,1,2... Each bucket can store pixels in the form of an address field plus a direction field. We designate bucket $d$ to store pixels with a distance value $d$.

In the initializing scan the pointers to the background pixels are stored in bucket 0. The direction field is not yet used. We then fill the other buckets one by one, putting a pixel with distance value $d$ in bucket $d$.

Suppose we are processing distance $d$. We note that distance value $d$ can be obtained from a 4-connected neighbour with distance value $d-1$, from a 8-connected neighbour with distance value $d-2$ or from an $k$-connected neighbour with distance value $d-3$. We could thus recall all pixels from bucket $d-1$, which has been processed already, and put all object pixels 4-connected to these pixels in bucket $d$. Of course bucket $d-2$ and $d-3$ should be processed analogously (our approach combines in this the 'reading backward' from dynamic programming (breadth-first though), and the uniform cost propagation from the A*-algorithm (writing forward though) [10] [11].

We did not yet mention the purpose of the directional field. We can save a lot of computational effort by observing that only a few of the possible 4, 8 and $k$-connected neighbours have to be checked if we know from which direction a recalled pixel was generated. For all other neighbours can be reached via a path of equal or less length. In the case of two paths of equal length we only use one of them (the path which starts with the largest chamfer distance; this choice is not entirely free if the algorithm is the be used for skeletonization, see next chapter).

In figure 3a a visualization of the principle is presented. Although the principle to leave out "all candidates that can be reached via a path of equal or less length" seems obvious a full proof is not easy and falls beyond the scope of this paper.

Two remarks: the initiating background pixels have to generate in all directions and the number of buckets required is $d_3+1$ since bucket $d_3$ is of no more use when processing bucket $d$ (only buckets $d_3$ to $d$ are required). We therefore reuse bucket 0 for storing pixels with distance values $d_3+1$.

It is obvious that the algorithm is extendible to larger sets of basic vectors. The number of buckets required is then $d_{\text{max}}+1$.

Summarizing, the power of the algorithm is the integration of:

1. A good metric;
2. Direct addressing;
3. Simple and efficient storage;
4. Simple and efficient propagation.

3. THE MILDITCH SKELETON

3.1. Introduction

Skeletonization is an often used operation in image processing. Its importance lies in the data reduction of a set of pixels (the binary image) to a set of line segments (the skeleton).

In the continuous case we define the skeleton of an object to be a line connecting points which lie midway between the boundaries of the object, with the same topology as the object itself.
In the discrete case this leads to 4 conditions for the skeleton of an object:
- It has the same connectedness as the object;
- It is one pixel thick;
- It preserves endpixels;
- It lies in the middle of the object.

Some remarks:
1. Connectedness is defined in the usual 4- or 8-fold way on a rectangular grid.
2. The conditions listed above can be in conflict (e.g. figure 4), in which the skeleton is not 1 pixel thick, but in which the connectedness condition does not allow any of the skeleton pixels to be removed. Some conditions are therefore more important than others. Usually the ordering as listed above is taken, connectedness being the most important condition, accuracy the least important.

![Image](image.png)

Figure 4. A pathological image in which the object is its own skeleton and in which none of the pixels may be deleted, though the object is not one pixel thick.

3. Note that some authors call our skeleton-operation thinning and reserve the word skeleton for a not-necessarily connected set of points from which an object can be reconstructed (e.g. Montenari [12]).

The conditions have lead to many different skeletonization algorithms, which in turn lead to different skeletons. We will use the Hilditch skeletonization method [6], though our approach is not restricted to this method. Algorithms based on other methods should be easy to implement.

### 3.2. The Hilditch skeleton

Hilditch [9] uses non-recursive, recursive and partially recursive neighbourhoods. Non-recursive neighbourhoods are neighbourhoods in the image before a particular iteration started. Recursive neighbourhoods are neighbourhoods in the image in which the values calculated are updated immediately; partially recursive neighbourhoods consist of seven neighbours taken from a non-recursive neighbourhood and one neighbour taken from a recursive neighbourhood.

Non-recursive neighbourhoods are used to avoid removal of end and break pixels; recursive neighbourhoods to avoid removal of single pixels and two pixel thick lines; partially recursive neighbourhoods to avoid erosion of two pixel thick lines from the end. Which partial recursive neighbourhoods have to be tested is dependent on the scan direction. Hilditch used a fixed scan direction (top-down, left to right) and the newly found values for the north and west neighbours are put in the neighbourhoods in the original image to avoid removal of two pixel thick lines. In our skeletonization algorithm, in which a fixed scan direction does not exist, all 4-connected neighbours have to be tested in partially recursive neighbourhoods.

The crossing number of Hilditch $\gamma_h$ represents the connectivity:

$$\gamma_h = \sum_{k=1,2,3,4} h_k$$

$\Phi_4$ counts the number of 4-connected neighbours:

$$\Phi_4 = \sum_{k=1,2,3,4} p_k$$

$\Phi_8$ counts the number of 8-connected neighbours:

$$\Phi_8 = \sum_{k=1,2,\ldots,8} p_k$$

where $p_k$ denotes the binary pixel value at position $k$ (figure 5) in the neighbourhood (figure 6) and where $h_k = 1$ if $(p_{2k-2} = 0)$ and $(p_{2k-1} = 1$ or $p_{2k} = 1)$, with $p_8 = p_0$, else $h_k = 0$.

![Image](image.png)

Figure 5. 3*3 neighbourhood numbering.

![Image](image.png)

Figure 6. Non-recursive, recursive and partially recursive neighbourhoods.

If and only if the crossing number equals one, may a pixel be removed. If the crossing number exceeds one, a pixel is a link- or vertex pixel. If the crossing number equals zero, the pixel is either a single pixel or lies in the middle of an object.

Endpixels are preserved by counting the number of neighbours in the non-recursive neighbourhood. If a pixel has only one 8-connected neighbour it may not be deleted.

Summarizing, a pixel may be deleted ($p_8 = 1 \rightarrow p_8 = 0$), if it is a contour pixel ($\Phi_2 < 4$), if the crossing number is 1 ($\gamma_h = 1$) and if it is not an endpixel ($\Phi_8 = 1$).

The conditions can be implemented in two tables. One with end-pixel condition for the non-recursive neighbourhoods and one without end-pixel condition for the recursive and partially recursive neighbourhoods [14].
3.3 The Implementation

The Hilditch skeletonization method is implemented in our algorithm by testing the Hilditch conditions each time when all pixels with a certain distance value have been stored.

Two bitplanes are required: one recursive and one non-recursive. We propose to use the input and the output bitplane. The algorithm starts with copying the input bitplane to the output bitplane. Then the buckets are filled. A pixel stored during the 'bucketing' phase is removed temporarily from the output bitplane to prevent double storage. The Hilditch conditions are not yet tested! When all the pixels with a certain distance value have been found the temporarily removed output bitplane is first set again by recalling the pixels just stored. Next the pixels are recalled again, but now the Hilditch conditions are tested. If a pixel may be deleted, it is removed from the input bitplane. Now the non-recursive values can be accessed in the output bitplane and the recursive values from the combination of a set output bitplane and a reset input bitplane. Note that it is not sufficient to remove the output bitplane because in that way the non-recursive values would get lost.

An iteration finishes by resetting those pixels in the output bitplane, which are reset in the input bitplane and by setting the input bitplane again.

Note that pixels not deleted can be addressed again. In the case of skeletonization the double processing is negligible and the result does not change (once a breakpixel, always a breakpixel). In applications where the result can change an extra bitplane is required to separate the information 'has been processed' from 'output result'.

One modification has to be made to the basic algorithm. This is due to the discrepancy between the metric and the connectivity. In figure 7 the central pixel has the distance value 100. The object is a skeleton pixel because it keeps the connectivity between the border and the large unprocessed block. Its north neighbour will receive the distance value 105 and may be deleted, for the connectivity is preserved via the north-east neighbour with distance value 106 (which of course is processed after the pixels with distance value 105).

However, the rule of 'following the steepest gradient' [13] is not obeyed here (since (105-100)/5 > (106-100)/7). We are saved by the observation that the pixel with value 105 is generated from a skeleton pixel, so if we do not delete pixels which could have been generated from 4-connected skeleton pixels, the steepest gradient will be followed! The north-east pixel then of course will be removed subsequently.

Now the reader can understand why in the preceding chapter, to resolve paths of equal length, the paths with the largest chamfer distance first were chosen. In that case 4-connected neighbours are the first to generate successors, so the "could have been generated" is replaced by "are generated". The last statement is more easy to implement.

3.4 Accuracy

It is not possible to give an explicit error measure for a calculated skeleton relative to the 'perfect' skeleton, because a 'perfect' skeleton is not defined. This in view of the conflicting conditions as mentioned above. The reader can judge the performance of our algorithm with respect to the 'city-block' Hilditch algorithm by studying figures 8 . On top the 'city-block' Hilditch-skeletons are shown. Below the pseudo-euclidean Hilditch-skeletons.

Our skeletons in this case lie on top of the 'perfect' skeletons as defined by Dorst en Van Antwerpen [13]. As will be explained in the next section, their skeleton is obtained directly from a distance transformed image. The disadvantage of their skeleton is that it is not always one pixel thick, so in many applications a post-processing step is required.

An implicit measure can be given by considering the skeleton of figure 9. The 4 images contains holes, the line connecting which makes different angles with the horizontal.

The original Hilditch skeletons quickly converge to 2 of the 4 main directions of the grid, the horizontal, the vertical and the two diagonals. The new Hilditch skeleton does not converge to 2 of the 4 main directions of the rectangular grid but to 2 of the 16 preferred directions of the hexagonal chamfer metric. This can be understood by remembering the basic vectors of the metrics.
3.5 Processing times

Our discussing of processing times focuses around the measurements made in [15]. In that paper several sequential implementations of Hilditch' skeletonization were compared. We repeat the figures in table 2 together with the results of our algorithm. Method CLP of [14] uses a line based region of interest in which a sorting takes place after each iteration. Method SFW is a straightforward implementation, in which each pixel is processed once in each iteration. Method VVV [15] uses a similar approach as in this paper, based on city-block distances. The VVV method yields the traditional Hilditch skeleton. The city-block distances allow a much simpler storage mechanism, a single queue. It must be noted that the VVV-algorithm could be sped up by using directional information.

<table>
<thead>
<tr>
<th>Method</th>
<th>Figure 11</th>
<th>Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLP</td>
<td>12.0</td>
<td>19.5</td>
</tr>
<tr>
<td>VVV</td>
<td>3.4</td>
<td>5.3</td>
</tr>
<tr>
<td>SFW</td>
<td>22.3</td>
<td>34.6</td>
</tr>
<tr>
<td>New</td>
<td>3.0</td>
<td>5.0</td>
</tr>
</tbody>
</table>

The test images were a thresholded image containing gold particles in glass (figure 10) and the background of this image. The objects are of the shape frequently encountered in biomedical cell analysis, the background is used to observe the behaviour of the algorithms when a lot of holes are present in an object.

The computer used in this experiment was a MicroDutch. This is a VME-bus system built around a Motorola 68020 with a clock frequency of 12.5 MHz., running under UNIX V.2. All programs have been written in C.

From the table it can be concluded that improved accuracy does not deteriorate the processing time.

3.6 Memory allocation

Figure 11 gives some allocation data plus the dependency of the amount of memory allocated and the processing time on the chunksize of the dynamic allocation procedure. The chunksize is the number of pixels stored in one chunk of a bucket, new chunks being allocated or old chunks used when a chunk has been filled completely.

![Figure 11. Memory requirements and processing times for the new algorithm as a function of chunksize. The test image was the image of figure 11.](image-url)
As can be seen, the memory requirements are moderate, being proportional to the contour size of the objects in the image. It can also be seen that the processing time does not decrease noteworthy with chunksizes larger than 100 pixels, whereas the optimal chunksize for memory allocation is somewhere around 30 pixels. The graph is of course data-dependent, but the general shape and numbers will be the same for different images.

4. CONCLUSION

Pixels in images can be processed and accessed in order of increased distance. The distances are defined as minimum path length and can be chosen arbitrarily close to euclidean distances. The integrated approach using direct addressing and directional information leads to fast algorithms.

Applied to skeletonization the approach proved to work with a minor modification. Pixels whose distance values are generated from skeleton pixels should not be deleted.

The comparison with other skeletonization methods proved the better accuracy and the favorable speed.

REFERENCES

EPILOGUE TO CHAPTER 5

Three dimensional skeleton

The paper can be generalized to three dimensions. In the
5*5*5 neighbourhood there are 98 neighbours, 6 (1,0,0),
12 (1,1,0), 8 (1,1,1), 24 (2,1,0), 24 (2,1,1) and 24
(2,2,1). The errors in distance values are treated in
chapter 3. E.g. for the integer combination \(c_{100}=9,\)
\(c_{110}=13, c_{111}=16, c_{210}=20, c_{211}=22, c_{221}=27,\)
a maximum error of 2.7% is achieved. The improvement
over the three dimensional cityblock and chessboard
equivalents, maximum error 27%, is enormous.

The first skeletonization algorithm for three dimensions
appeared in 1980 (Lobregt et al.). The topology preserv-
ing test was based on the Euler number in a 3*3*3 neigh-
bourhood, being defined as the number of objects minus
the number of rings plus the number of caves. The Euler
number could be incrementally calculated by table lookup
in 8 2*2*2 subcubes (note that the usual table lookup
procedure for the complete neighbourhood is impossible
in three dimensions: \(2^7 = 10^6\) entries would be required).

Toriwaki (1982) showed that in a 3*3*3 neighbourhood
the Euler number can remain unchanged while the topo-
logy changes, e.g. removing the central pixel can create
an extra ring and increase the number of objects from one
to two (figure 1). Since for 6-connected objects only the
pixels in the 18-neighbourhood are of importance a table
lookup is possible to determine the number of objects
(\(2^{18}\) entries can be coded in 32k (Verwer, 1983).
Combined with the Euler number this information suf-
fices to preserve the topology.

As in two dimensions, in three dimensions a large portion
of the neighbours need not be addressed if the directions
are kept. Assuming that all directional vectors occur
equally often, only 4.5% of the neighbours are relevant.
Table 1 shows the required successors.

\[\text{Table 1 shows the required successors.}\]

\[\text{\textsuperscript{1} Verbeek et al. (1985) proposed an alternative method, which}\]
\[\text{requires tables of only 4096 entries.}\]

\[\text{\textsuperscript{2} 100-connected neighbours have 54 relevant successors on a total}\]
\[\text{of 6 successors times 98 possible directions. 110-connected 108,}\]
\[\text{111-connected 56, 210-connected 120, 211-connected 72 and 221-}\]
\[\text{connected 24:}\]

\[\frac{54+108+56+120+72+24}{6+12+8+24+24+24} * 98 = 4.5\% \text{ relevant successors}\]

Figure 1. If the pixel below the dotted pixel is
removed, the number of objects increases from
1 to 2 and the number of rings from 0 to 1.

Figure 2 shows a test image of 12 partially overlapping
spheres, positioned at the corners of an icosahedron in a
100*100*100 image. It has a cave in the center like the
object in figure 1 of the paper. In the lower left corner the
skeleton calculated by the the new algorithm is shown, in
the lower right corner the skeleton calculated by the old
algorithms (Toriwaki 1982, Lobregt 1980, Verbeek
1985). Both skeletons consist of a one pixel thick ‘skin’,
but the icosahedral form cannot be recognized in the
skeleton calculated by the old algorithms.

The processing time required to skeletonize the image of
figure 2 was 145s on a SUN 3/60. 86% of the time was
used in connectivity tests.

The ordering by distance took 21s. An ordinary distance
transform in two scans (Borgefors 1986) takes 66s. So
the algorithm is valuable as distance transform sec as
well. For skeletonization it has the extra benefit of
addressing pixels in order of increasing distance.

On a SUN 4/280 the ordinary distance transform takes
place in 16s while the new distance transform takes 9s.
Skeletonization takes 37s. The new distance transform is
thus approximately two times faster on a SUN 4/280 but
time times faster on a SUN 3/60. This is due to the
memory cache of the SUN4, which is utilized better by
scanwise addressing.

The memory required to store the addresses of the pixels
was 66kbyte. This is negligible compared to the 1Mbyte
the image itself occupies.
<table>
<thead>
<tr>
<th>index</th>
<th>vector</th>
<th>100</th>
<th>110</th>
<th>111</th>
<th>210</th>
<th>211</th>
<th>221</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2 2 1</td>
<td>-</td>
<td>23</td>
<td>-</td>
<td>22</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2 2 -1</td>
<td>-</td>
<td>23</td>
<td>-</td>
<td>24</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2 1 2</td>
<td>-</td>
<td>27</td>
<td>22</td>
<td>26</td>
<td>7</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>2 1 1</td>
<td>28</td>
<td>27</td>
<td>23</td>
<td>22</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2 1 0</td>
<td>28</td>
<td>23</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2 1 -1</td>
<td>28</td>
<td>29</td>
<td>23</td>
<td>24</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>2 1 -2</td>
<td>28</td>
<td>29</td>
<td>-</td>
<td>24</td>
<td>30</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>2 0 1</td>
<td>28</td>
<td>27</td>
<td>-</td>
<td>7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>2 0 -1</td>
<td>28</td>
<td>27</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>2 -1 2</td>
<td>-</td>
<td>27</td>
<td>-</td>
<td>32</td>
<td>26</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>2 -1 1</td>
<td>28</td>
<td>33</td>
<td>27</td>
<td>32</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>2 -1 0</td>
<td>28</td>
<td>33</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>2 -1 -1</td>
<td>28</td>
<td>33</td>
<td>29</td>
<td>34</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>2 -1 -2</td>
<td>28</td>
<td>33</td>
<td>-</td>
<td>34</td>
<td>30</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>2 -2 1</td>
<td>28</td>
<td>33</td>
<td>-</td>
<td>34</td>
<td>38</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>2 -2 -1</td>
<td>28</td>
<td>33</td>
<td>-</td>
<td>34</td>
<td>38</td>
<td>11</td>
</tr>
<tr>
<td>16</td>
<td>1 2 2</td>
<td>45</td>
<td>44</td>
<td>-</td>
<td>22</td>
<td>43</td>
<td>41</td>
</tr>
<tr>
<td>17</td>
<td>1 2 1</td>
<td>45</td>
<td>44</td>
<td>23</td>
<td>22</td>
<td>41</td>
<td>18</td>
</tr>
<tr>
<td>18</td>
<td>1 2 0</td>
<td>45</td>
<td>23</td>
<td>-</td>
<td>18</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>1 2 -1</td>
<td>45</td>
<td>46</td>
<td>23</td>
<td>24</td>
<td>42</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>1 2 -2</td>
<td>45</td>
<td>46</td>
<td>-</td>
<td>24</td>
<td>47</td>
<td>42</td>
</tr>
<tr>
<td>21</td>
<td>1 1 2</td>
<td>48</td>
<td>44</td>
<td>27</td>
<td>22</td>
<td>43</td>
<td>26</td>
</tr>
<tr>
<td>22</td>
<td>1 1 1</td>
<td>48</td>
<td>27</td>
<td>-</td>
<td>22</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>1 1 0</td>
<td>48</td>
<td>-</td>
<td>27</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>1 1 -1</td>
<td>48</td>
<td>-</td>
<td>23</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>1 1 -2</td>
<td>49</td>
<td>46</td>
<td>29</td>
<td>24</td>
<td>47</td>
<td>30</td>
</tr>
<tr>
<td>26</td>
<td>1 0 2</td>
<td>48</td>
<td>27</td>
<td>-</td>
<td>26</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>1 0 1</td>
<td>48</td>
<td>27</td>
<td>-</td>
<td>26</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td>1 0 0</td>
<td>48</td>
<td>-</td>
<td>27</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>29</td>
<td>1 0 -1</td>
<td>48</td>
<td>-</td>
<td>29</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>1 0 -2</td>
<td>49</td>
<td>29</td>
<td>-</td>
<td>30</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>31</td>
<td>1 -1 2</td>
<td>48</td>
<td>51</td>
<td>27</td>
<td>32</td>
<td>50</td>
<td>26</td>
</tr>
<tr>
<td>32</td>
<td>1 -1 1</td>
<td>48</td>
<td>33</td>
<td>-</td>
<td>32</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>33</td>
<td>1 -1 0</td>
<td>48</td>
<td>33</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>34</td>
<td>1 -1 -1</td>
<td>48</td>
<td>-</td>
<td>34</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>35</td>
<td>1 -1 -2</td>
<td>49</td>
<td>53</td>
<td>29</td>
<td>34</td>
<td>54</td>
<td>30</td>
</tr>
<tr>
<td>36</td>
<td>1 -2 2</td>
<td>52</td>
<td>51</td>
<td>-</td>
<td>32</td>
<td>55</td>
<td>50</td>
</tr>
<tr>
<td>37</td>
<td>1 -2 1</td>
<td>52</td>
<td>51</td>
<td>33</td>
<td>32</td>
<td>55</td>
<td>38</td>
</tr>
<tr>
<td>38</td>
<td>1 -2 0</td>
<td>52</td>
<td>33</td>
<td>-</td>
<td>38</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>39</td>
<td>1 -2 -1</td>
<td>52</td>
<td>53</td>
<td>33</td>
<td>34</td>
<td>56</td>
<td>38</td>
</tr>
<tr>
<td>40</td>
<td>1 -2 -2</td>
<td>52</td>
<td>53</td>
<td>-</td>
<td>34</td>
<td>56</td>
<td>54</td>
</tr>
<tr>
<td>41</td>
<td>0 2 1</td>
<td>45</td>
<td>44</td>
<td>-</td>
<td>41</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>42</td>
<td>0 2 -1</td>
<td>45</td>
<td>46</td>
<td>-</td>
<td>42</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>43</td>
<td>0 1 2</td>
<td>48</td>
<td>44</td>
<td>-</td>
<td>43</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>44</td>
<td>0 1 1</td>
<td>44</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>45</td>
<td>0 1 0</td>
<td>45</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>46</td>
<td>0 1 -1</td>
<td>46</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>47</td>
<td>0 1 -2</td>
<td>49</td>
<td>46</td>
<td>-</td>
<td>47</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>48</td>
<td>0 0 1</td>
<td>48</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Figure 2. Improved 3-d skeleton. At the top the test image consisting of 12 partially overlapping spheres. The object has a cave in it. In the lower left corner the improved version of the 3-d skeleton, in the lower right corner the old version.
P.W. Verbeek and B.J.H. Verwer

2-D ADAPTIVE SMOOTHING BY 3-D DISTANCE TRANSFORMATION


'It's amazing what you can get accustomed to, and at what speed,' Haroun reflected. 'This new world, these new friends: I've just arrived, and already none of it seems very strange at all.'

Salmon Rushie, Haroun and the Sea of Stories
2-D adaptive smoothing by 3-D distance transformation

P.W. VERBEEK and B.J.H. VERWER

Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands

Received 8 July 1988
Revised 7 October 1988

Abstract: Sternberg's rolling ball method for image smoothing demands specification of filter size. We want to make filter size adaptive. A criterion for smoothing heaviness stays necessary, but, unlike size, should be scale invariant. Hence we have pursued adaptation by first smoothing at all sizes and then selecting the best result.

The usual implementation is: first dilation, then erosion of the binary 3-D grey landscape. We have found an alternative implementation: first the 3-D medial axis of the landscape complement, then erosion of the landscape from the medial axis points. Both implementations are slow, but both can be speeded up through 3-D distance transformation, as needed for multisize operation. Although distance transformation yields virtually parallel multisize erosion, dilation and medial axis calculation on a single landscape, it cannot maintain this efficiency in eroding different dilation results or eroding from the medial axis.

We therefore have switched our interest from global to local adaptation. Unlike the usual implementation which can only be extended to global adaptation, the medial axis approach allows extension to local adaptation as well.

As a scale invariant criterion for smoothing heaviness we propose the number of objects or features left over after smoothing. The medial axis can easily be broken up into countable fragments, related to image segments. The importance of segments can be estimated from the values on the medial axis fragment. A given number of most important segments can thus be selected for local small size (detail conserving) closing. As both light objects on dark background and dark objects on light background can be important, the method simultaneously processes the grey landscape and its complement.

Key words: Adaptive, distance transform, filtering, image processing, medial axis, rolling ball, segmentation, smoothing.

1. Introduction

Many methods are known to smooth an image. Global or local, linear or nonlinear, they are most often not scale invariant, do not commute with scaling, have a built-in filter size, in short give a qualitatively different result when applied to differently sized copies of an image.

In the present paper we propose a method for adaptive smoothing. Rather than filter size, the number of objects (or other image features) remaining after smoothing is taken as a measure of smoothing. In Section 2 we discuss Sternberg's rolling ball smoothing, our starting point. Section 3 shows generalizations of the ball shape. Section 4 gives the demand for symmetry between black and white. Section 5 presents the need for adaptivity. Section 6 summarizes the demands. Section 7 is devoted to multisize opening by distance transformation. Section 8 shows how adaptive smoothing can be achieved. Section 9 treats dilation from the medial axis parts. In Section 10 we discuss some practical results and in Section 11 we give conclusions.

2. Rolling ball smoothing

The 3-D graph of grey value as a function of position in a 2-D grey image can be considered as a hilly landscape and so as a 3-D binary image (umbra). Sternberg [1] has proposed smoothing of the 2-D grey image by closing (dilation followed by erosion)
of the 3-D binary image. The structuring element originally used was a sphere of fixed radius, hence the name rolling ball algorithm.

Rolling ball smoothing can be visualized as follows: Cover the landscape with wet plaster, roll the ball over the landscape so that valleys are filled up to the level where the ball can touch it (see Figure 1).

Opening (erosion followed by dilation) yields a different kind of smoothing, rounding peaks. It is equivalent to closing the complement of the landscape and taking the complement of the result. It can be visualized by rolling a virtual ball under the landscape surface.

In dilation/erosion terms: erode with a sphere (fixed radius) as structuring element, i.e. remove a layer $L_1$ of overlapping spheres, with centers on the landscape surface $S_1$. The layer $L_1$ defines a new surface $S_2$. Then dilate with the same structuring element by adding the slightly different layer $L_2$ of similar spheres now centered on $S_2$. This results in the smoothed surface $S_3$ (Figure 2).

For reasons to become clear we prefer to discuss smoothing in terms of opening. When in need of closing the landscape we shall open its complement.

3. Non-spherical structuring elements

As far as the choice of structuring elements is concerned the argument is the same for closing and opening.

3.1. Paraboloid opening

The arbitrary relative scaling of grey-level axis and $x$-$y$-axes poses a problem. One would like to have a simple relation between smoothing results at different scalings. A sphere does not simply become a sphere of different radius under scaling but becomes an ellipsoid. Indeed an ellipsoid simply becomes an ellipsoid with different size and eccentricity. However, one would like the structuring element to be characterized by only one parameter: size. The paraboloids Sternberg uses in his later work (cf. [2]) do have one size parameter (focal length). Under scaling $g \rightarrow a \cdot g$, $r \rightarrow b \cdot r$, a paraboloid ($g = (r/s)^2$, size $s$) does simply become a paraboloid of different size

$$s' = \frac{a}{b}s.$$

3.2. Poweroid and anisotropic opening

A similar scaling invariance holds in general for structuring elements defined by a power law $g = (r/s)^n$, where the structuring element may even be anisotropic. (Then $s = s(\phi)$ and $s' = (a^{1/n}/b)$ $s(\phi)$). For

$$s(\phi) = s_0 \min \left\{ \frac{1}{\cos(\phi)}, \frac{1}{\sin(\phi)} \right\}$$

the structuring element is a square.

3.3. Maxmin opening

For $n = \infty$, the structuring element is a cylinder
with flat bottom; opening and closing amount to Serra’s grey opening and closing (maxmin and min-max filters, [3], [4]).

3.4. Additive constants in grey value

Replacing the structuring element by a translated copy does not change the opening result. The shift obtained after erosion is compensated in the dilation process. Poweroids \( g = (r/s)^n \) can be extended to include shifted poweroids \( g = (r/s)^n + c \) without complications.

4. Black/white symmetric smoothing

Closing the landscape fills holes, opening removes peaks. We want to do both, hence we open both the landscape and its complement. In doing so we maintain symmetry between dark and light objects. The two opening results must be combined in some symmetric way. The surface of the opened landscape (the lower envelope [4]) is always below that of the opened complement (the upper envelope). One way of combining them is to take the average grey value (altitude) at each position; another is to dilate both surfaces until they meet and take the meeting points as the result. An asymmetric way would be closing the landscape, followed by opening or opening followed by closing [5]. Taking the average would symmetrize this at the cost of double computation time.

5. Structuring element size, an unwanted magic number

In most applications one finds the structuring element size, such as the size of the rolling ball, by trial and error. Even then it is not always possible to determine one size that will do for the whole image.

We want to eliminate size as a magic number, i.e. let size be determined by the image content, adaptively. Still, there will always be a measure of smoothing indicating where between original and blank image a smoothing result is situated, how much detail has been removed. In short: choosing a smoothing measure is ordering according to detail.

Now the task remains to give a definition of ‘detail’. Traditionally, detail was ‘the high frequency (frequency \( > f_0 \)) contribution in the Fourier spectrum’; when rolling the ball it is ‘what is filled in by small \( (\text{size} < s_0) \) balls’. These definitions have in common that they depend on image scale rather than content. In general, different problems may ask for different definitions. In other words, the aim of the smoothing should be made explicit before a smoothing measure can be defined. Our aim will be a rough rendition of the image, with main features such as clear edges and large smooth areas unharmed.

We believe that features should be ordered according to prominence and that the smoothing measure should be the number of represented features rather than the size of the structuring element. Opening with adaptive element size can be done at two levels of adaptiveness:

1. global adaptation where one size is automatically chosen for the whole image;
2. local adaptation where size may vary with position in the image. In many images (e.g. due to perspective) global adaptation is insufficient.

6. Summary of demands

In adaptive smoothing we want:
- spatial scale invariance, achieved by taking the number of remaining objects (features) as a smoothing measure. Distinguishing objects or image features allows
- local adaptation, in which smoothing is tailored to objects. In order to get objects or features one needs
- segmentation into objects or other features;
- a measure of relative object (feature) importance.

Further we want
- symmetry in processing light and dark objects;
- greater speed than via repeated non-adaptive processing.

We shall now show how these demands can be fulfilled.

7. Multisize opening

Opening consists of two steps: size \( s \) erosion fol-
lowed by size $s$ dilation. By multisize opening we mean a set of opening operations with different values of $s$.

7.1. Multisize erosion by distance transformation

Adaptation may be done by 'trying all sizes and keeping the best'. We thus would like to do the opening for all sizes, if possible at once. Indeed, the first step in opening, erosion, can be done for all sizes at once through a 3-D distance transformation. Distance transformation fills an object with a field of values each of which represents the local distance to the object border. For a 3-D object like the landscape, the same can be done. Borgefors [6] has given an efficient algorithm for isotropic or pseudo-isotropic distance measures which yields a good approximation. Planes of constant distance $s$ coincide with the surfaces $S_z$ of the (size $s$) eroded landscapes, hence distance transformation can be seen as multisize erosion. The dilations remain to be done (Figure 3).

7.2. Non-spherical distance transformation

We propose to extend distance transformation to non-isotropic distance definitions. For distance transformation to be applicable it is sufficient that the structuring elements firstly be convex and secondly be scaled copies of each other forming a nested set.

Poweroids (if necessary with additive grey constant)) fulfil these conditions.

The recipe for the coefficient scheme of a distance transform (isotropic or not) is:

$$C_{sys} = \frac{z}{G}$$

with $x, y, z = 0, \pm 1, \pm 2, \ldots$ and where one finds $G$ by eliminating $r$ from:

$$G = g(r),$$

$$r = \frac{z}{\sqrt{x^2 + y^2}}.$$

**Example.** Paraboloids, nested around focus, focal distance $p$, are described by:

$$g(r) = \left(\frac{r}{s}\right)^2 + c$$

with $s = 2\sqrt{p}$ and $c = -p$.

In this case we find for $G$:

$$G = \frac{2pz(\pm \sqrt{x^2 + y^2 + z^2})}{x^2 + y^2}.$$

![Figure 3](image.png)

Figure 3. Multisize erosion by distance transformation. The original $S_z$ is eroded by spheres of different sizes to give the planes $S_{z1}, S_{z2}, S_{z3}$, and $S_{z4}$. The same planes directly result as equidistance planes from a single distance transformation.

![Figure 4](image.png)

Figure 4. The coefficient schemes of the standard Borgefors 2-D spherical distance transformation [6] and of the proposed 3-D parabolic distance transformation. In practical implementation coefficients are scaled up and approximated by integers.
Hence the coefficient recipe for paraboloids is:

\[ C_{xyz} = \frac{z}{G} = \frac{-z + \sqrt{x^2 + y^2 + z^2}}{2p}. \]

Similarly, for flat bottom cylinders or nested disks as structuring element, multisize erosion amounts to 2-D distance transformation in each grey value plane separately. In this case little is gained over repeated maximum filtering of the grey image. A more important drawback is that ripples on a slope are less effectively smoothed than those on a level area.

7.3. An alternative to the erosion step in opening

We shall show that in order to achieve opening, instead of erosion followed by dilation one may take the clipped medial axis transform followed by dilation.

Dilation can be visualized as adding a set of (spatially overlapping) spheres, or more general: copies of the structuring element. The spheres added in the dilation step of an opening operation constitute when 'orred', a layer \( L_2 \). The layers \( L_2 \) of all opening operations down to the smallest size can be orred to reconstruct the landscape. This is equivalent to orring all spheres of all sizes.

However, a subset of spheres is sufficient to reconstruct the landscape. E.g. spheres that are contained in a larger sphere can be left out, which yields the medial axis subset (MAS) of which the set of centers, with respective radii, is called the medial axis.

Conversely, dilating each medial axis center with its own radius yields the orred MAS.

Selecting a sub-subset allows partial reconstruction. E.g. selecting only large spheres from the MAS gives rough smoothing. More exactly: orring the MAS spheres of sizes \( s \geq s_0 \) is equivalent to opening the landscape with a sphere of size \( s_0 \).

**Proof.** Landscape opened with sphere of size \( s_0 \) is (cf. Figure 5)

\[ B(S_3(s_0)) = \bigcup \text{spheres}(S_2(s_0), s_0), \]

\( B(S) \) is the 3-D volume below surface \( S \), \( S \) is a function of size, spheres(\( S, s \)) is the set of the spheres centered on \( S \) with size \( s \), spheres(\( B, s \)) is the set of the spheres centered in \( B \) with size \( s \).

For all \( s > s_0 \) the smoothing is rougher: \( B(S_3(s)) \) is contained in \( B(S_2(s_0)) \).

We may thus write

\[ B(S_3(s_0)) = \bigcup \text{spheres}(S_2(s), s), \quad s \geq s_0. \]

The surface \( S_2(s) \) determines spheres that trace \( S_3 \), adding spheres(\( B(S_2(s)), s \)) does not change the result:

\[ B(S_3(s_0)) = \bigcup \text{spheres}(B(S_2(s)), s), \quad s \geq s_0. \]

Leaving out spheres contained in a larger sphere is allowed and expresses \( B(S_3(s_0)) \) as union of its medial axis subset:

\[ B(S_3(s_0)) = \bigcup \text{spheres}(B(S_2(s)), s), \quad s \geq s_0, \]

not contained in a sphere(\( B(S_2(s')), s' \), \( s' > s \).

On the other hand the orred MAS of the unsmoothed landscape is:

\[ \bigcup \text{spheres}(B(S_2(s)), s), \quad s \geq 0, \]

not contained in a sphere(\( B(S_2(s')), s' \), \( s' > s \).

Selecting the sub-subset with \( s \geq s_0 \) gives:

\[ \bigcup \text{spheres}(B(S_2(s)), s), \quad s \geq s_0, \]

not contained in a sphere(\( B(S_2(s')), s' \), \( s' > s \).

---

Figure 5. Opening \( S_1 \) by a sphere sized \( s_0 \) is: orring all spheres of size \( S_0 \) that fit under \( S_1 \), but also: all of size \( \geq s_0 \) and also: all of size \( \geq s_0 \) that are not contained in a bigger sphere that fits under \( S_1 \). These are centered on the medial axis, the set of local maxima of the distance transform. For opening it is sufficient to dilate each medial axis point by a sphere as big as its own distance value.
Indeed this equals the size \( s_0 \) opened landscape and its ordered medial axis subset. □

The medial axis of the size \( s_0 \) opened landscape is thus the medial axis of the original landscape where centers with radius less than \( s_0 \) have been removed: \( s_0 \) plays the role of a clipping level. Opening the landscape with a sphere of size \( s_0 \) is equivalent to dilating each medial center with its own radius if greater than \( s_0 \). Taking different clipping levels one can replace the multisize erosion step in multisize opening.

### 7.4. Medial axis from distance transform

In the previous section we have seen opening based on the medial axis. This axis too can easily be defined in terms of the distance transformation result. The medial axis is the set of local maxima of the discrete distance field, both in position and value [7]. Points at position \( p \) with value \( d \) with one or more neighbours at positions \( p_i \) with values \( d_i > d \) are included if sphere \((p,d)\) is not contained in any of the spheres \((p_i,d_i)\), i.e. for all \( d_i - d \) sufficiently small.

### 7.5. No multisize opening by distance transformation

In order to have multisize opening, we have to dilate each erosion result with its own size structuring element. The resulting \( S_3 \) surfaces may coincide. E.g. a blank image will give the same opening result for all sizes. Where \( S_3 \) surfaces coincide, it is sufficient to retain the \( S_3 \) surface that corresponds to the largest element size.

In the same way we might dilate from each medial axis point valued \( s \) with its own size \( s \) structuring element to give full reconstruction.

It would be attractive to perform multisize opening in a way similar to multisize erosion, by 3-D distance transformation. There each 3-D position was processed only once, as distance layer on distance layer is recursively deposited like in a growing crystal or tree. The calculation of the field of distance values proceeds as a wavefront from a transmitter. In the erosion step, the landscape \( S_1 \) is the transmitter generating distance layers \( S_2 \). In the second, dilation, step each distance layer \( S_2 \) must transmit its own wavefront and no further than its own limited range: its size. (They behave like TV transmitters covering a country; everywhere one asks what is the most distant transmitter to be received.) So at each 3-D point the different wavefronts must pass freely carrying the information of the shape of their range limit (such as the transmitter action radius). This seems to exclude a propagation process where each point has one scalar value processed once (cf. Figure 6).

So one-step multisize opening by distance transformation is impossible and our original goal of first smoothing at all sizes and then selecting the best result must be abandoned (note that this strategy would have been globally adaptive and that distinguishing, importance estimation and ordering of image features had not yet been taken care of).

However, in Section 8 we shall show that the medial axis embodies just the right information to distinguish objects (features), to estimate their importance and to perform selective partial reconstruction by dilation. This amounts to locally adaptive smoothing: only 'the upper ten' is - largely - reconstructed, the rest is wiped out. The reconstruction by dilation is done in one step by a variant of 3-D distance transformation. This will be described in Section 9.

![Figure 6](image)

Figure 6. Two erosion results \( S_2 \) and \( S_3 \) must each be dilated by their own size to give the opening results \( S_3 \) and \( S_2 \). The distance fronts emerging from \( A \) and \( B \) as drawn determine \( S_3 \) and \( S_2 \) above \( A \) and \( B \). Where they cross, each point belongs to two independent distance fronts. This excludes one-time processing of scalar valued points: multisize opening by distance transformation is impossible.
8. **Adaptive smoothing by selective dilation**

Still we have no ordering of detail as wanted, no smoothing measure (except for size). We earlier indicated the need to define image features, to attribute an importance value to them and to order them according to importance so as to allow tailored smoothing.

It would be desirable if the medial axis could be partitioned into segments that would correspond to image features. Their importance could be estimated from the summed distance values on the segment (or their summed squares), a measure for the integrated intensity of the corresponding feature. Ordering them would be no problem.

The task remains to partition the medial axis. The distance values on the medial axis change smoothly along the axis. Contrary to an arbitrary object, a landscape has a medial axis that is mainly connected although it may show small gaps. Thus no natural selection criterion is obvious. Still, clear landscape features such as peaks each have their own piece of medial axis roughly parallel to the grey axis; feature pieces are connected by other pieces less parallel.

*This allows us to break the continuous medial axis set into natural discrete parts that can be ordered.*

Partitioning is achieved as follows. Rather than the medial axis (the 3-D local maxima) we use the 2-D x-y local maxima of the distance field (like in 2-D skeletonization, [8]) to approximate the medial axis pieces parallel to the grey axis. The parallel pieces of the medial axis are labeled, their importance estimated by the integrated distance value, a measure for the integrated intensity below the corresponding feature. The pieces are ordered according to importance. Given a specified number N, the N most important pieces are selected to be reconstructed by dilation. Full reconstruction of main features is inappropriately accurate. Hence only a certain upper percentile of distances of each piece is really used in the dilation.

9. **Fast reconstruction from the medial axis parts**

Given the medial axis parts with associated size values, the goal of the dilation is straightforward.

![Figure 7. Fast reconstruction from the medial axis by a variant of the distance transform. The medial axis here consists of two points P_20 and P_8. From P_20 fronts of decreasing distance F_10 through F_8 are generated. Then P_8 is added to F_8 and fronts F_7 through F_0 are generated. F_0 is the reconstructed original. The same can be done for a medial axis segment, which - unlike this example - is connected.](image)

Around each pixel of the medial axis parts with a certain size value, a structuring element with that size has to be constructed. These structuring elements do not have to be spherical but may have any nested shape (see subsection 7.2).

First we give a simple illustration of the method (Figure 7) that will explain how the trap of subsection 7.5 can be avoided in this case. Suppose the medial axis segment consists of two points valued 20 and 8 respectively (in fact the segment is connected but that is irrelevant to the argument and would obscure the illustration; for the same reason spherical rather than parabolical distances have been drawn).

The variant of the distance transform introduced here starts at the largest medial axis value (20) and grows rings of decreasing distance (19, 18, etc.) around it, until the next lower medial axis value (8) is reached. Now the point with medial axis value 8 is simply added to the set of points with distance value 8. Then rings (7, 6, etc.) are grown around the point valued 8 and the ring valued 8. At some value (6) the rings may coalesce and from then on grow new rings (5, 4, etc.) together until the reconstruction is completed with ring 0.

At no instant, growing rings interfere in the sense of subsection 7.5, and indeed no multisize opening has been achieved: only the ring with distance value 0 has a meaning as an opening result.

Now the implementation will be described in more detail.
We have used a modified version of the uniform cost algorithm as described in [9] and [10]. The algorithm is based on the simultaneous generation of pixels with equal distance, distance being defined as the minimum distance to the border of the structuring elements to be constructed. The values on the medial axis are per definition these minimum distances.

For the algorithm we use a bucket structure. We have a number of buckets available. The algorithm fills these buckets subsequently. In bucket \( d \) all pixels with distance \( M - d \) will be stored. \( M \) is the maximum distance value occurring on the medial axis. A pixel can be stored in a bucket either because it is a medial axis pixel, or because the pixel is generated from a neighbour.

The algorithm starts by storing pixels with distance value \( M \) in bucket 0. Before starting, all medial axis pixels have been sorted according to their associated distance values and per part only an upper percentile has been included (say 90%). Then the pixels with distance values \( M - 1, M - 2, \ldots, M - d_1 \),

\[ \text{Figure 8. Three crosssections of the 3-D binary greyvalue landscape (umbra) of the 2-D electron micrograph image.} \]

\[ \text{Figure 9. A crosssection of the 3-D paraboloid distance transform applied to the landscape of Figure 8. Distance (intentionally modulo 16) is displayed as grey.} \]

\[ \text{Figure 10. The result of adaptive smoothing on the landscape of Figure 8 shows up in this crosssection as the dark corridor left by the dilate until meet operation on the (partially) reconstructed main features. The paraboloid shapes used in reconstruction are obvious.} \]

\[ \text{Figure 11. Another crosssection of the same result as in Figure 10.} \]
are stored, in which \( d_1 \) is the distance between two 6-connected neighbours as defined in the distance transform. In bucket \( M - d_1 \) the 6-connected neighbours of the pixels in bucket \( M \) are stored as well, if they are not already processed. Storing a pixel implies of course labeling the pixel, in order to create the dilated image. This procedure is repeated until bucket \( M - 1 \) is reached (all pixels with a distance greater than 1 have been processed).

Note that the pixels are directly accessed via their addresses, the dilation is not done in iterations. Secondly note that, when processing distance value \( d \), buckets with a number lower than \( d - d_{\text{max}} \) are no longer used. The number of buckets actually required is thus only \( d_{\text{max}} + 1 \) and not \( M \). The memory is moreover managed dynamically.

The combination of the results of the opening of the landscape and its complement is achieved almost identically. Now the two results of the previous dilation are grown further until they touch. The touching points define the new grey values of the 2-D pixels (when using paraboloids these touching points are unique for each \( x \)-\( y \) position).

10. Results

The method proposed has been tested on two images. One image is a 5-bit 128\(^2\) high SNR electron microscope image of gold particles embedded in glass (with mean grey value of objects 7, of background 24) to which uniform noise (range 0-7) was added (Figure 12.1). The other image is an 8 bit 128\(^2\) standard portrait (Figure 15.6) with little noise.

The electron micrograph gives a grey value landscape of which three crosssections are simultaneously shown in Figure 8. Figure 9 gives an impression of a cross section of the resulting distance transform. After selective reconstruction (here for 40 features) and the dilate until meet operation, cross sections can be displayed (Figures 10 and 11) in which the smooth result shows up as a dark curve between lighter reconstructed features.

The image processing capability of the method is demonstrated in Figures 12 and 13 to 15. Decreasingly heavy smoothing is achieved by increasing the number of features to be preserved. In order to fol-

low the development, Tables 1 and 2 give the list of features, designated 'light' or 'dark', their integrated distance value (used as importance measure), and their number of medial axis voxels.

For the electron micrograph the two most important features are light and constitute the background (Figure 12.2). Then the next six dark fea-

<table>
<thead>
<tr>
<th>Feature no.</th>
<th>Light/dark</th>
<th>No. of voxels</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L</td>
<td>7244</td>
<td>197139</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>5175</td>
<td>140258</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td>549</td>
<td>15488</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>655</td>
<td>14926</td>
</tr>
<tr>
<td>5</td>
<td>D</td>
<td>590</td>
<td>14413</td>
</tr>
<tr>
<td>6</td>
<td>D</td>
<td>598</td>
<td>14329</td>
</tr>
<tr>
<td>7</td>
<td>D</td>
<td>531</td>
<td>12872</td>
</tr>
<tr>
<td>8</td>
<td>D</td>
<td>320</td>
<td>12016</td>
</tr>
<tr>
<td>9</td>
<td>L</td>
<td>404</td>
<td>11048</td>
</tr>
<tr>
<td>10</td>
<td>L</td>
<td>316</td>
<td>10470</td>
</tr>
<tr>
<td>11</td>
<td>D</td>
<td>349</td>
<td>8430</td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>223</td>
<td>8069</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>368</td>
<td>7871</td>
</tr>
<tr>
<td>14</td>
<td>D</td>
<td>197</td>
<td>7348</td>
</tr>
<tr>
<td>15</td>
<td>D</td>
<td>292</td>
<td>6510</td>
</tr>
<tr>
<td>16</td>
<td>D</td>
<td>245</td>
<td>6017</td>
</tr>
<tr>
<td>17</td>
<td>D</td>
<td>217</td>
<td>5627</td>
</tr>
<tr>
<td>18</td>
<td>D</td>
<td>252</td>
<td>5564</td>
</tr>
<tr>
<td>19</td>
<td>L</td>
<td>291</td>
<td>5461</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>228</td>
<td>5191</td>
</tr>
<tr>
<td>21</td>
<td>D</td>
<td>244</td>
<td>5091</td>
</tr>
<tr>
<td>22</td>
<td>D</td>
<td>240</td>
<td>4755</td>
</tr>
<tr>
<td>23</td>
<td>D</td>
<td>212</td>
<td>4747</td>
</tr>
<tr>
<td>24</td>
<td>D</td>
<td>165</td>
<td>4721</td>
</tr>
<tr>
<td>25</td>
<td>D</td>
<td>216</td>
<td>4632</td>
</tr>
<tr>
<td>26</td>
<td>L</td>
<td>232</td>
<td>4520</td>
</tr>
<tr>
<td>27</td>
<td>D</td>
<td>229</td>
<td>4246</td>
</tr>
<tr>
<td>28</td>
<td>D</td>
<td>146</td>
<td>3934</td>
</tr>
<tr>
<td>29</td>
<td>D</td>
<td>136</td>
<td>3911</td>
</tr>
<tr>
<td>30</td>
<td>D</td>
<td>166</td>
<td>3792</td>
</tr>
<tr>
<td>31</td>
<td>D</td>
<td>57</td>
<td>3791</td>
</tr>
<tr>
<td>32</td>
<td>D</td>
<td>197</td>
<td>3711</td>
</tr>
<tr>
<td>33</td>
<td>D</td>
<td>163</td>
<td>3280</td>
</tr>
<tr>
<td>34</td>
<td>D</td>
<td>105</td>
<td>3200</td>
</tr>
<tr>
<td>35</td>
<td>L</td>
<td>219</td>
<td>3130</td>
</tr>
<tr>
<td>36</td>
<td>D</td>
<td>210</td>
<td>2593</td>
</tr>
<tr>
<td>37</td>
<td>L</td>
<td>105</td>
<td>2525</td>
</tr>
<tr>
<td>38</td>
<td>D</td>
<td>79</td>
<td>2509</td>
</tr>
<tr>
<td>39</td>
<td>D</td>
<td>55</td>
<td>2485</td>
</tr>
<tr>
<td>40</td>
<td>L</td>
<td>122</td>
<td>2455</td>
</tr>
</tbody>
</table>
tures build up two particles (Figure 12.3). Obviously one object may be broken up into several features. Five light features and fifteen dark ones compose the background and six particles of Figure 12.4. Fifty features give a largely adequate (Figure 12.5) and 662 a full reconstruction (Figure 12.6), which still shows inherent smoothing when compared to the original (Figure 12.1). This is due to the fact that the medial axes of the upper and lower part are not allowed to touch. One pixel thick peaks are therefore not reconstructed.

For an image of the micrograph type, a priori known to consist of dark objects on a light background, the method might be reduced to asymmetric processing of the dark parts (above the landscape) only. This will be studied further.

For the portrait, the subsequent stages of decreasing smoothing by reconstruction of an increasing number of features, show identifiable image parts emerging one by one. Descriptions of these parts have been listed in Table 2. It may seem counterintuitive that the third part, the face, which is light, already contains the dark eyes in quite detailed shape. The reason is that one important connected part, the light face, encloses dark parts, the eyes, as a complement that, although little important in its own right, is filled in and made visible by the dilate until meet process.

Also some cases of insufficient partitioning can be observed. The upper four blocks in the shawl appear at once (Figures 14.1, 14.2) and form one feature (no. 9). Also the lower four are one feature (Figures 13.2, 14.4, no. 11).

Again, like in the electron micrograph case, fifty features give a subjectively acceptable rendition (Figure 15.4). The upper left and right background

Figure 13. Adaptive smoothing of a portrait (Figure 15.6). $N$ is the specified number of features. 1. $N = 1$: left hair + shawl, 2. $N = 2$: right hair added, 3. $N = 3$: face added, 4. $N = 4$: right shawl added, 5. $N = 5$: left hat added, 6. $N = 7$: piece of shawl under chin added.

Figure 15. Adaptive smoothing of a portrait (Figure 15.6). $N$ is the specified number of features. 1. $N = 14$: no obvious change from $N = 13$. 2. $N = 17$: background left of shawl added, 3. $N = 20$: grey, left in shawl, added, 4. $N = 50$: mouth still blurred, corners (except lower left) absent, 5. $N = 768$: full reconstruction includes inherent smoothing (value 1 omitted from medial axis). 6. Original portrait.

<table>
<thead>
<tr>
<th>Feature no.</th>
<th>Light/dark</th>
<th>No. of voxels</th>
<th>Importance</th>
<th>Description</th>
<th>Feature no.</th>
<th>Light/dark</th>
<th>No. of voxels</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D</td>
<td>7240</td>
<td>314136</td>
<td>left hair + shawl</td>
<td>21</td>
<td>L</td>
<td>376</td>
<td>10579</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>6913</td>
<td>300733</td>
<td>right hair</td>
<td>22</td>
<td>L</td>
<td>297</td>
<td>9114</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>4440</td>
<td>110879</td>
<td>face</td>
<td>23</td>
<td>D</td>
<td>362</td>
<td>8940</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>6615</td>
<td>108165</td>
<td>right shawl</td>
<td>24</td>
<td>D</td>
<td>119</td>
<td>8640</td>
</tr>
<tr>
<td>5</td>
<td>L</td>
<td>2409</td>
<td>69364</td>
<td>left hat</td>
<td>25</td>
<td>D</td>
<td>623</td>
<td>8168</td>
</tr>
<tr>
<td>6</td>
<td>D</td>
<td>1017</td>
<td>65600</td>
<td>?</td>
<td>26</td>
<td>D</td>
<td>243</td>
<td>8014</td>
</tr>
<tr>
<td>7</td>
<td>D</td>
<td>1571</td>
<td>64480</td>
<td>piece of shawl under chin</td>
<td>27</td>
<td>D</td>
<td>134</td>
<td>7624</td>
</tr>
<tr>
<td>8</td>
<td>L</td>
<td>3454</td>
<td>44311</td>
<td>left shawl</td>
<td>28</td>
<td>D</td>
<td>72</td>
<td>7529</td>
</tr>
<tr>
<td>9</td>
<td>L</td>
<td>1771</td>
<td>34188</td>
<td>upper 4 blocks of shawl</td>
<td>29</td>
<td>L</td>
<td>314</td>
<td>7261</td>
</tr>
<tr>
<td>10</td>
<td>L</td>
<td>2697</td>
<td>29437</td>
<td>center seesaw of shawl</td>
<td>30</td>
<td>D</td>
<td>137</td>
<td>6918</td>
</tr>
<tr>
<td>11</td>
<td>L</td>
<td>1670</td>
<td>28472</td>
<td>lower 4 blocks of shawl</td>
<td>31</td>
<td>L</td>
<td>381</td>
<td>6486</td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>419</td>
<td>27400</td>
<td>?</td>
<td>32</td>
<td>D</td>
<td>1015</td>
<td>6480</td>
</tr>
<tr>
<td>13</td>
<td>L</td>
<td>855</td>
<td>24290</td>
<td>right hat</td>
<td>33</td>
<td>D</td>
<td>95</td>
<td>6440</td>
</tr>
<tr>
<td>14</td>
<td>L</td>
<td>617</td>
<td>21345</td>
<td>?</td>
<td>34</td>
<td>D</td>
<td>449</td>
<td>6087</td>
</tr>
<tr>
<td>15</td>
<td>D</td>
<td>957</td>
<td>17819</td>
<td>background left of shawl</td>
<td>35</td>
<td>L</td>
<td>426</td>
<td>5877</td>
</tr>
<tr>
<td>16</td>
<td>L</td>
<td>592</td>
<td>17659</td>
<td>grey, left in shawl</td>
<td>36</td>
<td>D</td>
<td>124</td>
<td>5749</td>
</tr>
<tr>
<td>17</td>
<td>D</td>
<td>339</td>
<td>17185</td>
<td></td>
<td>37</td>
<td>L</td>
<td>231</td>
<td>5747</td>
</tr>
<tr>
<td>18</td>
<td>D</td>
<td>227</td>
<td>16315</td>
<td></td>
<td>38</td>
<td>D</td>
<td>137</td>
<td>5374</td>
</tr>
<tr>
<td>19</td>
<td>D</td>
<td>183</td>
<td>11727</td>
<td></td>
<td>39</td>
<td>D</td>
<td>71</td>
<td>4855</td>
</tr>
<tr>
<td>20</td>
<td>D</td>
<td>294</td>
<td>11627</td>
<td></td>
<td>40</td>
<td>D</td>
<td>39</td>
<td>4350</td>
</tr>
</tbody>
</table>
corners and the lower right white area are not yet represented. Full reconstruction taking 768 features (Figure 15.5) still gives inherent smoothing when compared to the original (Figure 15.6).

The images given thus demonstrate the principle characteristics of the method. Necessary refinements are the topic of further study.

11. Conclusions

We have described a method for adaptive smoothing which is spatially scale invariant. Instead of filter size we take the number of remaining objects (features) as smoothing measure. Moreover, distinguishing objects or image features allows us to perform local adaptation, smoothing tailored to objects. Segmentation into objects or other features together with a measure of relative object (feature) importance is derived from the 3-D medial axis of the grey value landscape. The medial axis is distilled from the 3-D distance transform and segmented according to the 2-D x–y local maxima of the latter. The medial axis values of a segment are integrated to define its importance. Dilating most important 3-D medial axis segments yields adaptive opening of the landscape. In order to assure symmetry in processing light and dark objects, the landscape and its complement are processed concurrently. The total number of segments to be dilated in both openings can be specified. We combine both opening results by dilating them until they meet to give the smoothed end result. We expect the method to be extendible along the lines described in [4] to selectively smooth ramps or textures (noise).

Acknowledgements

The critical contributions of Lucas van Vliet are gratefully acknowledged.

References

P.W. Verbeek and B.J.H. Verwer

SHADING FROM SHAPE, THE EIKONAL EQUATION SOLVED BY GREY-WEIGHTED DISTANCE TRANSFORM


7

θαύμα ἴδεσθαι

(A wonder to be seen)

Homerus, Ilias
Shading from shape, the eikonal equation solved by grey-weighted distance transform

P.W. VERBEEK and B.J.H. VERWER

Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, Delft, The Netherlands

Received 26 March 1990

Abstract: For an optical or acoustical wavefront running through a medium of space variant refraction index the eikonal equation connects local front arrival time with local refraction index. So-called difference approximation methods are known for solving the spatial wavefront development with time and thus, indirectly, the eikonal equation.

Here a novel fast method for the calculation of an approximative solution of the eikonal equation is proposed.

From literature it is known that by solving an eikonal equation one can construct a line pattern rendition of a given image. We have generalized this method and made it fit for line engravings.

We have found yet another kind of image display based on solving an eikonal equation: shading from shape. We propose to construct a matte 3-D surface (shape) that, when illuminated perpendicularly and imaged in eye or camera, yields a grey value (shading, luminance) field that renders the image.

Both methods have been applied in a recent design for a Dutch coin.

Key words: Banknotes, coins, display, distance transform, eikonal equation, graphics, line pattern, shading, shape.

1. Introduction

Traditionally heads of coins and medals and line engravings of poststamps and banknotes are produced by hand. To design an algorithm that will perform this task is an obvious challenge. The method should give sufficient freedom to allow an artistic contribution. We were confronted with this problem when we were asked to take part in the design of a Dutch Dfl 50 coin.

An algorithm that produces coins based on a 3-D scan is easy to define, but presently range scanners are no standard equipment. This means that the subject to be imaged must come to the scanner, which limits the choice of subjects to the living. We tried to use an existing statue as an intermediate 3-D image stage, but optimal silhouette rendition is not as essential in sculpturing as it proved to be in coin design so we had to look for a different approach, based on photographs. In Section 2 we define two different methods for image display on a coin. They can be reduced to solving the same differential equation. In Section 3 the equation is identified as the eikonal equation, which pertains to a well-known problem in optics. In Section 4 the optics problem is identified with a well-known and solved image processing problem. In Section 5 we present a new efficient implementation of the solution technique, applicable in all three problem versions. Some aspects of the application to image display are discussed in Section 6. Section 7 shows results and Section 8 summarizes conclusions.

2. Display methods

An existing technique for the display of a 2-D grey image on a coin is rastering with a point grid (like newspaper photographs), or with one or more straight line grids. For poststamps computer generated variable-density equal-size dot patterns
have been used [1] which might also be applied to a coin.

For artistic reasons a novel technique was demanded, we shall present two.

2.1. First display method, surface generation, shading from shape

When illuminating a matte (Lambertian) surface from a direction $D$ the viewer experiences independently of the viewing direction a luminance $L(x, y)$ proportional to the cosine of the angle between $D$ and the normal $N$ on the surface [2].

On this basis we want to produce a 3-D surface of which the 2-D image matches a given 2-D image.

For a surface with altitude field $z(x, y)$ illuminated by a parallel beam along the $z$-axis the angle between $D$ and $N$ at a certain position equals the local slope angle $\alpha(x, y)$. The tangent of $\alpha(x, y)$ equals the modulus of the altitude gradient:

$$\text{arctan} |\text{grad}[z(x, y)]| = \arccos L(x, y).$$

This is a differential equation of the type

$$|\text{grad}(R)| = I$$

with input

$$I = \sqrt{\frac{1}{L(x, y)^2} - 1}$$

and unknown

$$R = z(x, y).$$

The problem now is to construct a surface $z(x, y)$ for a given field of altitude gradient moduli $I(x, y)$. This problem is underconstrained: in general more than one surface gives the same image. For instance under perpendicular illumination a surface will give the same image as its cast (negative).

The surface must be everywhere differentiable in order that the altitude gradient exists. If this condition is relaxed in the sense that a limited number of non-differentiable points are allowed then many more solutions are possible.

2.2. Second display method, line pattern display

Schroeder [3] has shown that the density of dark isohypses (lines of equal altitude of a landscape), when viewed from a distance or otherwise blurred, is interpreted by the human visual system as a negative luminance. On the other hand the density of isohypses is a measure for the modulus of the altitude gradient. Therefore we again have [3]:

$$|\text{grad}(R)| = I$$

now with input

$$I = \text{constant} - L(x, y)$$

and unknown

$$R = z(x, y).$$

For light isohypses on a dark background one must use input

$$I = L(x, y).$$

3. Relation to eikonal equation from optics and acoustics

As remarked by Schroeder, equation (1) is known in optics and acoustics as the eikonal equation that—for a wavefront running through a medium of space variant refraction index—connects local front arrival time $R$ (optical path length) with local refraction index $I$:

$$|\text{grad}(R)| = I.$$ (1)

The actual wave propagation is described by the wave equation [4]. The conditions under which the arrival time (or the optical path length or the isophase lines) is described by the eikonal equation are given in [5].

In the literature [4] so-called difference approximation methods have been mentioned by which the wave equation can be solved. These methods calculate phase and amplitude in the space-time domain time-layer by time-layer. From this arrival times (isophase) lines can be indirectly calculated.

4. Eikonal equation solved by grey-weighted distance transform

Already in 1968 Rutovitz [6] introduced the grey-weighted distance transform in which the
distance of a point to a set of reference points (the sources) is calculated as the lowest path sum of grey values (i.e. the lowest of the sums over all possible paths from that point to the sources).

Here we propose to apply the grey-weighted distance transform to the problem of wavefront arrival time, i.e. for the solution of the eikonal equation.

The argumentation goes as follows. The refractive index is proportional to the time it takes a wave to traverse a given short distance. This time may vary locally. The time needed to go from a source to a given point along a possible path is proportional to the path integral of the refractive index. It is well known that the wavefront arrival time in a point is given by the minimal time in which the wave can reach the point. That is the minimum path-integral. The situation is completely analogous to grey-weighted distance transformation: refractive index corresponds to grey value and wavefront arrival time to grey-weighted distance.

The grey-weighted distance transform thus yields for a given field of refractive indices the wavefront arrival time (optical path length), more generally speaking: the solution of the eikonal equation. Here we have

\[ |\text{grad}(R)| = I \]

with input

\[ I = \text{grey value} \]

and unknown

\[ R = \text{grey-weighted distance}. \]

5. Efficient implementation of the grey-weighted distance transform

The modern unweighted distance transform [7] can be implemented as the first (cost accumulation) stage of the uniform cost \( A^* \)-algorithm, where the cost function is defined by a set of fixed step sizes [8]. The set is chosen so as to yield approximate isotropy \((5,7,11)\) [7].

Similarly we propose to implement the modern grey-weighted distance transform [9] as the first stage of the uniform cost \( A^* \)-algorithm, where the cost function is now defined by the set \((5,7,11)\) multiplied by local grey values as a weight function.

The \( I \)-field is read only once (but in a specific, non-scanwise order) and the \( R \)-field is constructed simultaneously. Hence this version may also be important for optics and acoustics.

5.1. Description of the algorithm

The algorithm corresponds to the first (cost accumulation) stage of the uniform cost \( A^* \)-algorithm, where cost function is defined by local grey value times stepsize [8].

Variables used: grey value input \( I \), binary reference set (sources) \( B \), grey-weighted distance result \( R \), stepsizes \((5,7,11)\).

Set all \( R \) to \( \infty \), except positions indicated in \( B \).
Set \( R \) positions indicated in \( B \) to 0 (or to another value if desired).
While there are unprocessed points, take unprocessed point \( p \) with lowest value \( R_p \).
For all neighbours

Determine candidate value \( C_n = R_p + S_n I_n \)
of neighbour \( n \)
If \( C_n < R_n \) replace \( R_n \) by \( C_n \)
Label point \( p \) as ‘processed’
Endfor
Endwhile

5.2. Validation of the algorithm

Lemma. The distance result achieved by this algorithm satisfies the eikonal equation when the gradient is defined as the maximum difference with a lower neighbour, normalized by step size.

Proof. Consider point \( p \) with distance result value \( R_p \). This can be the result of candidateing from the lower neighbours \( n1, n2, ..., nk \). The candidate values then were

\[ C_{ni} = R_{ni} + S_{ni} I_p \quad \text{with } i = 1, 2, ..., k. \]

Of these the lowest has become the result:

\[ R_p = \min C_{ni} = C_{nm} \leq C_{ni} \quad \text{for } i = 1, 2, ..., k \]

and thus

683
\[ R_{nm} + S_{nm} I_p = C_{nm} \]
\[ C_{ni} = R_{ni} + S_{ni} I_p \quad \text{for } i = 1, 2, \ldots, k, \]
\[ \frac{C_{nm} - R_{ni}}{S_{ni}} \leq I_p \quad \text{for } i = 1, 2, \ldots, k \]

and
\[ \frac{C_{nm} - R_{nm}}{S_{nm}} = I_p. \]

Indeed, the gradient (defined as maximum difference with the lower neighbours, normalized by step size) now is seen to equal the input grey value:
\[ \max \left[ \frac{R_p - R_{ni}}{S_{ni}} \right] = \max \left[ \frac{C_{nm} - R_{ni}}{S_{ni}} \right] = I_p. \]

6. Image display aspects

6.1. The use of contours and/or edges in image display

Considering that grey-weighted distance transformation is equivalent to local 1-D integration, it is not surprising that it is the inverse of 'modulus gradient' which is equivalent to local 1-D differentiation.

The sources (reference set) play an important role: their positions become non-differentiable points in the result, their values act as integration constants. One is free to adapt the choice of sources to technical or artistic demand. This freedom is exploited here to stress contours and/or edges: a binarized contour/edge image is used as source image. Editing the source image is the main artistic freedom of the method. Also geometric figures such as a circular or rectangular frame can be added to the source image.

6.2. Shading from shape

With the algorithm of 5.1 a 3-D surface as proposed in 2.1 can be produced quickly and automatically. The surface can be implemented as a coin, a medal, a plaquette, etc. It is essential that the surface is matte; display contrast is low and competes with glare.

6.3. Folding the surface

Without proper measures the altitude of the surface can run into impracticable values. In such cases we propose to replace \( z = R \) by a triangular wave function
\[ z = \min(R \mod M, (M - R) \mod M) \]

where \( M/2 \) is the desired altitude range of the coin.

A collection of roofs results which stays within range while the slopes still meet the conditions imposed by the grey value input. The price paid is the increase in the number of non-differentiable points (roof tops).

6.4. Line pattern display

The method of imposed sources to stress contours and/or edges can also be used in line pattern display. Now the procedure to generate a surface (this time with unlimited altitude) from gradients and sources is applied on the basis of eq. (3) instead of eq. (2). Next the isohypses are determined and displayed as a binary line pattern. Schroeder's result [3] corresponds to ours in the case of an extremely simple source: one point in the image center. The line pattern can be used on coins, poststamps and banknotes.

7. Results

Two images have been selected for display: an en-profil portrait (Figure 3a) of the present queen of the Netherlands and a composite image (Figure 1a) of the en-profil portraits of the four queens that reigned between 1890 and 1990.

Two source images were made from Figure 3a: the contour mask (Figure 3b) and the contour + edges image (Figure 3c). One source image was made from Figure 1a: the contour + edges image (Figure 1c). The shading from shape display method of Section 2.1 applied to Figure 1a with source image Figure 1c yielded the surface shown in Figure 1d (where grey value stands for altitude). The line pattern display method of Section 2.2 applied to Figure 3a with source image Figure 3b yielded the surface shown in Figure 3d.
For three choices of maximum altitude $M$ the results of folding Figure 1d are shown in Figures 2a, 2b and 3c. Figure 2a has been selected for production (see a picture of the backside of the 50 guider coin in Figure 6b).

For three choices of isohypse spacing the line results based on Figure 3d are shown in Figures 4a, 4b and 4c. Figure 4a was selected for production originally, since the line density of Figure 4c was too high for technical realization. A first production proof showed an image we did not anticipate. The lines reflected different amounts of light dependent on their orientation. This ruined the image. We therefore chose an image with black lines on a white background, Figure 5a (which could be used for banknotes as well). A picture of the frontside of the coin is shown in Figure 6a, with a light source reflecting directly into the camera.

The isohypse display applied to Figure 3a with source image Figure 3c yields a surface of which the isohypses are shown in Figure 5b (same spacing as in Figure 4a). The edges can be seen to be more prominent.

The line pattern display applied to Figure 1a with source image Figure 1c yields a surface of which the isohypses are shown in Figure 2d. For acceptable contrast we had to choose line density in light areas already too high for technical realization and line density in dark areas so low that featureless open spaces occur.

The shading from shape display applied to Figure 3a with source image Figure 3c yields a surface of which the folds are shown in Figure 4d. The maximum altitude has been chosen unrealistically
large in order to assure sufficient contrast on long slopes.

Figure 5c shows the line pattern display applied to Figure 3a with the central point as the only source. The result is comparable to Schroeder's [3]. Figure 5d shows the line pattern display applied to Figure 1a with a circular frame as the only source. Although the same objections hold as for Figure 2d the four heads being built from shared sets of concentric closed rings offer an intriguing effect.

For banknote printing the constraint on line density may be relaxed so as to make Figures 2d, 4c and 5d acceptable.

8. Conclusions

- The grey-weighted distance transform yields an approximative solution of the eikonal equation \( |\text{grad}(R)| = I \).
- In the proposed implementation of the grey-weighted distance transform the I-field is read only once (but in a specific, non-scanwise order) and the R-field is constructed simultaneously.
- The proposed implementation allows specifying constraints of the R-field as integration constants.
- Image display is possible by choosing:

\[
I = \sqrt{\frac{1}{L(x, y)^2} - 1}.
\]
A matte surface $z = R$ under perpendicular illumination will then show a luminance distribution that corresponds to the image to be displayed.

This property could be used to verify genuinity. The property could also be used for orienting the surface perpendicularly to the light source or vice versa.

It is essential that the surface is matte; display contrast is low and competes with glare. If the illumination is not perpendicular the surface shows a luminance distribution with artistic rather than technical value.

• Image features (like edges or contours) and other geometric figures can be introduced as constraints. The option of editing adds to the artistic value of this imaging technique.

• In order to limit the surface altitude range to $M/2$ the altitude $z = R$ can be replaced without loss of the display property by a triangular function $z = \min(R \mod M, (M - R) \mod M)$.

• The freedom of constraint choice decisively improves artistic controlability of the line pattern display proposed by Schroeder.
Figure 4. Results for the queen image: (a) line pattern display, selected version; (b) enlarged spacing; (c) reduced spacing; (d) folded surface display (from Figures 3a and 3c, cf. 2.1).
Figure 5. Various isohypse displays: (a) dark lines based on negative of Figure 3a and on Figure 3b; (b) from Figures 3a and 3c; (c) from Figure 3a and point source; (d) from Figure 1a and circular frame.

Figure 6. A picture of the 50 guilder coin with direct light in the camera: (a) frontview; (b) backview.
Acknowledgement

The artistic patronage, continued interest and stimulating remarks of P. Stryck en are gratefully acknowledged. Delt Spline Systems are thanked for the supple cooperation in producing the perspex masters by numerically controlled milling. The Dutch State Mint is thanked for valuable feedback and technical support in the realization of the coin.

References

In the paper the relevance of the eikonal equation for seismics was mentioned. With a few simple examples I will illustrate the applicability of the grey-weighted distance transform to forward modelling (Kelly et al., 1982). In forward modelling one tries to calculate first arrival times, given a point source in and a velocity model of the earth. Using the inverse of the velocity, the slowness, as weights in the grey-weighted distance transform, solves that problem (see also Moser, 1989 and Saito 1989).

Figure 1. Average slowness between two points = \[ \frac{0.5a+b+c+d+e+f+g+h+0.5i}{8} \]
where a...i are the local slownesses

The standard way to calculate first arrival times is ray tracing. As Huygens (1690) has shown, rays are locally perpendicular to wavefronts. The grey-weighted distance transform calculates the wavefront. The advantage of the latter is that for all points on the surface an arrival time is found, whereas in ray tracing certain positions are difficult to reach. The disadvantage is the amount of computation involved. Because of the high accuracy required, large neighborhoods are necessary (17*17 in 2-d to have a maximum error below 10^-4, see chapter 3).

Large neighborhoods present another problem. When a vector is used which spans several pixels, the slowness in these pixels is not necessarily equal. To allow for all possible directions in a 17*17 neighborhood one would have to oversample 8 times (this is not the first time oversampling is required for better measurements, see also Young, 1983 and Verbeek, 1985). An ad hoc approximation is to use the average slowness of the pixels traversed in the transition (see figure 1). This is an approximation since in the pixels traversed only one direction is now allowed.

A first, simple example, assumes a uniform velocity in the earth. Using a point source below the earth results in hyperbolic arrival times at the surface. Figure 2 plots the relative error as a function of the angle \( \alpha \) between the normal on the surface and the straight line from source to destination. The error is calculated with respect to the Euclidean distance between start and surface.

The maximum error, as predicted in chapter 3, is 9.67 \( 10^{-4} \).

Figure 2. Maximum relative error as a function of angle between normal on surface and line connecting source and goal.

The second example uses a gradient velocity field. This a common situation because velocity normally increases with depth. Using a gradient from 1500 m/s at \( z=0 \) m to 6500 m/s at \( z=5000 \) m, with a sampling density of .2 pixels/m, leads to a measured arrival time of 1464.92 ms and a predicted arrival time of 1466.34 (In 65/15). The error is then 9.68 \( 10^{-4} \), negligibly more than the maximum error in the uniform case. The remainder is due to the numerical integration and is estimated to be one third of the difference between the outcome of the integration using \( n \) steps and the outcome using 2n steps (trapezium rule).

That estimate is plotted in figure 3 along with the measured error (since in our case we know the function, we know the real error). The estimate is seen to describe the real error well. It can also be noted that in this case 64 steps would suffice to reach the required accuracy of \( 10^{-4} \).

The third and final test concerns a fault. The fault makes an angle \( \beta \) with the perpendicular bisector of the line connecting source to goal. Above the fault the velocity \( v_a \) is 5000 m/s, below the fault the velocity \( v_b \) is 2500 m/s. The field was digitized on a 5x5 meter grid (slowness below therefore 1 ms/5m, above 2 ms/5m, slowness on the lines interpolated). Figure 4 shows the error as a function of \( \beta \). Two cases are shown. In case a source and
and fault are rotated with respect to the grid (figure 5, rhs). Now the fault is perpendicular to the vertical. In both cases the maximum error, as expected, is below $10^{-4}$: the error is now a function of the simple errors of the two principal directions the path is built from.

The errors are calculated with respect to the theoretical value found by solving Snellius' equation:

$$\frac{\sin(\beta - \arctan(\frac{x}{d-x\cdot\tan(\beta)}))}{\sin(\beta + \arctan(\frac{x}{d+x\cdot\tan(\beta)}))} = \frac{1}{2}$$

from which the travel time is trivial to deduce:

$$t = \frac{\sqrt{x^2 + (d+x\cdot\tan\beta)^2}}{v_a} + \frac{\sqrt{x^2 + (d-x\cdot\tan\beta)^2}}{v_b}$$

These three examples have shown that, in principle, the grey-weighted distance transform can replace ray-tracing. What remains is to study the performance on real world problems (e.g. figure 6).

To calculate the arrival times for a 400*800 image with a 17*17 neighborhood (176 neighbors) takes ±260s calculation time on an Ardent 3000. Apart from the expansion phase, the algorithm is not trivially vectorizable or parallelizable. Better suited for vectorization is the scan-wise distance transform (Dorst, Verbeek 1986), although that algorithm introduces unnecessary calculations (see chapter 2).

![Figure 3. Error due to the numerical integration as a function of the number of steps.](image)

![Figure 4. Relative error in arrival time as function of \( \beta \).](image)
Figure 5. (lhs) A simple deflection experiment. A fault makes an angle $\beta$ with respect to the perpendicular bisection of the line connecting start and goal. The ray - deduced from the first arrival times, consisting of points - from 400 m to the point at the same lateral position is shown.(rhs) Source, goal and fault rotated with respect to the grid as a control experiment. The distance between source and goal is equal to the depth in (lhs).

Figure 6. A real world example: velocity model, top, first arrival times, bottom.
B.J.H. Verwer

HEURISTIC SEARCH IN ROBOT CONFIGURATION SPACE USING VARIABLE METRIC

Proceedings 3rd Conference on Artificial Intelligence for Space Applications,
Huntsville, Alabama, November 1-2, 1987
NASA Conference Publication 2492, 153-154

In order to judge what was a 'reasonable distance', in a 'reasonable direction', I found myself entirely dependent on external, or visual, landmarks - marks on the floor, or marks triangulated with reference to the furniture and walls. I had to work out each step fully, and in advance, and then advance the leg, cautiously, empirically, until it had reached the point I had calculated and designated as secure.

*Oliver Sacks, A Leg to Stand On*
HEURISTIC SEARCH IN ROBOT CONFIGURATION SPACE USING VARIABLE METRIC

Ben J.H. Verwer
Faculty of Applied Physics
Delft University of Technology
The Netherlands

Intelligent path planning methods are of utmost importance for robots operating in space. In unexpected situations, reliable solutions must be generated. Trial and error may not suffice, nor does active human involvement. An autonomous robot path planning method is not yet available. Obstacles are, if at all, detected but not avoided.

We propose a method to generate obstacle free trajectories for both mobile robots and linked robots. The approach generates shortest paths in a configuration space. The metric in the configuration space can be adjusted to obtain a tradeoff between safety and velocity by imposing extra costs (= distance) on paths near obstacles.

A configuration space is a space in which each point corresponds to a unique position and shape of the robot in real space. The number of dimensions of the configuration space equals the degrees of freedom of the robot. E.g., a mobile vacuum cleaner has 3 degrees of freedom (2 translational and 1 rotational), an industrial manipulator with 6 rotational joints has 6 degrees of freedom. A survey on configuration spaces was presented by Lozano-Perez in [1]. A point in configuration space may be allowed or forbidden; forbidden if the robot in real space would collide with an obstacle or with itself, else allowed.

In [2] it was suggested to quantize the configuration space and to obtain a shortest path using a constrained distance transformation. Distance waves were propagated until convergence occurred, a rather time-consuming procedure.

We now propose to use a standard heuristic search method in the configuration space: the A*-algorithm [3]. From a given start point, costs are propagated until the goal-point is reached. A suitable heuristic is the length of the shortest path to the goal, assumed that obstacles are not present.

Metrics which can be used are approximations of the euclidean metric (like the chamfer metric, see Borgerfors [4]) or non-homogeneous metrics. We propose to use the latter, to be built of chamfer distances, multiplied by a factor dependent on the distance to the obstacles. The distances to the obstacles can be calculated by a standard distance transformation. The minimum distance in the real space from the robot to the obstacles is decisive for the metric in the configuration space.

In the surroundings of obstacles local distances are larger. The robot will avoid these areas whenever possible. During execution of the path the robot is given a fixed velocity in the configuration space. The local metric provides a slow speed near obstacles and a high speed away from obstacles. If the goal and start are defined as "obstacles" in the determination of the metric an acceleration from the start and a deceleration towards the goal is obtained simultaneously.

Note that our approach differs from the penalty approach [5]. In the penalty approach the finding of a path can not be guaranteed, because only local information is used. With the A*-algorithm however, the shortest and safest path is always found.
References:


EPILOGUE TO CHAPTER 8

Due to an unexpected premature deadline, only the abstract of the paper was printed in the proceedings of the conference. On the next pages the full paper - as originally submitted - is printed.
HEURISTIC SEARCH IN ROBOT CONFIGURATION SPACE USING VARIABLE METRIC

Ben J.H. Verwer
Faculty of Applied Physics
Delft University of Technology
The Netherlands

SUMMARY

A method is proposed to generate collision free paths for mobile robots and for manipulator robots. The method finds paths in a digitized configuration space using the A*-algorithm. The metric in the configuration space is adjusted to obtain a tradeoff between safety and velocity. Simulations are shown for three-dimensional configuration spaces.

1. INTRODUCTION

Robot path planning can be divided in two levels, as pointed out by Lumelsky [6]. The lower level deals with motion along predetermined trajectories. Topics of this level have been investigated extensively. The upper level concerns itself with the autonomous planning of trajectories based on complete or partial knowledge of the environment.

Intelligent path planning methods are of utmost importance for robots operating in space. In unexpected situations, reliable solutions must be generated. Active human involvement may not be possible.

Existing methods for the autonomous planning of trajectories are trial and error methods, methods based on penalty functions or free space methods [3]. Only free space methods guarantee that a trajectory is found if any trajectory exists. Our approach is a free space method (references to other free space methods can be found in [5]). It generates obstacle free trajectories for both mobile and manipulator robots. The trajectories are shortest paths in a quantized configuration space and found by heuristic search. In section 2 the configuration space idea is explained and in section 3 the heuristic search algorithm used. Section 4 treats the consequences and possibilities of the choice of metric in the configuration space. Throughout the paper simulation results are shown.

2. CONFIGURATION SPACE

Lozano-Perez [5] defines a configuration of an object as any set of parameters that completely specifies the position of any point of the object. The configuration space is then the space consisting of all configurations of an object (e.g. the configuration space of a vacuum cleaner is three dimensional, since it has two rotational degrees of freedom and one translational; the configuration space of a industrial robot with 6 joints is 6-D). The obstacles in the real world may be transformed into forbidden regions in the configuration space. The complement of the forbidden regions constitutes the free space of an object.

The transformation of the real world obstacles to configuration space obstacles can be sped up if the obstacles can be predicted. More on transformation can be a.o. be found in [7].
An example of a configuration space is shown in figure 1. It is a part of the configuration space of a robot manipulator with 3 joints. The forbidden regions are due to obstacles which will be shown in the next section. The figure is only indicative of the complexity that configuration spaces may have.

![Configuration Space Diagram]

Figure 1. Part of the configuration space of the problem to be presented in the next section (figure 3-5). The 3-D space is 31*57*57 large (calculated by constraints on minimum and maximum angles and by the resolution specified) and shown in slices. In each slice the angle of the 1st link with respect to a base is shown horizontally and the angle of the 2nd with respect to the 1st link vertically. Slices differ in the angle of the 3rd link with respect to the 2nd link.

3. HEURISTIC SEARCH

In [7] it was suggested to quantize the configuration space and to obtain a shortest path by constrained distance transformation: given a configuration space with attached to each point the distance to the goal point the shortest path to the goal point can easily be found by following the steepest gradient downward. The constrained distance transformation is a technique from image processing. The image, casu quo the configuration space is processed iteratively until convergence occurs. This is rather time consuming.

We now propose to use a standard heuristic search technique to find a path from the start point to the goal point: the A*-algorithm [4]. The A*-algorithm finds paths in graphs. The algorithm expands nodes from the start by selecting that node which is connected to an already expanded node, which is the most promising for lying on the optimal path. A path is called optimal if the sum of the costs of the arcs which connect the start node to the goal node is minimal among all possible paths.

We will discuss the costs attached to the arcs in the next section. Two simulations are shown in figures 2-5. Figure 2 shows a moving object with three degrees of freedom (2 translational degrees of freedom and 1 rotational degree of freedom). The start point in the configuration space represented a position in the lower left room with zero rotation, the goal point represented a position in the lower right room with zero rotation. Notice that periodic boundary conditions were applied (an object rotated over 360 degrees is in the same configuration as a non-rotated object at the same position).
Figure 2. An U-shaped object has to move from the lower left room to the lower right room. The pictures should be read top-down left to right. Adjacent path points in the configuration space are shown as a certain position and orientation of the object in the original obstacle scene.

Figure 3-5 show a 3 joint robot moving in a plane. The configuration space was shown in figure 1. The start and goal point were defined as to find a path with the end-effector in the 1st compartment (numbering from top to bottom) via the 2nd and 3rd compartments into the 4th compartment. Notice in figure 4 that the robot goes to the 4th compartment before entering the 3rd compartment in order to rotate its largest link to the right position without colliding with the compartment boundaries.

Figure 3. A robot moves its end-effector from the 1st compartment to the 2nd compartment.
Figure 4. The robot moves its end-effector from the 2nd compartment to the 3rd compartment.

Figure 5. Finally, the robot moves its end-effector from the 3rd compartment to the 4th compartment.
4. METRIC

The configuration space is digitized by a square grid. Each sample point is connected to neighbours. Neighbours of a point are those points whose coordinates differ at most one in each grid direction from that point. The costs attached to the transitions between points are decisive for the metric in the configuration space. The costs may be chosen position or direction dependent.

If the shortest distance in configuration space is desired a discrete approximation of the Euclidean distance is best (see examples in sections 3). Borgefors [1] has proposed chamfer-distances to obtain good approximations of Euclidean distances. Chamfer distances take into account that diagonal neighbours lie further apart than horizontal or vertical neighbours. In two dimension a good integer approximations is the pair (5,7) as distances to respectively the horizontal/vertical and the diagonal neighbours. The mean square error is 1% for this pair as opposed to about 10% for the conventional city- or chessboard distances used on square grids [2].

We propose to modify the metric in the configuration space. The reason is shown in a simple example in figure 6, where a path is shown between two points. Because the path is optimized by distance, the path found lies very close to the obstacles. In applications this will certainly present problems.

![Figure 6](image)

Figure 6. A path between two points lies close to the obstacles because the distance is minimized.

A position dependent metric solves the problem. The chamfer distances are adjusted dependent on the distance to the obstacles. These distances are calculated for each point by a normal distance transformation (for mobile robots and robot manipulators the minimum distance to the obstacles has to be taken). For the obstacles of figure 6 the distance image is shown in figure 7.
The metric in the configuration space becomes reciprocal to the real world distance to the obstacles. In the surroundings of obstacles the local metric is large. The robot will tend to avoid these areas (see figure 8). During execution of the trajectory the robot is given a fixed velocity in the configuration space. The local variable metric provides a low speed near obstacles and a high speed away from obstacles.

5. CONCLUSION

Heuristic search in the configuration space of a robot is a powerful tool for solving the findpath problem. The metric is adjustable to obtain a tradeoff between safety and velocity. Future research will concentrate on the theoretical description of the metric and on hardware implementability. This is necessary because the number of calculations increases significantly if the dimensionality of the problem increases.
REFERENCES


B.J.H. Verwer

A MULTiresolution Work Space, Multiresolution Configuration Space Approach to Solve the Path Planning Problem


9

(Europea òlon ħn ħkástou toû āntherôπou to îidôs strôgômîon, nótoûn kai plêuvraîs kûklîf exon, xêîrapa de têtparasses éîxe, kai skêleî tâ îsâ xaîs xepôîn, kai prôsòpta dû' ēk' aîxhêîn kûklôterêî, ómôia pântî.

(...)

nûn mêv ār aîntouîs, ēfî, diatêmîô dîkha ħkástou, kai òmà mêv âsônevêîteroi ēsônîaî, òmà de xhripermôteroi ëmîn daî tî plêious tôn ârîbêmûn xehôînmaî. kai bâdîôunîaî ðrôîoi ëpî ðuoîn skêloûîn. ëan ð' élî doîkîsín âsêglâînîaîn kai ùî 'thêlôsin ëskhîan ëghêîn, pâliûn aûî, êfî, têmîô dîkha, ðôsî' êfî 'ênûs pòrepûsônîaî skêloûîs âskîlîâîzontes.

(Furthermore, each person looked completely spherical, with the back and ribs all-around, and had four hands; and the same number of legs as a dog; and two faces on the round neck, equal in all respects.

(...)

I will cut each of them in two, he said, and they will be weaker and more valuable at the same time, because there will be more of them. They will walk straight on two legs. And if they turn out to be presumptuous still and will not be quiet, then I'll split them again, he said, so they will walk on one foot, running a sack race.)

Plato, Symposium
A MULTIRESOLUTION WORK SPACE,
MULTIRESOLUTION CONFIGURATION SPACE APPROACH
TO SOLVE THE PATH PLANNING PROBLEM

Ben J.H. Verwer
Pattern Recognition Group
Faculty of Applied Physics
Delft University of Technology
Lorentzweg 1, 2628 CJ Delft, the Netherlands

ABSTRACT

The algorithm SCOUT is proposed to solve the path planning problem for robots moving between stationary obstacles. The configuration space of a robot is represented as a binary tree of blocks, each of which can be efficiently tested for collisions using a hierarchy of bubbles in work space. The configuration space is explored independently from start and goal until a contiguous path of free blocks connecting the two is found. The algorithm finds a first path quickly, then improves on the length of the path by testing more parts of the configuration space and finally converges to the optimal path for any imposed metric. The tradeoff between the length of the path and time/memory usage is explicit and can be used to the advantage of integrated systems. The method is applicable to mobile robots as well as to manipulators. Results are shown for a 2 degrees of freedom planar arm and a 5 degrees of freedom industrial manipulator.

1. INTRODUCTION

Path planning for robots has received a lot of attention in the last decade. Satisfying practical solutions do not yet exist. The output of a path planning algorithm should be a collision free path between a start position and a goal position given the geometrical descriptions of the robot and the obstacles in the work space. In this paper we will present a practical solution assuming stationary obstacles. Sections 2 and 3 treat the path planner in configuration space. Sections 4 and 5 describe the collision detector in work space. We have named the algorithm SCOUT (Simple Calculation Of Useful Tracks).

2. OBSTACLES IN CONFIGURATION SPACE

The configuration space or C-space of a robot has proved to be a fruitful concept in robot path planning \[1\] [2]. It is defined as the set of all possible configurations of a robot. C-space is N-dimensional where N is the number of degrees of freedoms of the robot. Each point in it uniquely represents one position of the robot, e.g. the position and orientation of a mobile robot or the joint angles of an industrial manipulator.

We will denote the real world in which the robot exists and in which it has to perform its task the work space or W-space. W-space is normally 3-dimensional.

Some properties of the two spaces:

- A point in C-space is labelled 'prohibited' if the robot in that configuration would collide with any of the obstacles in W-space. Else the configuration is labelled 'free'.
- Each obstacle in W-space maps to one or more regions of prohibited configurations in C-space. The complement of these regions constitute the free part of C-space.
- Finding a collision free path in W-space is equivalent to finding a set of contiguous configurations in the free part of C-space between the start and goal configuration (actually more than one start and one goal configuration are possible, but for sake of clarity we will speak of the start and the goal configuration).

Since a direct calculation of prohibited regions in C-space is computationally expensive, we propose to explore C-space on the fly. In that manner only relevant parts need to be investigated.

3. PATH PLANNING WITH BLOCKS

Our path finding algorithm consists of two parts. A path planner which operates in C-space and a collision detector which operates in W-space. The collision detector will be described in section 4. It can test a set of configurations in W-space taking into account the geometry of the robot and the geometry of the obstacles. Efficiency dictates a rather crude, but safe test. The test has two possible outcomes: 'free' or 'uncertain'. The larger the set the more conservative the outcome. Thus if a large set results in a 'uncertain' verdict, one can keep trying with its subsets. But we will first describe the path planner.

The path planner finds a contiguous path of free configurations between the start and goal configuration in C-space. It starts by locating two free regions: one containing the start and one containing the goal. It then explores C-space until a connection between the two is found. In the exploration phase regions close to obstacles are neglected. In order to prevent a long search for the goal if the goal is close to an obstacle, the space is explored simultaneously from start and goal. Once a path has been found the search continues for more optimal paths.

The basic entity used by the path planner is a block. A block is a rectangular set of configurations in C-space. It is either 'free' or 'uncertain'; it is free if all configurations included in the block are free, else it is uncertain. When a block is uncertain it is divided in two halves. The direction of the split depends on the type of robot and the cause of the collisions in W-space (e.g. if the second link of a manipulator hits an obstacle it is unnecessary to split a block in any other dimension than the first or second).

The path planner uses 4 lists:
- F_s: Free blocks connected to the start.
- F_g: Free blocks connected to the goal.
- I_s: Interesting uncertain blocks adjacent to F_s.
- I_g: Interesting uncertain blocks adjacent to F_g.

N-dimensional blocks A and C are adjacent (or neighbours) if they touch in a hyperplane of dimension N-1. Block A is adjacent to a set of blocks if A is adjacent to at least one block in the set. Blocks A and C are connected if there exists a set of blocks B_i, i=1..n, with A adjacent to B_1, B_{i-1} adjacent to B_i, i=2..n-1, B_n adjacent to C. This guarantees that a path specified by a list of adjacent free blocks is collision free.
The path planner starts by dividing C-space until two free blocks have been detected: one containing the start and one containing the goal. These blocks constitute initial F_S and F_G. The blocks adjacent to the start and goal blocks constitute initial l_S and I_G. Since we will be splitting interesting blocks sequentially, an ordering has to be introduced. We propose to order blocks by size (other heuristics are applicable as well, e.g. goal/start directed):

Then a loop is entered (figure 1):
1. Split the largest block from I_S and the largest from I_G. Send the halves to the collision detector for evaluation.
2. If a half is free and adjacent to F_S add it to F_S, if free and adjacent to F_G add it to F_G, if free and adjacent to both F_S and F_G stop (call this mutual block M).
3. If a half is uncertain and adjacent to F_S add it to I_S, if uncertain and adjacent to F_G add it to I_G.

![Figure 1](image)

Figure 1 (a) The largest interesting block from I is selected for splitting (works for l_S and l_G) (b) Both halves free, both added to F, plus the free block which was not yet connected to the start or goal (c) Both halves not free, left added to I, right not interesting (d) Left half free, added to F, right half uncertain, added to I (e) Left half not free, added to I, right half free but not yet connected to start or goal.

Once a breakthrough has been obtained the path from the mutual block to the goal G is found using the distances d(B,G) of the free blocks B in F_S to G [3][4]. The distance d(B,G) is calculated during the search process from the distance of B's neighbours A to G:

\[ d(B,G) = \min \{ d(A,B) + d(A,G) \} \quad (d(G,G) = 0) \]

The optimal path from M to G given the set of free blocks follows from steepest gradient following through that set (the same procedure can be applied to find the optimal path, backwardly found though, from the start to M):

1. Optimal path P is (M), M is the last block L.
2. Select the block A adjacent to L for which the gradient \( \frac{d(L,G) - d(A,G)}{d(A,L)} \) is maximal (a trivial prove shows that the maximal gradient is always 1, so we stop if we encounter a gradient of 1).
3. Add A to the path P.
4. Now A is the last block L.
5. If A is the goal block stop else goto 2.

Different optimization criteria demand different metrics [5]. We have chosen to minimize joint rotation, which is achieved by the Euclidean metric in C-space. Whereas the errors caused by a uniform discretization have been studied extensively [6], the errors caused by a non-uniform discretization remain to be studied. Other useful metrics minimize the number of blocks traversed or the distance covered by the tool center point in W-space.

Once the path planner has found a path, the search for better paths starts. The path planner will neglect interesting blocks at a distance to start or goal greater than the length of the best path found so far (every new free block found can of course necessitate updating of distances). The search stops when all remaining interesting blocks are split to the finest possible resolution. Given the metric and the resolution used, the final path is the optimal path. A proof for this statement goes beyond the scope of this paper.

The output of the path planner, a list of adjacent free blocks connecting start to goal, is send to a trajectory planner. A trajectory planner can take into account dynamical aspects or send the robot through a series of setpoints located in the midst of the surfaces connecting adjacent blocks (a safe trajectory) [7]. We will not discuss trajectory planners in this paper.

**4. COLLISION DETECTION WITH BUBBLE HIERARCHIES**

To test if a robot in a given configuration collides with a number of obstacles is a non-trivial task [8]. To test if a robot collides for a set of configurations even more so. We propose a practical solution for the latter problem with low computational costs. It uses a datastructure called a bubble hierarchy. Each bubble in this datastructure encloses a part of the robot. If we use circular or spherical bubbles a simple computation suffices to find out if two circles or two spheres intersect. Bubbles can burst upon collision and expose other bubbles which among them enclose the same part of the robot more tightly.

Assume a planar work space with a small mobile robot and some complex obstacle. In most cases a quick test suffices to assert that the robot in a given configuration does not collide with the obstacle. If the bubble surrounding the robot (radius r_b) does not intersect the bubble surrounding the obstacle (radius r_o) the robot will not collide with the obstacle (figure 2):

\[ r_b + r_o < d \rightarrow \text{no collision} \]

where d is the distance between the bubbles. If they do intersect the robot is enclosed by two bubbles and test 1 tried again. This process continues up to a necessary resolution. The approximation of the obstacle is improved simultaneously.

![Figure 2](image)

The bubble enclosing the object does not intersect the bubble enclosing the obstacle: the configuration is "free"
Figure 3. The bubble enclosing the robot intersects the bubble enclosing the obstacle: the conclusion "free" cannot be drawn.

Figure 4. The previous bubble has burst. The configuration is "free".

It is possible to check a set of configurations of a robot. For that purpose we define a sweep bubble \([9]\). A sweep bubble encloses a robot bubble for all configurations in the block. If test (1) for the sweep bubble succeeds, we can guarantee the collision free movement of the robot in the block.

Suppose the center of a bubble enclosing a robot is constrained to stay within a block sized \(dx\) by \(dy\) (no rotation). The center of the sweep bubble in that case lies in the center of the block and the radius \(r_s\) of the sweep bubble is the radius of the robot bubble plus half the diagonal of the block:

\[
r_s = r_f + \sqrt{dx^2 + dy^2}
\]

(2)

For a robot consisting of a number of ordered links a similar argument as for a mobile robot holds. Suppose we want to test a block of configurations of a 2 d.o.f. robot, where the first joint angle may vary between \(\alpha\) and \(\alpha + d\alpha\) and the second joint angle between \(\beta\) and \(\beta + d\beta\). A bubble representing a part of the first link is swept from \(\alpha\) to \(\alpha + d\alpha\) \((d\alpha < \pi)\) and can easily be enclosed in a sweep bubble:

\[
b_2 = \frac{b_1 + b_2}{2},
\]

\[
r_s = r_b + \frac{|b_1 - b_2|}{2}
\]

(3)

where \(b_1\), \(b_2\), and \(b_3\) are the centers of the sweep bubble and the bubbles at joint angles \(\alpha\) and \(\alpha + d\alpha\) respectively and where \(r_f\) and \(r_b\) are the radii of the sweep bubble and the bubble being swept.

Sweep bubbles for other links are calculated by the same procedure (figure 5). The rotation invariance of bubbles facilitates the calculations. Only the centers have to be rotated over the different joint ranges. In fact that is the only reason for using bubbles instead of e.g. boxes aligned with the coordinate system.

When a sweep bubble intersects an obstacle bubble (possible collision) one of two actions can be taken. Either we burst the robot bubbles or we divide the block in two smaller blocks. Note that it is not useful to split the joint range of joint \(n\) if a bubble representing a part of any of the links 1 to \(n-1\) collides with an obstacle. This is an important observation: it is useless to improve the resolution of the gripper when the upper arm might hit an obstacle! We propose to split the largest of the relevant joint ranges and then improve the approximation of the robot bubbles for each half until the radii of the robot bubbles are less than twice at large as the corresponding sweep bubbles. This heuristic couples the resolutions in W-space with the resolution in C-space (section 3).

Figure 5. A bubble enclosing part of the second link is first swept over range \(d\beta\), the sweep bubble resulting is swept over range \(d\alpha\).

The path planner described in section 3 keeps for each block a list of robot bubbles. Only bubbles which sweep bubbles intersect with the obstacles stay on the list. If a block is split the sweep bubbles are recalculated and reevaluated for each half. Sweep bubbles which do not collide are removed from the list. If no bubbles remain a block is free, else it is uncertain.

Since it is cumbersome to have a list of obstacle bubbles for each sweep bubble (one sweep bubble can intersect more than one obstacle), we propose to use a distance transform of the free space \([8][10]\). A distance transform yields for all points within the free space the distance to the nearest obstacle. Mica \([11]\) calls these distances space bubbles, see figure 6. Discretization of the free space is doable since the dimensionality of the W-space is at most three. Relation 1 becomes:

\[
r_f < r_b \rightarrow \text{no collision}
\]

(3)

where \(r_f\) is the radius of the space bubble = the distance to the closest obstacle at the center of the robot bubble.

The problem of finding a hierarchy of bubbles for a robot remains. We will treat that in the next section.

Figure 6. Space bubble encloses sweep bubble of a mobile robot = no collision
5. ROBOTS ENCAPSULATED BY BUBBLES

Several methods can be described to find a hierarchy of bubbles. We need to find a hierarchy only once for each robot. We will present two simple methods, one based on triangles and one on boxes. More efficient bubble placement algorithms can be described, e.g. based on the medial axis [12], but these do not alter the validity of the paradigm presented. Better hierarchies will approximate a given robot better at a certain level in the hierarchy. Since these algorithms are too elaborate we will not treat them in this paper.

In 2-d each link of a robot is approximated by a triangulated polygon. Each sharp triangle on the outside of the polygon is enclosed by a bubble, the center of which lies at the intersection of the three perpendicular bisectors of the sides (for a obtuse triangle at the center of the longest side). To obtain the next level in the hierarchy, we split the triangle into four congruent triangles. Only the three on the outside need to be enclosed in bubbles. The radius of each of those bubbles is half the radius of the original bubble. Each center lies midway the original center and one of the corners of the original triangle. We now repeat this procedure for the three new triangles. Note that this method treats a link as a collection of triangles. The bubbles surrounding the triangles are merged pairwise into larger bubbles.

In 3-d each link is represented by a box. This box is surrounded by a bubble. To obtain the next level in the hierarchy, we split the box along its longest side and obtain two new boxes. These boxes are enclosed again by a bubble. The process continues up to a necessary resolution.

Boxes (as with the triangles) completely inside a figure do need to be modelled. Although a configuration can erroneously be classified as prohibited, such a configuration can never be reached via a path of contiguous collision free configurations.

6. RESULTS

We have tested our method using two robots: a 2 degrees of freedom planar arm (example 1) and a 5 degrees of freedom industrial manipulator (example 2).

Figure 7a shows the 2 degrees of freedom planar arm in the start configuration. Link 1 consists of 8 lines, link 2 of 4 lines. Both links can rotate indefinitely (periodic boundary conditions). The finest resolution allowed was 2π/512 for both links. The processing time to find a first path (length in C-space 545 units, figure 7b) was 0.9 seconds on a SUN4/260 (without graphics). The memory usage was 30 kBytes. 60 blocks of C-space were explored.

The best path (length 332 units, figure 8) took 9.3 seconds and 348 kBytes. 3632 blocks were explored.

![Figure 7a](image1.png)
![Figure 7b](image2.png)

**Figure 7.** Path find example 1 (a) Robot in start configuration (b) First path found, robot stays away from obstacles as much as possible

![Figure 8a](image3.png)
![Figure 8b](image4.png)

**Figure 8.** Path find example 1, (a) First part of the optimal path (b) Second part of the optimal path.
Figure 9 shows a 5 degrees of freedom industrial manipulator, a table and two pillars of wooden block. The table and pillars were located by a range finding system, figure 10 [13]. Start and goal were placed to the right and left of the wooden blocks (512 units apart, which would be the path length if no obstacles were present).

In figure 11 the joint angles belonging to the first path are shown (see figure 13 for the notation). The robot lifts his arm (joints 2 & 3), pulls it in (joints 2 & 3) and rotates around his base (joint 1). The freedom left for a trajectory planner is defined by the space between the upper and lower contours which represents the size of blocks in C-space. Wrist action (joint 4 and 5) is less constrained than arm action because of the low spatial impact of gripper movements.

Figure 12 shows the path after 615 seconds of calculation time. The path is shorter because the arm is less lifted and less pulled in. The average size of the blocks on the path has decreased; shorter paths are located closer to obstacles.

Note that the execution time required to find an optimal path will still be exponentially dependent on the resolution in C-space and the dimensionality of C-space. A number of (N-1)-dimensional hyperplanes bounding the prohibited regions in N-dimensional C-space has to be detected up to the finest resolution (assuming a non trivial problem). That is why we stopped the calculation process after 10 minutes.

The first path was 1326 units long. It took 19.6 seconds to find and 340,000 kBytes of memory were used for 3380 C-space blocks. After 615 seconds the path length had dropped to 846 units, 67745 blocks were explored taking 22 MBytes. Times and memory usage do not include the time taken to distance transform the free space and the memory it occupied (9 seconds, 100*100*100 voxels, 2 MBytes).

Figure 10. Obstacles as discovered by the range finding system.

Figure 11. Joint angles $\theta$, for path find example 2. All angles in degrees. First path found.

Figure 12. Joint angles $\theta$, for path find example 2. All angles in degrees. Path after 615 seconds of calculation time.
ACKNOWLEDGEMENT

This work started while the author worked as a summer student at Philips Laboratories, Briarcliff Manor, New York. Sweep bubbles are an original idea of Dr. Roy Featherstone. The stimulating discussions with him and with Dr. Leo Dorst, who created the opportunity for me to stay at Philips are greatly appreciated. The work is now supported by the SPIN-FLAIR project as part of the Delft Intelligent Assembly Cell. Eric Bouts, Stefano Mica and Piet Verbeek made valuable contributions to parts of this work.

REFERENCES


7. CONCLUSIONS

We have proposed an efficient path planning method. Our approach applies multi-resolution techniques in both W- and C-space. C-space is locally tessellated by binary division in order to profit from the natural ordering of links. Bubbles are used to approximate the geometry of the robot in W-space and to conclude 'no collision' as quickly as possible. The resolution in W-space is increased in step with the resolution in C-space.

A first path is found as quickly as possible by searching bidirectionally and by giving higher priority to larger blocks in C-space. The algorithm converges to the optimal path given the set of obstacles and assuming a certain metric in configuration space.

We have tested the algorithm up to a 5 degrees of freedom robot. The processing times all stay within acceptable limits.
Ik zie dat u op een slotbeschouwing zit te wachten, maar u bent al te dwaas als U denkt dat ik me nog iets herinner van wat ik heb gezegd; ik heb er van alles zoveel uitgeflapt! Het oude spreekwoord zegt: 'Ik haat de man die niet vergeet wat onder 't drinken is gezegd.' Het nieuwe luidt: 'Ik haat de man die niet vergeet wat hij bij 't luisteren heeft gehoord.' Daarom: vaarwel, betoon Uw bijval, geniet van het leven en drink, gevierde dienaren der ZOTHEID.

(I see that you are waiting for some concluding observations, but you are too silly if you think that I remember anything about what I've said, I blurted out so much about so many things! The old saying goes: 'I hate the man who doesn't forget what has been said while drinking.' The new one goes: 'I hate the man who doesn't forget what he has heard while listening.' Thus: goodbye, show your approval, enjoy life and drink, beloved servants of FOOLISHNESS.)

_Erasmus, Lof der zotheid_
CONCLUSIONS

Metrics

In the real world the Euclidean metric holds. In the digital world the city-block and chess-board connectivity have often determined the metric. As shown in chapter 3, chamfer metrics can approximate the Euclidean metric better. These metrics build distances by a sequence of local vectors, the length of each of which is optimized with respect to the final goal: a minimum mean square error or maximum error of the estimated distance with respect to the real distance.

For applications in which other metrics apply (chapter 6, chapter 7) a chamfer approach can often be a satisfying solution.

For those metrics, and for metrics used in robot configuration spaces, be they homogeneously sampled or not, error calculations still need to be developed.

Algorithms

The distance transform algorithms all built around the concept of sorting. Efficient sorting allows wavefronts to be processed in order of increasing distance. Then each pixel is addressed only once. The most simple sorting mechanism is the queue (chapter 4). It is preferably used in situations where metrics are not important, since it implies a city-block or chess-board metric (see also Vrooman & Maas, 1989 and Schmidt, 1989).

To approximate the Euclidean metric, bucket sort is best (chapter 2). It is still linear in the number of pixels sorted and involves only slightly more overhead than a queue structure. It can also be used to implement other homogeneous metrics, like a parabolic ‘metric’ (chapter 6). For non-homogeneous metrics, as encountered in the grey-weighted distance transform, bucket sort is cumbersome. Then heap sort is best (Dorst & Trovato 1988). A heap takes order $n \log n$ time when $n$ pixels have to be sorted. But it can sort real-valued numbers.

Applications

A number of applications are treated in this thesis, all of which use distance transforms as a basic tool. The flexibility of the tool can be judged by the variety of applications, from adaptive smoothing, via design of coins and seismics to robot path finding. Probably this list will become even more diverse in the future. New areas of interest are already being searched, e.g. cytometry and fine motion planning.
‘Things aren’t quite as simple as that in the real world, mister. There are plenty of
Pages with the same numbers; so they have to work out which ‘Chapter’ they
belong in, in which ‘Volume’, and so forth. Also quite often there are errors in the
uniforms, so they’ve got on completely the wrong number anyway.’

Salmon Rushie, Haroun and the Sea of Stories, 1990

REFERENCES

D.H. Ballard, C.M. Brown
Computer Vision

A.L.D. Beckers
Measurement of parameters for non-linear objectsize-
filters in images (in Dutch)

A.L.D. Beckers, A.W.M. Smeulders
A comment on “A note on ‘Distance transformations
in digital images’”
Computer Vision, Graphics and Image Processing 47

A.L.D. Beckers, A.W.M. Smeulders
The probability of a random straight line in two and
three dimensions

G. Borgefors
Chamfering: a fast method for obtaining approxi-
mations of the Euclidean distance in N dimensions
Proc. 3rd Scandinavian Conf. on Image Analysis,
Copenhagen, 1983, 250-255.

G. Borgefors
An improved version of the chamfer matching al-
gorithm
Proc. International Conference on Pattern
Recognition, Montreal, 1984, 1175-1177

G. Borgefors
Distance transformations in arbitrary dimensions
Computer Vision, Graphics and Image Processing 27

G. Borgefors
Distance transformations in digital images
Computer Vision, Graphics and Image Processing 34

E. Bouts
Collision detection with bubble hierarchies

R. A. Brooks
Solving the find-path problem by good representation
of free space
IEEE Trans. on Systems, Man and Cybernetics 13
(1983) 190-196

R. A. Brooks
A layered intelligent control system for a mobile robot

P. Broos
Queens portrayed by hills, slopes and contour lines

J.H. Connell
A behavior-based arm controller
784-791.

P.E. Danielsson
Euclidean distance mapping
Computer Vision, Graphics and Image Processing 14

P.P. Das, B.N. Chatterji
A note on “Distance transformations in digital images”
Computer Vision, Graphics and Image Processing 43

P.P. Das, B.N. Chatterji
Knight’s distance in digital geometry
S.T. Dekker  
Towards a hardware architecture for robot path finding  

E.W. Dijkstra  
A note on two problems in connexion with graphs  

L. Dorst  
Discrete straight line segments: parameters, primitives and properties  

L. Dorst, P.W. Verbeek  
The constrained distance transformation: a pseudo-Euclidean, recursive implementation of the Lee-algorithm  

L. Dorst  
Pseudo-euclidean skeletons  

L. Dorst, K. Travato  
Optimal path planning by cost wave propagation  

L. Dorst, I. Mandhyan, K. Travato  
The geometrical structure of path planning problems  

S.E. Dreyfus  
An appraisal of some shortest-path algorithms  
Operations Research 17 (1969) 395-412

R. Featherstone  
Method and apparatus for determining potential collisions using a swept bubble data hierarchy  

M.A. Fischler, J.M. Tenenbaum, H.C. Wolf  
Detection of roads and linear structures in low resolution aerial imagery using a multisource knowledge integration technique  

P.G. Frank, P.G. Bergmann, A. Yaspan  
Ray Acoustics  

K.S. Fu, R.C. Gonzalez and C.S.G. Lee  
Robotics, Control, Sensing, Vision and Intelligence  

A.C. Geerlings  
Heuristic tracking in seismic data  

A.C. Geerlings  
Adaptive tracking in seismic exploration  

W. Gilles  
An universal computerprogram for the determination of a collision-free path for an industrial robot, Graduation thesis (in Dutch)  

F.C.A. Groen, P.W. Verbeek  
Design of a Royal poststamp (in Dutch)  

F.C.A. Groen, N.J. Foster  
A fast algorithm for cellular logic operations on sequential machines  

R. Haralick, S.R. Sternberg, X. Zhuang  
Image analysis using mathematical morphology  

P.E. Hart, N.J. Nilson and B. Raphael  
A formal basis for the heuristic determination of minimum cost paths,  

C.J. Hilditch  
Linear Skeletons from Square Cupboards  

G. Honderd, W. Jongkind, C.H. van Aalst  
Sensor and navigation system for a mobile robot  

C. Huygens  
Traité de la Lumière  
1690

S. Inokuchi, K. Sato, F. Matsuda  
Range imaging system for 3-D object recognition  
P.P. Jonker, S.T. Dekker, B.J.H. Verwer
A hardware architecture for robot path planning
IAPR workshop on computer vision, special hardware
and industrial applications, 1988, Tokyo.

W. Jongkind, G. Honderd, B.J.H. Verwer, T. Rieswijk
The Delft intelligent assembly cell and its collisions
avoidance
Proc. of the Int. Conf. on Automation, Robotics and
Computer Vision, Singapore, 1990, 170-174

Kelly, Alford and Whitmore
Modeling - the forward method
in: Concepts and Techniques in Oil and Gas
Exploration, K.C. Jain and R.J.P. deFigueiredo (eds.), Soc.of Exploration Geophysicists, Tulsa,

Z. Kulpa
Area and perimeter measurement of blobs in discrete
binary pictures
Computer Vision, Graphics and Image Processing 6
(1977) 434-454.

C.J. Lee
An algorithm for path connections and its applications

J.M. Lester, H.A. Williams, B.A. Weintraub, J.F.
Brenner
Two graph searching techniques for boundary finding
in white blood cell images

S. Lobregt, P.W. Verbeek, F.C.A. Groen
Three-dimensional skeletonization: principle and algorithm
IEEE Trans. on Pattern Analysis and Machine
Intelligence 2 (1980) 75-77.

T. Lozano-Pérez
Spatial planning: a configuration space approach

T. Lozano-Pérez
Automatic planning of manipulator transfer movements
IEEE Trans. on Systems, Man and Cybernetics 11

T. Lozano-Pérez
A simple motion-planning algorithm for general robot
manipulators
IEEE J. of Robotics and Automation 3 (1987) 224-
238.

V.J. Lumelsky
Guest editorial
IEEE J. of Robotics and Automation 3 (1987) 169-
171.

A. Martelli
On the complexity of admissible search algorithms

L. Méro
An heuristic search algorithm with modifiable estimate

S. Mica
Finding a path with a bubble hierarchy

U. Montanari
A method for obtaining skeletons using a quasi-
Euclidean distance

U. Montanari
Continuous skeletons from digitized images

T.J. Moser
Efficient seismic ray tracing using graph theory
Soc. Exploration Geophysicists Dallas (1989), 1106-
1108.

W. Newman, N. Hogan
High speed robot control and obstacle avoidance using
dynamic potential functions
Proc. IEEE Int. Conf. on Robotics and Automation,

H. Ozaki, A. Mohri, M. Takata
On the collision free movement of a manipulator

R.P. Paul
Robot Manipulators: Mathematics, Programming and
Control

J. Pearl
Heuristics, Intelligent Search Strategies for Computer
Problem Solving
Reading MA, Addison-Wesley, 1984.

J. Pecht
Speeding up successive Minkowski operations with
bitplane computers

J.W. Perram, M.S. Wertheim
Statistical mechanics of hard ellipsoids, I. Overlap
algorithm and the contact function
J. Piper
Efficient implementation of skeletonisation using interval coding,

J. Piper, E. Granum
Computing distance transformations in convex & non-convex domains

I. Ragnemalm
Generation of Euclidean distance maps

A.A.G. Requicha, H.B. Voelcker
Solid modeling: a historical summary and contemporary assessment

A. Rosenfeld, J.L. Pfaltz
Sequential operations in digital picture processing

A. Rosenfeld, J.L. Pfaltz
Distance functions in digital pictures
Pattern Recognition 1 (1968) 33-61.

A. Rosenfeld, A.C. Kak
Digital Picture Processing

D. Rutovitz
Datastructures for operations on digital images
in: Pictorial Pattern Recognition, G.C. Cheng, R.S. Ledley, D.K. Pollock, A. Rosenfeld (eds.),
Thompson, Washington DC, 1968, 105-133.

H. Saito
Traveltimes and raypaths of first arrival seismic waves: computation method based on Huygens' principle

W. Schmidt
Segmentation and analysis of range images

M. Schroeder
The Eikonal Equation

J.T. Schwartz
On the piano movers' problem II, General techniques for computing topological properties of real algebraic manifolds

J. Serra
Image Analysis and Mathematical Morphology

K.G. Shin, N.D. McKay
Selection of near-minimum time geometric path for robotic manipulators

S.R. Sternberg
Language and architecture for parallel image processing

S.H. Suh, K.G Shin
A variational dynamic programming approach to robot-path planning with a distance-safety criterion

J. Toriwaki, S. Yokoi, T. Yonekura, T. Fukumara
Topological properties and topology-preserving transformation of a three-dimensional binary picture

J.I. Toriwaki, N. Kato, T. Fukumura
Parallel local operations for a new distance transformation of a line pattern and their applications

K. Travato, L. Dorst
Autonomous planning of vehicle maneuvers

P.W. Verbeek
A class of sampling-error free measures in oversampled band-limited images

P.W. Verbeek, F.C.A. Groen, H.A. Vrooman, B.J.H. Verwer
Speeding up 3-d binary image processing for robot vision

P.W. Verbeek, L. Dorst, B.J.H. Verwer, F.C.A. Groen
Collision avoidance and path finding through constrained distance transformation in robot state space
P.W. Verbeek, H.A. Vrooman, L.J. Van Vliet
Low-level image processing by max-min filters

P.W. Verbeek, B.J.H. Verwer
2-D adaptive smoothing by 3-D distance transformation

P.W. Verbeek, B.J.H. Verwer
Robot Collision Avoidance,
Flexibele Automatisering en Informatica, Centrum voor Informatica en Computertechniek, Delft, 1989,
84-91.

P.W. Verbeek, B.J.H. Verwer
Shading from shape, the eikonal equation solved by
grey-weighted distance transformation

B.J.H. Verwer
Implementation of a 3d image processing package on
the GEMINIX, including a fast algorithm for
neighborhood and skeleton operations (in Dutch),

B.J.H. Verwer
Distance transforms for path planning

B.J.H. Verwer
Heuristic search in robot configuration space using
variable metric
Proc. 3rd Conf. on Artificial Intelligence for Space
Applications, Huntsville, Alabama, 1987, NASA
Conf Publ. 2492, 153-154.

Robot path planning by heuristic search
Proc. 12th world congr. scientific computation

B.J.H. Verwer
Improved metrics in image processing applied to the
Hilditch skeleton
Proc. International Conference on Pattern
Recognition, Rome, 1988, 137-142.

B.J.H. Verwer, P.W. Verbeek, S.T. Dekker
An efficient uniform cost algorithm applied to distance
transforms
IEEE Trans. on Pattern Analysis and Machine

B.J.H. Verwer, F.C.A. Groen, R.J. Ekkers, R. Ellens
Image processing on personal computers
Proc. IEEE Int. Conf. on Systems Engineering,
Fairborn, Ohio, 1989, 183-186.

B.J.H. Verwer
A multiresolution work space, multiresolution con-
figuration space approach to solve the path planning
problem
Proc. IEEE Int. Conf. on Robotics and Automation,
Cincinatti, Ohio, 1990, 2107-2112.

L.J. van Vliet, B.J.H. Verwer
A contour processing method for fast binary neighbour-
hood operations

L.J. van Vliet, I.T. Young, A.L.D. Beckers
A non-linear laplace operator as edge detector in noisy
images
Computer Vision, Graphics and Image Processing 45

H.B. Voelecker, A.A.G. Requicha
Geometric modeling of mechanical parts and pro-
cesses

A.M. Vossepoel
A note on “Distance transformations in digital images”
Computer Vision, Graphics and Image Processing 43

H.A. Vrooman
Processing 2-D and 3-D images in a C/UNIX envi-
ronment with an application to magnetic resonance
images

H.A. Vrooman, A.A.M. Maas
New image processing algorithms for the analysis of
speckle interference patterns Proc. Symp. on Optical &
Optoelectronic Applied Science & Engineering, San

M. Yamashita, T. Ibaraki
Distances defined by neighborhood sequences

S. Yokoi, J. Toriwaki, T. Fukumura
On generalized distance transformation of digitized
pictures
IEEE Trans. on Pattern Analysis and Machine

S. Yokoi, J. Toriwaki, T. Fukumura
Theoretical considerations on a family of distance
transformations and their applications
in: Real-time/parallel computing, M. Onoe, K. Preston
jr., A. Rosenfeld, Plenum NY, 1981.
I.T. Young, R.C. Peverini, P.W. Verbeek, P.J. van Otterloo
A new implementation for the binary and Minkowski operators

I.T. Young
Measurement accuracy in digital image analysis
Abstracts of the British Pat. Recgn. Ass.'s 2nd Int. Conf. on Pat. Recognition, Oxford University, 17.

T.P.H. Warmerdam, B.J.H. Verwer
Method and apparatus for controlling robots and the like using a bubble data hierarchy placed along a medial axis

M. Werman, S. Peleg
Min-max operators in texture analysis

S. Wolfram
Mathematica™, A System for Doing Mathematics by Computer
De beste historiestijl, heeft men in onze dagen naar waarheid gezegd, is nog altijd de stijl van Rembrand, veel weglaten, veel overdrijven, en op een klein getal feiten of beweegredenen veel licht doen vallen.

(The best style of history, as has been truthfully told in our days, is still the style of Rembrand, leave out much, overdo much, and shine, on a small number of facts or reasons, much light)

Conrad Busken Huet, Het Land van Rembrand

SUMMARY

Distance transforms are operations which calculate for each point in an image the distance to a reference set. This distance can be measured as the crow flies or restricted by obstacles. The latter case is interesting for path finding in robotics. Other applications include binary image processing, adaptive smoothing, measurement and design.

Distances are defined by metrics. In the discrete world we normally need to approximate the continuous Euclidean metric. For this purpose optimal parameters in two and three dimensions are calculated. A position dependent metric (grey-weighted) is introduced for the design of a coin, a anisotropic ‘metric’ (parabolic) for adaptive smoothing.

Bucket sort is the best way to determine the order in which to process pixels. In cases where accuracy requirements are low the simple queueing mechanism suffices, when using position dependent metrics heap sort is more appropriate, albeit more cumbersome.

SAMENVATTING

Afstandstransformaties zijn operaties die voor ieder punt in een beeld de afstand tot een referentieverzameling uittrekenen. Deze afstand kan hemelsbreed of rekening houdend met obstakels gemeten worden. Het laatste geval is interessant voor het padzoek probleem in de robotica. Andere toepassingen zijn onder andere binair beeldbewerken, adaptief effenen, meten en ontwerpen.


De volgorde waarin punten behandeld worden kan het beste gestuurd worden met emmers. Als nauwkeurigheid niet van belang is, is een simpele rij voldoende. Als plaatsafhankelijke metrieën worden gebruikt is de hooibergmethode, hoewel omslachtiger, beter.
DANKWOORD


Ook Ted Young, enthousiast, prof en profi, bedankt.


Ook alle deelnemers aan DIAC hebben voor de nodige inspiratie gezorgd en van buiten Delft: Frans Groen, Karen Travato, Leo Dorst, Peter Struycken en Roy Featherstone, thanx.

Jan Straver, José Bil, Rob Ekkers en Wim van Oel, lekker die taart op zijn tijd.


Tenslotte, Liesa, you are always on my mind.
Falke & Lara, jullie lach verdrijft alle zorgen.
Pa & Ma, na 27 jaar hoeven jullie hier tenminste geen zorgen meer over te maken.
Nicole, je blijft een zusje uit duizenden.
Oma, je weet wel.
He is quick, thinking in clear images;
I am slow, thinking in broken images.

He becomes dull, trusting to his clear images;
I become sharp, mistrusting my broken images.

Trusting his images, he assumes their relevance;
Mistrusting my images, I question their relevance.

Assuming their relevance, he assumes the fact;
Questioning their relevance, I question the fact.

When the fact fails him, he questions his senses;
When the fact fails me, I approve my senses.

He continues quick and dull in his clear images;
I continue slow and sharp in my broken images.

He in a new confusion of his understanding;
I in a new understanding of my confusion.

Robert Graves, In broken images

RESUME

Ben Verwer was born June 28, 1963 in Leidschendam. At the Huygens Lyceum Voorburg he laid the foundation for a technical study, although he spend considerable time on classical languages. His study Applied Physics in Delft started in 1981 and was brought to a good result in 1987.

From 1984 onwards he studied at the Interuniversitary Interfaculty Management Sciences in Delft and at the Erasmus University in Rotterdam, which lead to an MBA in 1986.

The last 4 years Ben worked on his PhD in the Pattern Recognition Section of the Applied Physics Faculty in Delft, continuing his MSc. work on path planning with distance transforms.

Ben is married to Liesa Chapman and has two children, Falke and Lara.

CURRICULUM VITAE


Vanaf 1984 studeerde hij ook Bedrijfskunde aan eerst de Interuniversitaire Interfaculteit in Delft en later de Erasmus Universiteit in Rotterdam, afgerond in 1986 met een doctoraal.

De laatste 4 jaar heeft Ben aan zijn promotie gewerkt in de sectie Patroonherkenning van Technische Natuurkunde Delft, voortbouwend op zijn afstudeerwerk over pad plannen met behulp van afstandstransformaties.

Ben is getrouwd met Liesa Chapman en heeft twee kinderen, Falke en Lara.