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

CITR-TR-54 November 1999

The Length of Digital Curves

Reinhard Klette[1] and Ben Yip[1]

Abstract

The paper discusses one of the elementary subjects in image analysis: how to measure the length of a digital curve? A digital curve in the plane is defined to be a cycle given either as an alternating sequence of vertices and edges, or an alternating sequence of edges and squares. The paper reports about two length estimators, one based on the partition of a frontier of a simply-connected isothetic polygon into digital straight segments, and one based on calculating the minimum-length polygon within an open boundary of a simply-connected isothetic polygon. Both techniques are known to be implementations of convergent estimators of the perimeter of bounded, polygonal or smooth convex sets in the euclidean plane. For each technique a linear-time algorithm is specified, and both algorithms are compared with respect to convergence speed and number of generated segments. The experiments show convergent behavior also for perimeters of non-convex bounded subsets of the euclidean plane.

[1] The University of Auckland, Computer Science Department, CITR,
Tamaki Campus, Glen Innes, Auckland, New Zealand

The Length of Digital Curves

Reinhard Klette and Ben Yip

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

Abstract. The paper discusses one of the elementary subjects in image analysis: *how to measure the length of a digital curve?* A digital curve in the plane is defined to be a cycle given either as an alternating sequence of vertices and edges, or an alternating sequence of edges and squares. The paper reports about two length estimators, one based on the partition of a frontier of a simply-connected isothetic polygon into digital straight segments, and one based on calculating the minimum-length polygon within an open boundary of a simply-connected isothetic polygon. Both techniques are known to be implementations of convergent estimators of the perimeter of bounded, polygonal or smooth convex sets in the euclidean plane. For each technique a linear-time algorithm is specified, and both algorithms are compared with respect to convergence speed and number of generated segments. The experiments show convergent behavior also for perimeters of non-convex bounded subsets of the euclidean plane.

1 Introduction

The field of digital geometry and topology has an extensive list of publications, see the bibliography by A. Rosenfeld in [9]. This paper discusses one important aspect in this field, the multigrid convergence of estimators of the length of digital curves. It follows our previously published conference paper [10] which had space limitations, and we provide a more extensive presentation in this paper. For example, we specify the convergence theorems for multigrid measurements of the length of a curve and provide a detailed discussion of related implementations and experimental results.

Recent literature for image analysis still recommends different formulas combining numbers of local space configurations along a specified boundary of a digital simply-connected set [1] for estimating the length of a curve, or the perimeter of a planar simply-connected bounded set. However, it is well-known for several years that these *local methods* (ie. the approximating elements are always defined within sets of constant maximum diameter) are imprecise and, what is more important, their precision cannot be improved by increasing the resolution of the digitization. The phenomenon is well-known from popular examples such as the failure of measuring the length of a diagonal by the length of an approximating

[1] Definitions of digital sets, connectedness, and boundary are often based on regular grids and "m-neighborhood" definitions, eg. for $m = 4$, $m = 6$ or $m = 8$. We do not follow this approach.

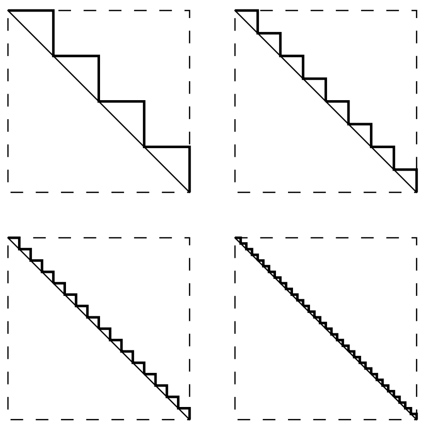

Fig. 1. The multigrid staircase approximates the diagonal of a square with respect to the Hausdorff distance, but the length of the multigrid staircase (length of a so-called "4-curve") remains constant twice the side length of the given square.

(in the sense of the Hausdorff distance) staircase, see Fig. 1. Experiments and theoretical studies reported in the literature lead to conclusions about the failure of local methods, see, eg., [19, 20, 30].

The digitization of curves or boundaries has been studied in image analysis for more than 30 years, see the bibliography by A. Rosenfeld in [9]. The paper deals only with curves, ie. boundaries of simply-connected sets. However, the discussed algorithms may also be easily modified for arcs, ie. connected subsets of curves. Curves and digital curves are introduced in Section 2. This section also informs about the multi-grid digitization approach.

The paper presents in Section 3 a linear-time DSS algorithm [14] and in Section 4 a linear-time MLP algorithm [10]. For both methods further algorithms have been discussed elsewhere, for example three linear-time DSS algorithms [4] and basic ideas of MLP algorithms [24, 31]. The selected algorithms are chosen because of its simplicity and computational efficiency.

Both algorithms are described in detail to support reimplementations. Section 5 compares both methods with respect to the speed of convergence and the number of generated segments, and compares both algorithms with respect to its computational time complexity.

Finally, Section 6 lists a few conclusions and open problems. For example, it would be worthwhile to extend the comparative study given in this paper for two algorithms only, to a broader range of test sets, or algorithms with respect to a broader diversity of measurements or algorithmic features.

2 Curves and Digital Curves

The specification of the length of a curve (normally assumed to be a set in the euclidean plane) is one of the fundamental problems in mathematics. Studies of different curve definitions (in different geometries) and different approaches for length measurement are important subjects in mathematics. This section specifies curves, digital curves and the multi-grid digitization model.

2.1 Curves in the euclidean plane

In the euclidean geometry there are basically two different ways for defining a curve, via parametrization or via topological mappings.

A *planar Jordan curve* γ as defined by C. Jordan in 1893 is given by a parameterized path $\phi : [a, b] \to R^2$ with $a \neq b$, $\phi(a) = \phi(b)$, $\phi(s) \neq \phi(t)$, for all s, t with $a \leq s < t < b$. It holds that

$$\gamma = \{(x,y) \ : \ \phi(t) = (x,y) \ \wedge \ a \leq t \leq b\} \ . \tag{1}$$

A *planar Jordan arc* γ is defined by a subinterval $[c, d]$ with $a \leq c < d \leq b$. A Jordan curve is also called a *closed Jordan arc.* Note that a Jordan curve is homeomorphic to the boundary of a unit circle, ie. it does not have "crossings" or "touchings". A *rectifiable Jordan arc* γ has a bounded *arc length*

$$|\gamma| = \sup_{c=t_0<\cdots<t_n=d} \sum_{i=1}^{n} d_2\left(\phi\left(t_i\right), \phi\left(t_{i-1}\right)\right) \quad < \infty \ , \tag{2}$$

where d_2 denotes the euclidean distance, here in 2D euclidean space.

It might be of interest to note that C. Jordan initially defined in 1883 a curve γ in parametric form as

$$\gamma = \{(x,y) \ : \ x = \alpha(t) \ \wedge \ y = \beta(t) \ \wedge \ 0 \leq t \leq b\} \ , \tag{3}$$

but G. Peano constructed in 1890 a curve satisfying Equ. (3) and filling the whole unit square. Equation (1) excludes the Peano curve. However, Equ. (3) is still in common use for analytical arc length calculations, and the arc length is equal to

$$d(\gamma) = \int_0^b \sqrt{\frac{d\alpha(t)}{dt}^2 + \frac{d\beta(t)}{dt}^2} \ dt \ , \tag{4}$$

assuming *smooth arcs* with C^1-functions α and β.

The Jordan curve definition is relevant for curves which possess parametric forms. Not all curves possess parametric forms, and some of them may have a parametric form but it may remain an open problem to find out about it. Especially in image analysis we have to deal with curves which are given in digitized pictorial form, and where a parametric description is typically not a final goal of the analysis process. Here, topological approaches for curve definitions seem to be more relevant, see, eg. [10, 16] for topological curve definitions based on the theory of abstract cellular complexes, or [25] for definitions based on the notion of a one-dimensional grid continuum.

The first topological definition of a curve was given by G. Cantor: a *planar Cantor curve* γ is defined to be a connected compact set in R^2 which does not have any internal points. This topological approach was further developed (eg. with respect to the analysis of branching points) by P. Urysohn and K. Menger. More details about the history of curve definitions in euclidean space may be found in [24, 25].

Complex topological situations in image analysis such as segmentations of digital images require a carefully designed topological theory, see, eg., the theory of abstract cellular complexes as discussed in [15, 17, 21, 27]. For readers familiar with the theory of abstract cellular complexes this paper could simply be based on notations and results of this theory. However, for the discussion of single digital curves we chose this time, different compared to [10], a presentation not based on abstract cellular complexes for the benefit of readers not familiar with this theory. However, the introduced notations for the euclidean plane follow notations of abstract cellular complexes. In the sense of [21] it may be stated that we use *simplicial euclidean complexes* as a special model of an abstract cellular complex in this paper, however we will not discuss this explicitly.

2.2 Digitizations of real preimages

The estimation of the perimeter of a measurable planar set behaves essentially different compared to the estimation of its area with respect to multigrid convergence: the area may be estimated by means of counting the number of space elements such as grid points contained in the given set. These studies have been initiated by C.F. Gauss (1777–1855) for disks. He and P. Dirichlet (1805–1859) knew already that the number of grid points inside of a planar convex curve γ estimates the area of the set bounded by this curve within an order of $\mathcal{O}(l)$, where l is the length of curve γ. Generally speaking, this estimate converges to the true area for measurable sets in the euclidean plane. *Gridding techniques* (based on regular grids in euclidean space) avoid paradoxes such as Schwarz's paradox that the infinite sum of areas of triangles defined by a (disjoint) recursive triangulation of a rectangle may increase without bound [2].

Volume estimation was studied by *C. Jordan* [6] based on gridding techniques. Any grid point $(i, j, k) \in E^3$ is assumed to be the center point of a *closed grid cube* with faces parallel to the coordinate planes and with edges of length 1. Let S be a set contained in finitely many of such closed grid cubes. Dilate the set S with respect to an arbitrary point $p \in E^3$ in a ratio $r : 1$, where r is a positive real. This transforms S into S_r^p. Let $l_r^p(S)$ be the number of all closed grid cubes completely contained in the interior of S_r^p, and let $u_r^p(S)$ be the number of all closed grid cubes having a non-empty intersection with S_r^p. Then it holds [6] that $r^{-3} \cdot l_r^p(S)$ and $r^{-3} \cdot u_r^p(S)$ always converge towards limit values $L(S)$ and $U(S)$, respectively, for r to infinity, independent upon the chosen point p. Jordan called $L(S)$ the *inner volume* and $U(S)$ the *outer volume* of set S, or the *volume* $vol(S)$ of S if $L(S) = U(S)$. Again, as in the two-dimensional case: the estimation of the surface area of a measurable compact body behaves essentially different compared to the estimation of its volume with respect to gridding techniques. Adding just local configurations (as, eg., suggested in marching cubes, marching tetrahedron, opaque cubes, or dividing cubes approaches) does not lead to convergent surface area estimates, see [12]. Again we note that gridding techniques avoid paradoxes known for 3D body analysis such as the Banach-Tarski paradox [2].

This paper addresses the two-dimensional case. Estimates of the length of a planar curve cannot be based on counts of local space elements such as grid edges or grid diagonals. The scaled counts may not change at all when the grid resolution is increased, see Fig. 1, or may differ for different orientations of an otherwise identical curve before digitization.

For multi-grid studies we assume a Cartesian coordinate system and orthogonal grids with grid constant $0 < \vartheta \leq 1$ in the euclidean plane, ie. ϑ is the spacing between grid points parallel to one of the coordinate axes. Furthermore, let $r \geq 1$ be the *grid resolution* defined as being the number of grid points per unit, ie. any grid edge is of length $\vartheta = 1/r$. We consider *r-grid points* $g^r_{i,j} = (\vartheta \cdot i, \vartheta \cdot j)$ in the euclidean plane, for integers i, j. For $r = 1$ we may simply speak about *grid points* (i, j) in the euclidean plane.

Cellular complexes provide a proper mathematical model for sets of grid points and more complicated discrete structures [1, 8, 16]. For specifying cellular digitizations of subsets S of the euclidean plane we use cells as defined for simplicial euclidean complexes [21].

Definition 1. For $r \geq 1$, any r-grid point is the center of an open *r-square* whose *r-edges* are of length $1/r$ and parallel to the coordinate axes, and an r-edge connects two *r-vertices* but does not contain any of these two end-points.

The closure of an r-square contains four r-edges and four r-vertices. Besides the uniform Cartesian coordinate system we assume integer coordinates $[i, j]_r$ for r-vertices defining a specific coordinate system for grid resolution r. The proper grid resolution r has to be used to map these *integer r-vertex coordinates* back into the euclidean plane, ie. r-vertex $[i, j]_r$ has actually Cartesian coordinates $(1/r \cdot (i + 0.5), 1/r \cdot (j + 0.5))$. This formula specifies a general affine coordinate transform τ_r from the assumed Cartesian coordinates into integer r-vertex coordinates,

$$\tau_r(x, y) = [r \cdot x - 0.5, r \cdot y - 0.5]_r \ ,$$

for points (x, y) in the euclidean plane.

Let S be a bounded subset of the euclidean plane and $C_r(S)$ be the union of all closed r-squares with centers in S, $I_r(S)$ be the union of all closed r-squares completely contained in the interior of S, and $O_r(S)$ be the union of all r-squares having a non-empty intersection with set S. All three sets $C_r(S)$, $I_r(S)$ and $O_r(S)$ are closed and *isothetic*, ie. all of the boundary segments are parallel to one of the coordinate axes.

Definition 2. Let S be a planar set. Then the set $C_r(S)$ is its *center-point digitization*, and the sets $I_r(S)$ and $O_r(S)$ are the result of its *Jordan digitization* called *inner* and *outer digitization* of set S, respectively.

Note that these digitization results depend upon the chosen grid constant ϑ or grid resolution r. It holds $I_r(S) \subseteq C_r(S) \subseteq O_r(S)$, for any subset S of the Euclidean plane, and $I_r(S) \subset O_r(S)$, if S is not empty and closed. For any set S having a non-empty interior there exists an $r_0 \geq 1$ such that $\emptyset \subset I_r(S)$, for all $r \geq r_0$, where the set $I_r(S)$ may consist of more than one component.

For defining the notion of a boundary related to the Jordan digitization, we use the *Hausdorff-Chebyshev distance*

$$d_\infty(A, B) = \max\left\{\max_{p\in A}\inf_{q\in B} d_\infty(p,q), \max_{p\in B}\inf_{q\in A} d_\infty(p,q)\right\}$$

which generalizes the maximum-distance d_∞ between points to a metric between sets A, B of points. It holds that

$$d_\infty(\partial I_r(S), \partial O_r(S)) \geq \frac{1}{r} \tag{5}$$

for the boundaries of sets $I_r(S)$ and $O_r(S)$, for any non-empty, bounded and closed subset S of the euclidean plane.

2.3 Frontiers and open boundaries

In this section we define digital curves as special sequences of cells of simplicial euclidean complexes.

Definition 3. Let $r \geq r_0$ and S be such a planar set that its boundary $\partial C_r(S)$ is connected. Then $\partial C_r(S)$ is called an *r-frontier* (of set S).

This definition specifies r-frontiers which consist of a finite number of r-edges and r-vertices if the set S is bounded. Any run around an r-frontier of a bounded set S passes through an alternating sequence of r-edges and r-vertices. Note that this definition allows "touchings" but no "crossings".

Definition 4. Let $r \geq 1$ and S be a closed subset of the euclidean plane such that $I_r(S)$ is non-empty and simply-connected, and

$$d_\infty(\partial I_r(S), \partial O_r(S)) = 1/r \,.$$

The open set $O_r(S) \setminus (I_r(S) \cup \partial O_r(S))$ is an *open r-boundary* (of set S).

An open r-boundary is a finite, alternating sequence of r-edges and r-squares if the set S is bounded. If a given set S is such that the Hausdorff-Chebyshev distance between boundaries of $I_r(S)$ and $O_r(S)$ is greater than $1/r$ then there are different options within a multigrid approach: **(i)** This distance may be reduced to its minimum value of $1/r$ by shrinking of $O_r(S)$ and expanding of $I_r(S)$: We erode (by subtraction of closed r-squares and a final closure, see "peaks" C and D in Fig. 4) the set $O_r(S)$ into a subset $O_r^-(S)$ and dilate (by addition of closed r-squares, see cavities A and B in Fig. 4) the set $I_r(S)$ into a superset $I_r^+(S)$ such that $d_\infty(\partial I_r^+(S), \partial O_r^-(S)) = 1/r$. Because the sets $I_r(S)$ and $O_r(S)$ are

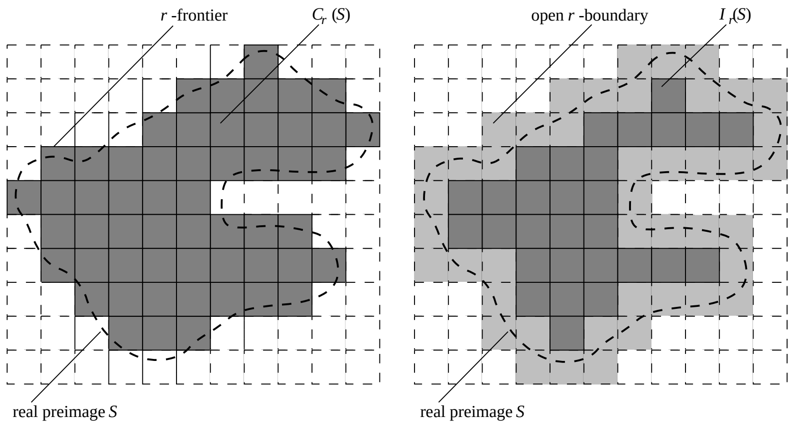

Fig. 2. Two digital r-curves both generated by the same real preimage S. Left: an r-frontier. Right: an open r-boundary.

unions of only a finite number of r-squares it follows that alternating sequences of erosion steps and dilation steps allow to construct (not necessarily in a unique way) sets $O_r^-(S)$ and $I_r^+(S)$. It also follows that both sets are again isothetic. Such a dilation-erosion approach is one option if digitizations of a given set S are considered at different resolutions r. Then an open r-boundary (of width $1/r$) can be constructed which is a subset of $O_r(S) \setminus (I_r(S) \cup \partial O_r(S))$. **(ii)** A second option is that an approximating algorithm is capable to deal with "cavities" or "peaks". We will go this way, and this is explained in Section 4.

Open r-boundaries consist of a finite number of r-squares and r-edges. Any run around an open r-boundary passes through an alternating sequence of r-squares and r-edges. These definitions are just euclidean versions of definitions cited in [10] from the theory of abstract cellular complexes.

Definition 5. A *digital r-curve* is either an alternating sequence of r-vertices and r-edges whose union is an r-frontier, or an alternating sequence of r-edges and r-squares whose union is an open r-boundary of a subset of the euclidean plane. A *digital r-arc* is a finite connected subsequence of a digital r-curve.

Both types of digital curves are illustrated in Fig. 2. Often we consider a given digital r-curve as being *directed.* This means that every r-edge has one *starting r-vertex* and one *end r-vertex* following a unique global orientation of the given digital r-curve.

Oriented digital r-arcs are specified as finite tuples $(v_1, e_1, v_2, e_2, ..., e_{n-1}, v_n)$ or $(e_1, s_1, e_2, s_2, ..., s_{n-1}, e_n)$, for $n \geq 1$, r-vertices v_i, r-edges e_i, and r-squares s_i. For example, vertex v_1 is the start r-vertex and v_n is the end r-vertex of a digital r-arc in case of the frontier model. These digital r-arcs are r-curves if $v_1 = v_n$, or $e_1 = e_n$.

The applied approach is just one example of a mapping of real preimages into a finite number of representing units. Such a mapping is related to ideas of Zeno of Elea (about 450 B.C.) that the real world is made up of indivisible units [2], which also may be called *Democritus' atoms*. Here, the real objects are not infinitely divisible anymore. The parameter r defines the scale of the atoms, but there are always just finitely many for a bounded set S. In general, we define that a *Zeno representation* of a given set is related to a specific geometric feature (area, perimeter etc.), depends upon one parameter only, the grid resolution r, and is always finite for a given bounded set.

After these preparations the perimeter measurement problem reduces to the problem of measuring the length of r-frontiers or open r-boundaries, but in a way such that these measurements converge towards the true value. The given sets S are assumed in Zeno representations $\partial C_r(S)$ or $O_r^-(S) \setminus (I_r^+(S) \cup \partial O_r^-(S))$, and both representations are relevant to the feature "perimeter".

2.4 Multigrid convergence

Let $\mathbf{F}$ be a family of sets S in the euclidean plane, such as the family of all smooth bounded convex sets. Multigrid digitization is of interest for studying feature calculations in image analysis in general [11]. Assume that a feature M is defined for all sets in the family $\mathbf{F}$, and that $Z_r(S)$ is a Zeno representation of set S at resolution r allowing approximate calculations of feature M.

Definition 6. We call an estimator $\mathcal{M}$ of M *convergent on* $\mathbf{F}$ with respect to the Zeno representation Z_r if

$$M(S) = \lim_{r \to \infty} \mathcal{M}(Z_r(S)) \ ,$$

for all sets $S \in \mathbf{F}$ also meaning that $\mathcal{M}$ is defined on all sets $Z_r(S)$, for all sets $S \in \mathbf{F}$, and for all $r \geq r_S$ where r_S may exclude a finite range of r-values smaller than r_S.

This multigrid convergence approach has been used by different authors, see, eg., [7, 9, 15, 24, 26, 28] for related work and references. Analyzing the accuracy of estimators means that the convergence needs to be satisfied (also towards the true value, see Fig. 1), and that the speed of convergence may be analyzed by specifying the worst-case or the expected error bounds. The feature of interest in this paper is the length of a rectifiable curve, and we consider two different estimators for this feature defined on Zeno representations introduced in Definitions 3 and 4.

In image analysis the input situation is characterized by a given (!) resolution of digital images, and digital curves arise in the form of specific substructures of such a digital image. Normally there is no way to refine the given image resolution, ie. the multigrid convergence idea seems to be irrelevant. However, the main argument for studying multigrid convergence is to ensure that the chosen algorithm is "sound" with respect to the given problem - to analyze a feature based on a given Zeno representation. And this soundness may be defined

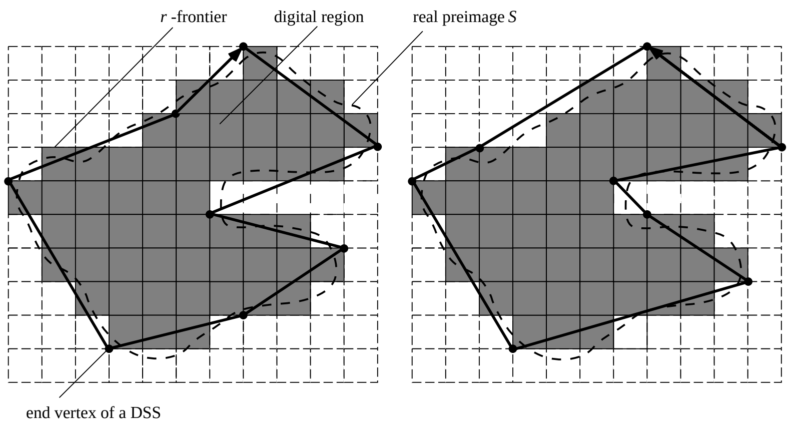

Fig. 3. Examples of partitions of the r-frontier into maximal DSS's, starting in both examples at the uppermost-leftmost r-vertex in $C_r(S)$. Left: clockwise. Right: counterclockwise. The dashed line shows the boundary of the real preimage S before center-point digitization.

in many ways: invariance with respect to geometric transforms, robustness with respect to noise etc. Multigrid convergence is just one aspect in this evaluation process of an algorithm whether it is "sound" or not, and if "yes" then improved imaging technology supporting images of higher resolution will actually lead to improved measurements.

2.5 Two convergence theorems

This paper presents a comparison of two methods, abbreviated as *DSS method* which is based on a partition of an r-frontier into **d**igital **s**traight **s**egments [5], see Fig. 3, and *MLP method* which is characterized by the calculation of the **m**inimum **l**ength **p**olygon in the closure of an open r-boundary [23], see Fig. 4. For both techniques it is known that the measured r-curve length converges towards the true value if a bounded, convex, polygonal or smooth subset of the euclidean plane is digitized with increasing grid resolution [15, 24]. The MLP method leads to a uniquely specified result, *the* minimum-length polygon. The DSS method may have different results depending upon orientation (clockwise or counter-clockwise) of the approximation procedure and the starting point.

The following theorem addresses the DSS method and provides not only an asymptotic upper bound $\mathcal{O}(1/r)$ but even an explicit specification of the asymptotic constant. The value of r_0 depends upon the given set, and $\varepsilon_{DSS}(r) \geq 0$ is an algorithm-dependent approximation threshold specifying the maximum *Hausdorff distance* (generalizing the euclidean distance between points to a distance between sets of points) between the r-frontier $\partial C_r(S)$ and the constructed, not

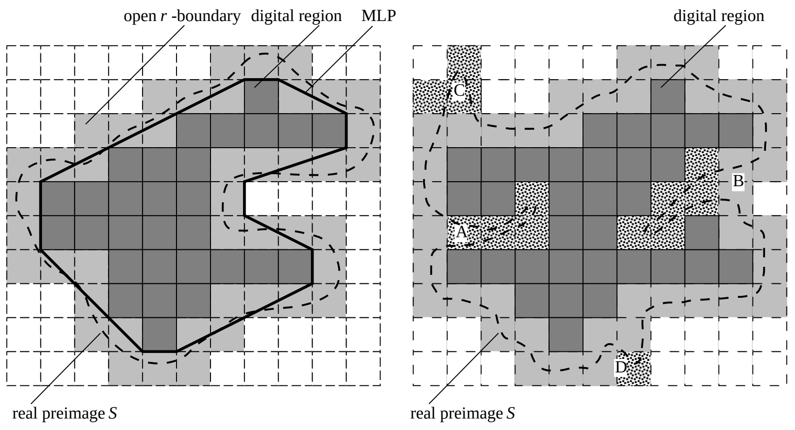

Fig. 4. Left: An example of an open r-boundary and its minimum length polygon for the same preimage (dashed line) as in Fig. 3 using the Jordan digitization scheme. Right: An isothetic polygon with cavity A ("hole entrance of width 1/r") and cavity B (width 2/r) illustrating two types of problems which may arise for a simply-connected inner digitization, and "peaks" such as C and D may arise for the outer digitization.

uniquely specified DSS approximation polygon. Its "classical" value is the grid constant, ie. $1/r$, see [22].

Theorem 7. (Kovalevsky/Fuchs 1992) *Let S be a bounded, convex, smooth or polygonal set in the euclidean plane. Then there exists a grid resolution r_0 such that for all $r \geq r_0$ it holds that any DSS approximation of the r-frontier $\partial C_r(S)$ is a connected polygon with perimeter l_r satisfying the inequality*

$$|Perimeter(S) - l_r| \leq \tfrac{2\pi}{r} \left(\varepsilon_{DSS}(r) + \frac{1}{\sqrt{2}} \right) .$$

This theorem and its proof may be found in [11] but the proof was actually fully based on material given in [15].

For the case of MLP approximations there are several convergence theorems in [24] showing that the perimeter of the MLP approximation is a convergent estimator of the perimeter, for bounded, convex, smooth or polygonal sets in the euclidean plane. Originally the MLP technique was only discussed for open r-boundaries, excluding situations such as illustrated on the left of Fig. 4. Our MLP algorithm will cover such situations as well.

The following theorem is basically a citation from [24] and specifies the asymptotic constant for MLP perimeter estimates. In [24] only powers of two have been considered as grid resolution r. However, it may be concluded from the material given in [24] (eg. Theorem 4.15 and Lemma 4.3) that the following theorem may actually be stated for arbitrary grid resolutions $r \geq 1$.

Theorem 8. (Sloboda/Zatko/Stoer 1998) *Let S be a bounded convex set in the euclidean plane such that its boundary is contained in the closure of the open r-boundary of S, for $r \geq 1$. Then it holds that the MLP approximation of the open r-boundary is a connected polygonal curve with length l_r satisfying the inequality*

$$l_r \leq Perimeter(S) < l_r + \frac{8}{r}\,.$$

These two theorems comprise the theoretical fundamentals of the curve length estimators discussed in this paper. Assuming $\varepsilon_{DSS}(r) = 1/r$ it follows that the upper error bound for DSS approximations is characterized by

$$\frac{2\pi}{r^2} + \frac{2\pi}{r \cdot \sqrt{2}} \approx \frac{4.5}{r}$$

and the upper error bound for MLP approximation is characterized by $8/r$. This ratio of about 1:2 is also the result of our experimental studies (as reported later: but *sliding means* are calculated instead of upper error bounds).

3 DSS approximation

This section discusses a convergent estimator for the perimeter of a bounded planar set S where an r-frontier is assumed to act as a Zeno representation of S. The estimator is based on a segmentation of such a digital r-curve into maximum length digital straight segments.

3.1 Principle of DSS approximation

Digital straight segments (DSS's) were initially defined by H. Freeman and A. Rosenfeld, see, eg., [22]. Typically, a DSS segmentation algorithm traces an r-frontier, ie. an alternating sequence of r-vertices and r-edges, and subdivides it into maximum length digital straight segments. A linear-time off-line algorithm is contained in [5], and linear-time on-line algorithms in [3].

The DSS algorithm presented in this paper follows [14]. This algorithm detects for each maximum length DSS the coordinates of its end points (two r-vertices) and calculates the length of each DSS as the euclidean distance between these two points. The sum of the lengths of these DSS's is finally used as *DSS estimator* of the perimeter. The algorithm is defined identically for different values of r which allows us to suppress r in the rest of this section. We speak about *frontiers*, *grid points*, vertex integer coordinates $[x, y]$, $C(S)$ etc. instead of r-frontiers, r-grid points, vertex integer coordinates $[x, y]_r$, $C_r(S)$ etc. However, for the perimeter estimation problem the scaling factor r has to be treated properly in the final multi-grid approximation algorithm.

Let S be a closed half-plane defined by a linear inequality. Then $C(S)$, $I(S)$ and $O(S)$ are isothetic, closed, unbounded subsets of the euclidean plane. These sets specify the frontier $\partial C(S))$ and the open boundary $O_r(S) \setminus (I_r(S) \cup \partial O_r(S))$ of half-plane S. These digital curves are dual representations of *digital straight* *lines*. For our DSS estimator it suffices to consider only bounded segments of frontiers.

Definition 9. A *digital straight segment* (DSS) is a finite, alternating sequence of vertices and edges whose union is a connected subset of the frontier of a half-plane.

The second option, a DSS defined as a finite, alternating sequence of edges and squares, whose union is a connected subset of the open boundary of a half-plane, corresponds to the notion of a *visual DSS* suggested in [18] as a preferred option for visualizations of digital straight segments (because our squares or their center points correspond to "pixels").

Correspondingly to Definition 9, for DSS estimators we only consider digital oriented arcs $g_n = (v_1, e_1, v_2, e_2, ..., e_{n-1}, v_n)$, $n \geq 1$ being subsequences of frontiers, and not of open boundaries.

We introduce an auxiliary notation. Assume an ordered pair of points (p_1, p_2) in the euclidean plane whose Cartesian coordinates are (x_1, y_1) and (x_2, y_2), respectively. A half-plane S is called to be *defined by the ordered pair* (p_1, p_2) if it contains exactly all those points (x, y) satisfying the inequality

$$h(x,y) = (x - x_1) \cdot (y_2 - y_1) - (y - y_1) \cdot (x_2 - x_1) \ \geq \ 0 \ . \tag{6}$$

Let $S_{\geq}(p_1, p_2)$ be the half-plane defined by the pair (p_1, p_2).

Now consider all ordered pairs of vertices (v_i, v_{i+k}), for integer $k > 0$, of a digital oriented arc g_n, for $n \geq 2$, such that the half-plane $S_{\geq}(v_i, v_{i+k})$ defined by this pair contains all vertices of g_n. A vertex pair of maximum euclidean distance satisfying this condition is called a *base* of g_n. The first vertex of a base is called the starting vertex of this base, and the second one is its end vertex.

Let (v_i, v_{i+k}) be a base of g_n and h be the defining function, ie. $h(x, y)$ is the value of the left-hand side of equation (6), $[x_1, y_1]$ are the integer vertex coordinates of the starting vertex v_i and $[x_2, y_2]$ are those of the end vertex v_{i+k}. Such a base (v_i, v_{i+k}) specifies a straight line $h(x, y) = 0$ which is called a *negative tangent* of the oriented digital arc g_n. The *positive tangent* is defined by *maximum vertices* of g where h takes its maximum value on the set $\{v_1, v_2, ..., v_n\}$. If there are at least two such maximum vertices then the positive tangent is defined to be the straight line incident with these maximum vertices. If there is only one maximum vertex then the positive tangent of g_n consists of this single vertex only. See Fig. 5 for an example of a negative and positive tangent.

Note that a dual approach could be to specify a half-plane $S_{\leq}(p_1, p_2)$ by the condition $h(x, y) \leq 0$, then the base of a digital oriented arc g_n, for $n \geq 2$, with respect to $S_{\leq}(p_1, p_2)$ which would also specify the positive tangent, and finally the negative tangent with respect to *minimum vertices*, i.e. of vertices producing the minimum value of h on the set $\{v_1, v_2, ..., v_n\}$. However, in this paper we use the approach as defined before ensuring that the negative tangent is always defined to be a straight line, for digital oriented arcs g_n with $n \geq 2$.

In previous publications [14, 18] these tangents have been called *right* and *left tangents* with respect to the orientation of g_n. The definition of these tangents

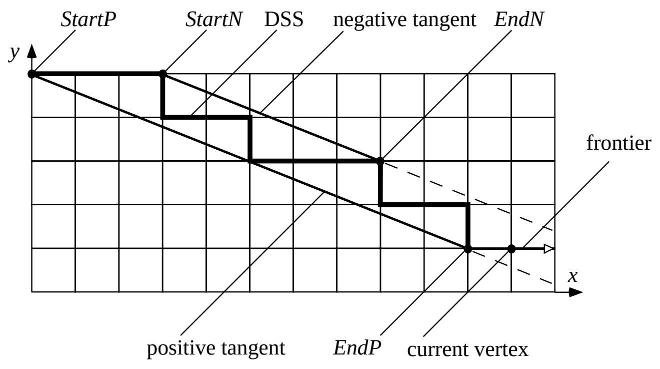

Fig. 5. The features of a DSS used in the DSS approximation algorithm.

depend upon the choice of the coordinate system. For example, if the positive y-axis may be reached by a 90° clockwise rotation of the positive x-axis (as it is usual in image processing) then the negative tangent lies on the right-hand side of the vertices of g_n. To ensure independence from the chosen coordinate system we use positive and negative tangents in this paper rather than right and left tangents since the DSS recognition by means of the sign of h is independent of the choice of the coordinate system.

If the given digital oriented arc is a DSS then its vertices (in integer vertex coordinates !) may be characterized by a triple a, b and c of integers [14].

Theorem 10. (Kovalevsky 1990) *All vertices $[x, y]$ of a DSS, containing $n \geq 2$ vertices, satisfy the following two inequalities*

$$0 \leq b \cdot x - a \cdot y + c \leq |a| + |b| - 1$$

where $[a, b]^T$ is a vector parallel to the negative tangent of the DSS having mutually prime integer components, and c is a constant integer chosen such that

$$b \cdot x - a \cdot y + c = 0 ,$$

for any vertex $[x, y]$ lying on the negative tangent.

These triples of integers a, b and c are defining parameters of a DSS. In the rest of this section we use the function $h(x, y) = h_{a,b,c}(x, y) = b \cdot x - a \cdot y + c$ defined by such a triple of integer parameters. Consider a digital oriented arc of n vertices whose first $n - 1$ vertices $v_i = [x_i, y_i]$ satisfy $h(x_i, y_i) \geq 0$. If $h(x_n, y_n) = -1$, then the vertex $v_n = [x_n, y_n]$ does not belong to the DSS with the given parameters a, b and c, but it is still possible that these parameters are updated in such a way that all n points belong to one DSS characterized by a triple of updated parameters.

This updating method is implemented in the function $DSS\_Cont()$, see Fig. 7 below. The value of $h(x, y) = -1$ corresponds to a "negative outlier", i.e. to a point which lies on the wrong side of the negative tangent. Similarly, there might be a positive outlier. In this case $h(x, y)$ is equal to $|a| + |b|$. In cases of $h(x, y) < -1$ or $h(x, y) > |a| + |b|$ there is no possibility of parameter adjustment, i.e. there exists no DSS containing all previous vertices together with the current vertex $[x, y]$.

3.2 DSS algorithm

The input sequence of vertices is assumed to follow a frontier of a digitized set. In other words, the vertex integer coordinates of two subsequent vertices in such a sequence with indices i and $i{+}1$ satisfy the condition $|x_{i+1}-x_i|+|y_{i+1}-y_i| = 1$. In image analysis applications the sequence is being generated during a process of tracing a frontier of a given isothetic connected polygon [16]. Assume that the *input* of the DSS algorithm is a sequence of n vertices $v[1 : n]$ on a frontier of a simply-connected isothetic polygon $C(S)$, each vertex being described by a data structure $(v[i].x, v[i].y)$ containing the vertex integer coordinates of the ith vertex $v[i]$. Since we assume a bounded frontier, $v[n]$ must be equal to $v[1]$. For the DSS algorithm the *output* is a value $Perim$ which is the DSS estimate of the perimeter of set S. The task consists in tracing a given digital oriented curve edge by edge, and sequentially subdividing it into DSS's of maximum length. To make the description more comprehensive we assume that all vertices of the frontier were put into the array $v[1 : n]$ specified above. The *DSS algorithm* reads the vertex integer coordinates of the n vertices $v[i]$ one after another and performs the following steps:

(i) initialize a recent DSS by a single edge,
(ii) consecutively test the current vertex whether it still belongs to the recent DSS,
(iii) if this is not the case then try to adjust the parameters of the DSS in such a way that it contains all previous vertices, i.e. from the beginning of the recent DSS up to the preceding vertex, as well as the current vertex, and
(iv) close the recent DSS and generate a DSS termination message if no adjustment is possible.

The algorithm *DSS_Perimeter* is shown in Fig. 6. The algorithm uses variables:

1. *DIR*[1:2] for an array containing both edge directions allowed for the recent DSS,
2. $StartP$, $StartN$, $EndP$, $EndN$ for starting and end points of the positive and negative tangents (each vertex is described by a data structure (*int x*, *y*;) containing its vertex integer coordinates),
3. $Vertex$ takes the end vertex of the previous DSS as the start vertex of the new DSS, and it is described by a similar data structure,
4. $tang$ for a tangential vector specifying the direction of the tangents of the actual DSS and described by a similar data structure, where $tang.x = a$ and $tang.y = b$,

```
program DSS_Perimeter (v[1:n]; Perim)
  Begin
    N_DSS:=-1;              +++ not zero, index of first DSS
    Perim:=0.0;
    for i:=2 to n do        +++ v[n]=v[1]
    begin
      dir:=F(v[i]-v[i-1]);  +++ direction of last step specified by F( )
      If (i=2) then BRK:=1; +++ for initialization of parameters at start
      else BRK:=DSS_Cont(v[i], dir);            +++ call of DSS_Cont( )
      If (BRK>0) then       +++ an end vertex of a DSS is found
      begin
        If (i>2) then
          Perim:=Perim+Distance(Vertex,v[i-1]); +++ Euclidean distance
        StartN:=StartP:=v[i-1]; EndN:=EndP:=v[i];
        tang:=v[i]-v[i-1];  +++ vector subtraction
        DIR[1]:=dir; DIR[2]:=-1;                +++ DIR[2] is yet unknown
        HN:=0;              +++ HN stands for h(x,y)
        Vertex:=v[i-1];
        N_DSS:=N_DSS+1;
      end                   +++ of "If (BRK>0)"
    end                     +++ of "for...do"
    N_DSS:=N_DSS+1;
    Perim:=Perim + Distance(Vertex,v[i]);
  End.                      +++ of "DSS_Perimeter"
```

Fig. 6. Estimation of the perimeter based on DSS segmentation of a frontier.

5. *HN* for the integer value of $h(x, y)$ of the left side of the linear inequality of the half-plane based on the negative tangent of the DSS while *HN* is non-negative for the vertices of the DSS, and
6. *N_DSS* for the index (label) of the DSS's just detected, i for the index of the current vertex, *dir* for the direction of the last edge, $dir \in \{0, 1, 2, 3\}$ and *BRK* as a flag for starting the next DSS if $BRK = 1$.

The integer function *DSS_Cont* repeatedly called in the algorithm *DSS_Perimeter* has as *input* the current vertex *Point* described by the data structure $(Point.x, Point.y)$ containing the coordinates of *Point* and the direction *dir* of the last edge whose endpoint is *Point*.

The *output* is equal to 0 if *Point* belongs to the recent, eventually adjusted DSS, or it is equal to 1 if there exists no DSS containing the current point *Point* and all previous points. The variables are as in algorithm *DSS_Perimeter*. The algorithm for the integer function *DSS_Cont* is sketched in Fig. 7.

```
function DSS_Cont (Point, dir)
  Begin
    If (DIR[2]<0 AND dir=DIR[1]) then +++ next to initial direction
    begin
      EndN:=EndP:=Point;              +++ for EndN and EndP see Figure 3
      return 0;
    end
    If (DIR[2]>=0 AND dir<>DIR[1] AND dir<>DIR[2]) then return 1;
                                      +++ a third direction is prohibited
    If (DIR[2]<0 AND dir<>DIR[1]) then DIR[2]:=dir;
                                      +++ a second direction is found
    switch(dir)        +++ updating of the value of the left side of
    begin              +++ the inequality of the negative tangent
      case 0: HN:=HN + tang.y; break; +++ HN stands for h(x,y)
      case 1: HN:=HN - tang.x; break;
      case 2: HN:=HN - tang.y; break;
      case 3: HN:=HN + tang.x; break;
    end                               +++ of "switch"
    If (HN>0 AND HN<abs(tang.x)+abs(tang.y)-1) then
      return 0;                       +++ allowed value of HN
    If (HN<-1 OR HN>abs(tang.x)+abs(tang.y)) then
      return 1;                       +++ a non-repairable value
    If (HN=0) then                    +++ vertex on the negative tangent
    begin  EndN:=Point; return 0;
    end                               +++ of "If (HN=0)"
    If (HN=abs(tang.x)+abs(tang.y)-1) then +++vertex on positive tangent
    begin  EndP:=Point; return 0;
    end                               +++ of "If (HN=abs...)"
    If (HN=-1) then
              +++ vertex is negative outlier, DSS is being adjusted
    begin
      EndN:=Point;  StartP:=EndP;  tang:=Point-StartN;
                                      +++ vector subtraction
      HN:=0; return 0;
    end                               +++ of "If (HN=-1)"
    If (HN=abs(tang.x)+abs(tang.y)) then
              +++ vertex is positive outlier, DSS is being adjusted
    begin
      EndP:=Point;  StartN:=EndN;  tang:=Point-StartP;
                                     +++ vector subtraction
      HN:=(Point.x-StartN.x)*tang.y-(Point.y-StartN.y)*tang.x; return 0;
    end                               +++ of "If (HN=abs...)"
  End.                                +++ of "DSS_Cont"
```

Fig. 7. If the DSS is extendable then its parameters a, b, c are adjusted if necessary (with $a = tang.x$ and $b = tang.y$).

4 MLP approximation

This section discusses a convergent estimator for the perimeter of a planar bounded set S where an open r-boundary is assumed to act as a Zeno representation of S. The estimator is based on a minimum-length polygon approximation of such a digital curve.

4.1 Principle of MLP calculation

Originally the MLP method [23, 24, 25] has been defined using the notion of a grid continuum. However, the notion of an open r-boundary is suitable as well. The *MLP estimator* of the perimeter of a bounded, closed subset S of the euclidean plane is the length of the *minimum length polygon* (MLP), lying completely in the open r-boundary $O_r(S) \setminus (I_r(S) \cup \partial O_r(S))$ of set S and circumscribing $\partial I_r(S)$. The MLP is uniquely defined [24]. We will specify a linear-time MLP algorithm for the MLP construction.

With respect to the multigrid digitization approach we have to ensure that for $r \geq r_0$ the set S is such that $I_r(S)$ is non-empty and simply-connected. If the Chebyshev-Hausdorff distance between $\partial I_r(S)$ and $\partial O_r(S)$ exceeds $1/r$ then there are different options: a subset of $O_r(S) \setminus (I_r(S) \cup \partial O_r(S))$ of set S can be constructed to be taken as an open r-boundary of S (with distance equal to $1/r$), or the MLP algorithm can be modified in such a way that also general situations such as "peaks" or "cavities" (see Fig. 4) are covered. We take the second option.

Our MLP algorithm is defined identically for different values of r which allows us (as in case of the DSS algorithm) to suppress r in the rest of this section. For the perimeter estimation problem the scaling factor r has to be treated properly in the final multi-grid approximation algorithm.

For describing a method of finding the MLP we need a few auxiliary definitions. Consider the sequence of vertices which occur during tracing a frontier $\partial I(S)$ or $\partial O(S)$ of the given set S. The tracing is always supposed to run in positive direction of rotation, i.e. counterclockwise in a coordinate system where the positive y-axis points upwards and the positive x-axis points to the right, or clockwise in a coordinate system (often used in image processing) where the positive y-axis points downwards and the positive x-axis points to the right.

A vertex on $\partial I(S)$ or $\partial O(S)$ is a *convex vertex* if the frontier makes a positive turn at this point, i.e. the angle between an oriented edge and the positive x-axis increases. A vertex is a *concave vertex* in case of a negative turn, and a *collinear vertex* in case of a zero turn.

To recognize the sign of a turn it is common to use the determinant value

$$det(v_1, v_2, v_3) = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1) \qquad (7)$$

for three-vertex sequences $v_i = (x_i, y_i)$, for $i = 1, 2, 3$. The sign of the determinant value corresponds to the sign of the turn in any ortho-normal coordinate system.

Our algorithm traces frontiers $\partial I(S)$ or $\partial O(S)$, detects its vertices where a non-zero turn takes place and puts their integer vertex coordinates into a list. The vertex integer coordinates of two subsequent vertices, with indices i and $i+1$, satisfy the condition $|x_{i+1} - x_i| + |y_{i+1} - y_i| = 1$. Each entry in the list, briefly called *positive* or *negative vertex*, contains besides the coordinates also the sign of the turn as explained above. The positive vertices are the convex ones, and negative vertices are concave.

It is known [24] that only the positive vertices of $\partial I(S)$ may become vertices of the MLP, or negative vertices of $\partial O(S)$. For $\partial I(S)$ and $\partial O(S)$ we have a Chebyshev-Hausdorff distance of one in the integer vertex coordinate system. It is easy to see that there exists a mapping from the set of negative vertices of $\partial I(S)$ to that of $\partial O(S)$ such that each negative vertex of $\partial O(S)$ corresponds to at least one negative vertex of $\partial I(S)$. The algorithm starts with putting all vertices of $\partial I(S)$ into one list. Then we replace each negative vertex of $\partial I(S)$ in this list by its corresponding negative vertex of $\partial O(S)$ simply by modifying the coordinates by ± 1 according to the direction of the incident edges. Now the list contains all vertices which may become vertices of the MLP. It is important that the vertices are ordered according to the order of the vertices of $\partial I(S)$. Thus the ordered vertices may be processed one after another.

A vertex of the list may become an MLP-vertex if the following conditions are fulfilled. Suppose we already know that the ith vertex v_i of the list is an MLP-vertex. Another vertex v_j, with $j > i$, may belong to the MLP if all positive vertices v_k^+ with $i < k < j$ lie on the positive side of (v_i, v_j) or are collinear with it, i.e. $det(v_i, v_k^+, v_j) \geq 0$ holds. Similarly, all negative vertices v_l^- with $i < l < j$ must lie on the negative side of (v_i, v_j) or be collinear with it. Otherwise the pair (v_i, v_j) would cross either $\partial I(S)$ or $\partial O(S)$ which is not allowed to happen. Suppose there exist two vertices – a positive one denoted by v^+ and a negative one denoted by v^- – both satisfying the above conditions. When testing a vertex v from the list whether it may become an MLP vertex, the following three situations may occur:

1. v lies on the positive side of the pair (v_i, v^+), i.e. $det(v_i, v^+, v) \geq 0$;
2. v lies on the negative side of the pair (v_i, v^+) or is collinear with it and at the same time it lies to the positive side of (v_i, v^-) or is collinear with it;
3. v lies on the negative side of the pair (v_i, v^-).

It is easy to see that in the case (2) vertex v becomes a candidate for MLP and must replace either v^+ or v^- corresponding to the sign of v. It has been proven [24] that in the case (1) vertex v^+ becomes the next MLP vertex. Similarly, in the case (3) v^- becomes the next MLP vertex. The described analysis holds in a trivial way also in the case when either v^+ or v^- or both of them coincide with v_i. Thus it is possible to start with a vertex v_1 which is known to be an MLP vertex, to set v^+ and v^- equal to v_1 and then to test all subsequent vertices as described above. As soon as the next MLP vertex is detected, it becomes the next starting vertex and so on.

4.2 MLP Algorithm

As in case of the DSS algorithm we assume that the *input* of the MLP algorithm is a sequence of n vertices $v[1:n]$ on a frontier of a simply-connected isothetic polygon, but this time $I(S)$, each vertex being described by a data structure $(v[i].x, v[i].y)$ containing the integer vertex coordinates of the ith vertex $v[i]$. Since we assume a frontier, $v[n]$ must be equal to $v[1]$. For the MLP algorithm the *output* is a value $Perim$ which is the MLP estimate of the perimeter of set S.

```
program MLP_Perimeter (v[1:n]; Perim)
Array of points Turn[1:NV];
Array of points MLP[1:NM];
Begin                                   +++ ****************************
  call FindTurns(v; Turn; Nvert);       +++ puts turns of v[] into Turn[]
  Perim:=0.0;
  MLP[1]:=Turn[1];
  iM:=iNeg:=iPos:=1;                    +++ fist vertex belongs to MLP
  for iV:=2 to Nvert+1 do               +++ ============================
  begin
   If iV<=Nvert then iP:=iV;
   else iP:=1; +++ again to the starting vertex
   If det(MLP[iM], Turn[iPos], Turn[iP])>0 then
   begin   +++ iP lies on positive side +++ --------
      iM:=iM+1;
      MLP[iM]:=Turn[iPos];
      iV:=iNeg:=iPos;
      Perim:=Perim+Distance(MLP[iM-1],MLP[iM]);
    end    +++ on positive side         +++ --------
    else
      If det(MLP[iM], Turn[iNeg], Turn[iP])>=0 then
        begin                           +++ iP in sector ----------------
          If Turn[iP].SIGN>0 then iPos:=iP;
          else iNeg:=iP;
        end                             +++ in sector    ----------------
      else
      begin                             +++ iP lies on negative side ----
        iM:=iM+1;
        MLP[iM]:=Turn[iNeg];
        iV:=iPos:=iNeg;
        Perim:=Perim+Distance(MLP[iM-1],MLP[iM]);
      end                               +++ on negative side         ----
  end      +++ for iV:=2 to ...         +++ ============================
End;       +++ of program MLP_Perimeter +++ ****************************
```

Fig. 8. Estimation of the perimeter based on MLP calculation within the closure of an open boundary.

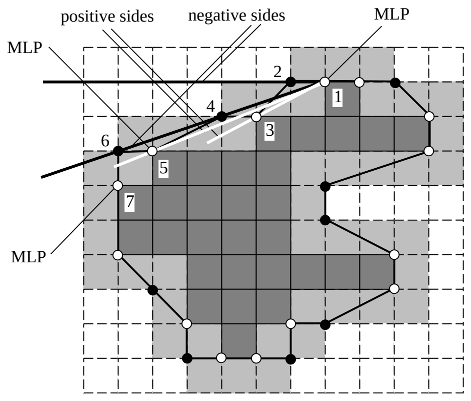

Fig. 9. Finding the next MLP vertex: the numbers indicate subsequent vertices in the list $Turn[1:NV]$; 1 is the already known MLP vertex; 3 and 5 are subsequent positions of WHITE; 2,4 and 6 are that of BLACK; the vertex 7 does not lie between the negative (black line) and positive (white line) sides of the sector (6,1,5), therefore 5 is the next MLP vertex.

During the tracing of $\partial I(S)$ all its vertices where a turn takes place must be detected and their coordinates and signs of turns must be put into the list of vertices. The algorithm then changes the coordinates of the concave vertices as explained in Section 4.1. The list of these vertices is represented by the array $Turn[1:NV]$. The first entry in $Turn[1:NV]$ corresponds to the uppermost-leftmost corner of $I(S)$ which is always an MLP vertex.

The algorithm consists in repeating the procedure of finding the next MLP vertex when an MLP vertex is already known. The first MLP vertex is the first vertex in $Turn[1:NV]$, as explained above. The algorithm uses two movable vertices called the WHITE and the BLACK *beetles* to label the vertices which are candidates for the next MLP vertex (see Fig. 9). At start both WHITE and BLACK are set equal to the known MLP vertex. The algorithm reads the vertices of $Turn[1:NV]$ one after another and steps ones again to the first vertex after having reached the last one. The algorithm performs for every vertex v_T the test described in Section 4.1. After the test either one of the beetles moves to v_T or one of them becomes the next MLP vertex v_M. In the latter case the beetles and the current vertex v_T move to v_M. In the next step of the loop v_T moves to the vertex next after v_M. The algorithm repeats the procedure until the current vertex reaches again the first vertex in $Turn[1:NV]$.

The algorithm *MLP_Perimeter* is shown in Fig. 8. Practically its input can be identified with that of *DSS_Perimeter*: a sequence $v[1:n]$ of n vertices on the frontier $\partial C(S)$. However, according to our digitization model we will use $\partial I(S)$ as input in our experiments reported later. The output is the value $Perim$ which is the MLP estimate of the perimeter of S. The algorithm uses the following variables:

1. $Turn[1:NV]$ is an array of vertices as described above. Each entry contains the coordinates and the sign as a value +1 or -1,
2. $MLP[1:NM]$ is an array of vertices of the MLP,
3. iV, iP and iM are indices of elements in these arrays,
4. *Nvert* is the number of vertices found by the procedure "FindTurns" as described above,
5. *iNeg* and *iPos* are indices pointing to the "beetles" BLACK and WHITE, correspondingly,
6. the function *det( )* calculates the determinate value for three points as described above, and
7. the function *Distance( )*, the same as in DSS_Perimeter, calculates the Euclidean distance between two points.

Figure 10 illustrates how the algorithm proceeds, and how it is also able to deal with "cavities" A (width 1) and B (width 2) where examples were already

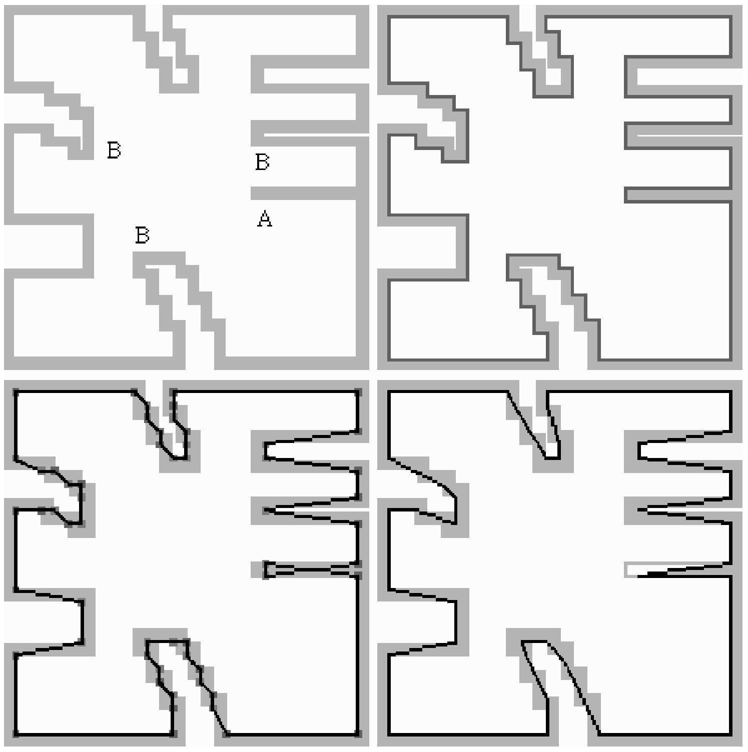

Fig. 10. Screen shots illustrating subsequent processing steps of the MLP algorithm where an arbitrary digital curve is assumed as input also containing "cavity" situations.

illustrated in Fig. 4. Top left shows an input example of a boundary of an isothetic polygon where the shaded "band" illustrates squares around the given boundary. Top right shows the given boundary as a dark curve, and this curve may be interpreted to be the boundary of a set $I(S)$. Bottom left shows the curve connecting all positive vertices (convex vertices) on this boundary and negative vertices in the order as they are used in the MLP algorithm. Bottom right finally shows the resulting MLP, also for "cavity" situations such as A and B in the input sequence.

5 Experimental evaluation

We report about experiments using the six sets S shown in Fig. 11: the function graph of the *sinc function*

$$y = sin(16\pi \cdot x)/(64\pi \cdot x)$$

within a bounded interval symmetric to the y-axis, a square rotated by $45°$, a square rotated by $22.5°$, a halfmoon generated by two overlapping circles of identical size, a circle, and the yin-part in the Chinese yinyang symbol. All sets have been digitized for (at least) all grid resolutions between $r = 32$ and $r = 1024$, including these two values.

A closed square ("pixel") is contained in $C(S)$ iff the midpoint of the corresponding square is in S. The frontier $\partial C(S)$ is used as input for the DSS algorithm. In our experiments we use an approximative scheme for $I(S)$: a closed square is contained in $I(S)$ iff all four vertices of the square are in S ("four-vertex scheme"). Note that our MLP algorithm only requires that $\partial I(S)$ is available as input, and $O(S)$ needs not to be calculated.

For the resulting isothetic polygons the DSS partition and the MLP were calculated (see positions of sets in Fig. 12) allowing to compute the DSS estimate

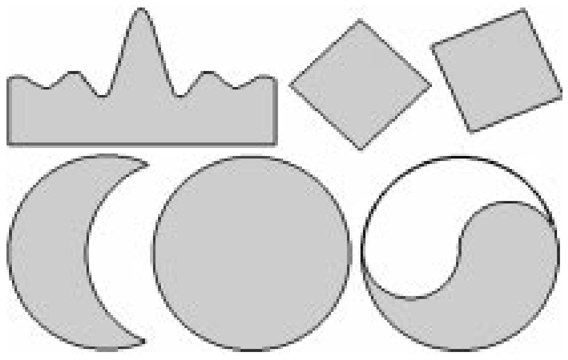

Fig. 11. The six sets used for multigrid digitizations.

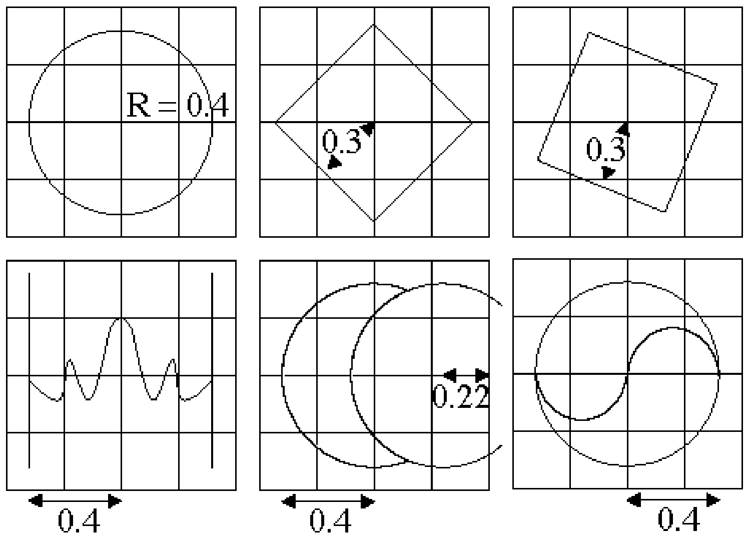

Fig. 12. Geometric specification of the positions of the used 6 sets within an area of size $1unit \times 1unit$.

and the MLP estimate of the perimeter of region S with respect to grid resolution r. Of course, $\partial C(S)$ may also be used as input for the MLP algorithm, and we call the result the *MLP_C perimeter estimate*, or $\partial I(S)$ as input for the DSS algorithm, and the result is the *DSS_I perimeter estimate*.

Let $N_{C(S)}$ be the number of vertices of $\partial C_r(S)$ (the input sequence of the DSS algorithm), let $N_{I(S)}$ be the number of vertices of $\partial I_r(S)$ (the input sequence of the MLP algorithm), let N_{DSS} be the number of vertices of the calculated DSS polygon, and let N_{MLP} be the number of vertices of the calculated MLP. The *errors* E_{DSS} and E_{MLP} are the length errors in percent compared to the true perimeter of set S (length of the curve defining the boundary of S). The *effectiveness of the approximation* (as defined in [24]) is $\theta_{DSS} = E_{DSS} \cdot N_{DSS}$ or $\theta_{MLP} = E_{MLP} \cdot N_{MLP}$, which specifies an interesting trade-off measure: this product of error times number of segments informs about the efficiency of the convergence. If this product decreases faster or increases slower for algorithm A in comparison to a second algorithm B then algorithm A is more efficient in achieving reduced errors without generating too many new segments. [2] Finally, we also measure the computing time T_{DSS} and T_{MLP}. In our tables time is specified in multiples of 10^{-3} seconds. Tables 1-6 show selected results (for a few values of r) for the six curves.

[2] Smaller values of this measure characterize a "more efficient" approximation, ie. the name "inverse effectiveness measure" might be more appropriate. However, we follow [24] because the given name is also reasonable.

Size	64x64	128x128	256x256	512x512	1024x1024	2048x2048
$N_{C(S)}$	122	238	474	954	1922	3838
N_{DSS}	18	29	46	74	121	187
$N_{I(S)}$	106	230	474	946	1906	3834
N_{MLP}	31	47	80	124	199	330
E_{DSS}	5.987	5.433	3.061	2.466	1.605	1.432
E_{MLP}	12.541	9.718	6.006	4.390	3.037	2.018
θ_{DSS}	1.0777	1.5755	1.4080	1.8249	1.9418	2.6776
θ_{MLP}	3.8877	4.5675	4.8045	5.4442	6.0427	6.6600
T_{MLP}	0.538	1.024	2.048	4.199	8.398	17.206
T_{DSS}	1.216	2.381	4.710	9.421	19.254	38.099

Table 1. Parameters and measurements for the yin curve of the Yinyang symbol.

Size	64x64	128x128	256x256	512x512	1024x1024	2048x2048
$N_{C(S)}$	68	148	304	584	1168	2336
N_{DSS}	20	23	39	54	91	139
$N_{I(S)}$	56	136	288	584	1160	2332
N_{MLP}	30	40	68	98	156	242
E_{DSS}	1.340	1.917	0.396	0.086	0.009	0.041
E_{MLP}	9.327	3.141	1.591	0.735	0.311	0.084
θ_{DSS}	0.2681	0.4409	0.1546	0.0466	0.0084	0.0576
θ_{MLP}	2.7981	1.2565	1.0820	0.7208	0.4848	0.2028
T_{MLP}	0.294	0.589	1.331	2.560	5.121	10.242
T_{DSS}	0.998	1.971	3.891	7.988	15.158	29.906

Table 2. Parameters and measurements for the sinc curve.

In these tables, the values in row *Size* correspond to the resolution r, ie. size is equal to $r \times r$, and the case $r = 1$ is shown in Fig. 12. The errors are given in percent. The effectiveness is scaled by factor 10^{-2}. Time is specified in multiples of 10^{-3} seconds.

Table 7 illustrates by a few numerical values that measurements such as errors will not decrease monotonously with an increase in resolution. This is also illustrated in Fig. 13 for the resulting errors in case of the circle, and in Fig 14 for the resulting effectiveness values of the approximation, also in case of the circle. This shows that tables such as Tables 1-6 showing a few isolated measurements only may be absolutely misleading! They provide a “first idea” about the behavior of the algorithms, but should not be used for generalizations.

Generalizations require statistical evaluations. Due to the inhomogeneous

Size	64x64	128x128	256x256	512x512	1024x1024	2048x2048
$N_{C(S)}$	124	236	476	964	1924	3836
N_{DSS}	16	28	36	68	104	168
$N_{I(S)}$	116	244	484	956	1916	3844
N_{MLP}	32	48	64	104	168	272
E_{DSS}	0.569	0.079	0.018	0.023	0.015	0.005
E_{MLP}	0.653	0.244	0.131	0.059	0.019	0.008
θ_{DSS}	0.0911	0.0222	0.0065	0.0159	0.0156	0.0084
θ_{MLP}	0.2091	0.1171	0.0837	0.0609	0.0327	0.0207
T_{MLP}	0.627	1.203	1.946	3.994	8.193	15.977
T_{DSS}	1.293	2.432	4.915	9.729	19.254	38.099

Table 3. Parameters and measurements for the circle.

Size	64x64	128x128	256x256	512x512	1024x1024	2048x2048
$N_{C(S)}$	212	428	860	1732	3468	6940
N_{DSS}	4	4	4	4	4	4
$N_{I(S)}$	204	420	852	1724	3460	6932
N_{MLP}	8	8	8	8	8	8
E_{DSS}	0.495	0.546	0.559	0.102	0.102	0.103
E_{MLP}	2.720	1.642	1.102	0.372	0.238	0.170
θ_{DSS}	0.0198	0.0218	0.0224	0.0041	0.0041	0.0041
θ_{MLP}	0.2176	0.1313	0.0882	0.0298	0.0190	0.0136
T_{MLP}	0.538	1.075	2.048	4.199	8.398	16.387
T_{DSS}	1.357	2.637	5.274	10.548	21.098	42.196

Table 4. Parameters and measurements for the square rotated 45 degrees.

increase in error and effectiveness values we use *sliding means* for graphical representations of results. These means are the arithmetic averages of 64 values, 32 values on the left of the recent position, the value at the recent position, and 31 values on the right of the recent position. In Fig. 13 and Fig. 14 these sliding means form curves.

Figures 13 and 14 also show results for the MLP_C estimates where the input polygons $I(S)$ have been replaced by the larger input polygons $C(S)$. This is just a minor increase of the input polygons but it illustrates the sensitivity of the MLP algorithm. Similar results have been obtained if the input $C(S)$ of the DSS algorithm is replaced by $I(S)$ or $O(S)$. The errors of the DSS_I estimate are slightly larger than the errors of the MLP_C estimate in case of the circle. However, for the halfmoon boundary the MLP_C estimates are best (ie. smallest errors with respect to the sliding means).

The computing time (see Fig. 15) is more regular in its behavior: the only

Size	64x64	128x128	256x256	512x512	1024x1024	2048x2048
$N_{C(S)}$	116	236	468	932	1876	3764
N_{DSS}	5	5	5	6	5	5
$N_{I(S)}$	116	228	460	940	1876	3764
N_{MLP}	20	16	20	24	32	40
E_{DSS}	0.733	0.088	0.113	0.246	0.005	0.053
E_{MLP}	3.554	1.535	0.835	0.286	0.260	0.087
θ_{DSS}	0.0367	0.0044	0.0057	0.0148	0.0002	0.0026
θ_{MLP}	0.7107	0.2457	0.1671	0.0687	0.0830	0.0348
T_{MLP}	0.461	0.870	1.843	3.789	9.832	17.206
T_{DSS}	1.152	2.304	4.506	9.012	18.025	36.051

Table 5. Parameters and measurements for the square rotated 22.5 degrees.

Size	64x64	128x128	256x256	512x512	1024x1024	2048x2048
$N_{C(S)}$	124	236	476	960	1924	3836
N_{DSS}	16	29	39	71	104	172
$N_{I(S)}$	112	240	476	952	1912	3836
N_{MLP}	32	50	76	116	186	282
E_{DSS}	1.632	1.062	0.460	0.168	0.036	0.047
E_{MLP}	6.470	3.141	1.527	0.711	0.310	0.107
θ_{DSS}	0.2611	0.3079	0.1793	0.1196	0.0369	0.0808
θ_{MLP}	2.0703	1.5707	1.1609	0.8244	0.5771	0.3023
T_{MLP}	0.602	1.152	2.202	4.301	8.603	16.796
T_{DSS}	1.267	2.458	4.864	9.626	19.254	38.099

Table 6. Parameters and measurements for the boundary of the halfmoon.

Size	64x64	65X65	66x66	67x67	68x68	69x69	70x70	71x71	72x72
$N_{C(S)}$	68	84	68	84	80	80	88	92	88
N_{DSS}	20	18	18	19	20	18	17	18	18
$N_{I(S)}$	56	68	72	76	68	72	76	84	76
N_{MLP}	30	30	30	32	26	30	34	32	32
E_{DSS}	1.340	0.036	5.355	2.576	0.744	2.112	1.442	2.301	0.877
E_{MLP}	9.327	8.562	8.434	7.508	6.564	6.729	5.932	7.798	9.216
θ_{DSS}	0.2681	0.0064	0.9639	0.4895	0.1489	0.3802	0.2452	0.4141	0.1579
θ_{MLP}	2.7981	2.5685	2.5303	2.4026	1.7067	2.0186	2.0167	2.4955	2.9492
T_{DSS}	0.294	0.312	0.317	0.348	0.313	0.345	0.350	0.383	0.360
T_{MLP}	0.998	1.053	1.003	1.059	1.088	1.090	1.106	1.122	1.138

Table 7. Stepwise increases of resolution leads to non-monotonously increases in measurements, here demonstrated for the sinc function.

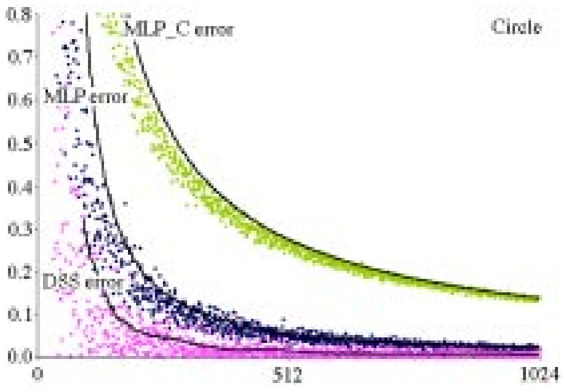

Fig. 13. Distribution of DSS and MLP errors in case of the circle. The input polygon $C(S)$ was also taken for MLP_C (instead of $I(S)$). The lines show the sliding means, always calculated for 64 values.

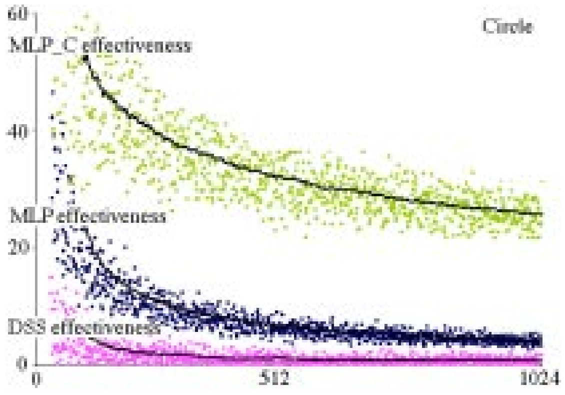

Fig. 14. Distribution of DSS and MLP trade-off values in case of the circle. The input polygon $C(S)$ was also taken for MLP_C (instead of $I(S)$). The lines show the sliding means, always calculated for 64 values.

difficulty is to measure these small time values accurately by a sufficient large number of repetitions of the same program run.

Figure 16 summarizes our experiments. It shows the averaged sliding means of error values E_{DSS} or E_{MLP}, and of sliding means of effectiveness values θ_{DSS} and θ_{MLP}, where each value at resolution r is generated by averaging of the results for the six examples of sets at this resolution r. The tangential geometric convergence of the yin curve has a special impact on slowing down numerical convergence to the true value. For that reason curves only averaging five sets

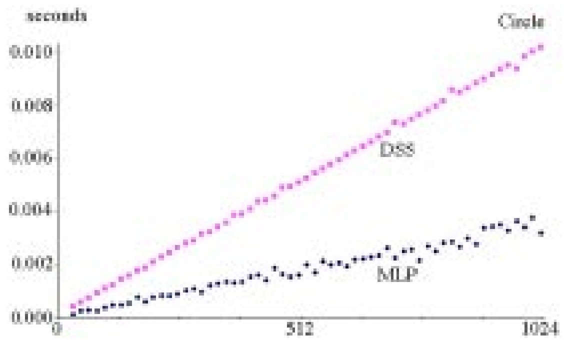

Fig. 15. Computing time of DSS and MLP algorithms in case of the circle. The computing time of the MLP_C algorithm is about the same as in the MLP case.

only (without yin curve) are also included in Fig. 16.

6 Conclusions

Typically the DSS errors in estimating the true perimeter have been at 0.1% or lower for grid resolutions of 600 or higher, also for non-convex sets such as the halfmoon or the sinc-curve. The errors were slightly larger for the yin-part of the Yinyang symbol. The DSS algorithm shows in general faster convergence and a better efficiency with respect to our trade-off measure, see Fig. 16.

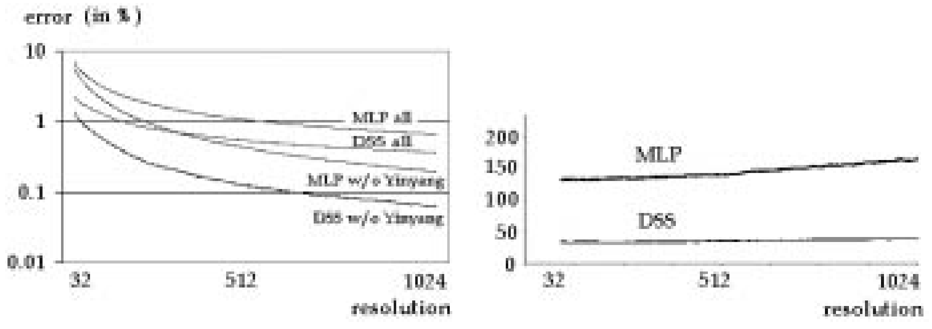

Fig. 16. These diagrams illustrate the behavior of both methods for the six selected sets as shown in Fig. 11 and Fig. 12 which have been digitized for grids of resolution 32, 33, ..., 1024. Left: The sliding means (averages of 64 values) of errors are shown for a logarithmic scale, with or without the results for the Yinyang region. Right: Effectiveness diagram.

The MLP is uniquely defined [24]. The DSS partition may (slightly) change depending upon the starting point and the orientation.

In our implementation, the DSS algorithm needs about 2-3 times more computing time compared to the MLP algorithm, and this is about the same for the different test sets. The times for generating the input data, ie. frontiers of $C(S)$ or $I(S)$, are not counted in this time comparison. Algorithms for calculating maximum length DSS's have been considered by many authors for more than 20 years. Calculations of MLP's could be still an interesting subject for discussing alternative algorithms.

Different digitization schemes or input noise should also be studied for the DSS and MLP method. For example, the four-vertex scheme might be a reason that MLP "typically" approximates the curve length from below but DSS results "typically" oscillate (i.e. there are both positive and negative differences) around the true value. For example, Fig. 13 shows that the MLP_C estimate (input $I(S)$ replaced by $C(S)$ leads to an increase in errors in case of a circle. But the situation may change for non-convex curves, as we found out for the half-moon where the MLP_C estimate is best.

Acknowledgement

The authors are very thankful for a fruitful collaboration with *Vladimir Kovalevsky* (Berlin) which led to the paper [10]. The reported DSS and MLP algorithms are basically citations from [10].

References

1. E. Artzy, G.T. Herman: Boundary detection in 3 dimensions with a medical application. *Computer Graphics* **15** (1981) 92–123.
2. B.H. Bunch: *Mathematical, Fallacies and Paradoxes.* Van Nostrand, New York (1982).
3. E. Creutzburg, A. Hübler, V. Wedler: On-line Erkennung digitaler Geradensegmente in linearer Zeit. in: Proc. GEOBILD'82, Wiss. Beiträge der FSU Jena (1982) 48–65.
4. E. Creutzburg, A. Hübler, O. Sýkora: Geometric methods for on-line recognition of digital straight segments. *Computers and Artificial Intelligence* **7** (1988) 253–276.
5. A. Hübler, R. Klette, K. Voss: Determination of the convex hull of a finite set of planar points within linear time. *EIK* **17** (1981) 121–139.
6. C. Jordan: Journal de Mathématiques, 4^e série, T. 8 (1892) 77.
7. B. Kamgar-Parsi, B. Kamgar-Parsi: Evaluation of quantization error in computer vision. *IEEE Trans. PAMI* **11** (1989) 929-940.
8. R. Klette: M-dimensional cellular spaces. CAR-TR-6, Univ. of Maryland, Center for Automation Research, College Park (1983).
9. R. Klette, A. Rosenfeld, F. Sloboda (eds.): *Advances in Digital and Computational Geometry.* Springer, Singapore (1998).
10. R. Klette, V. Kovalevsky, B. Yip: On the length estimation of digital curves. Proc. *Vision Geometry VIII*, SPIE Volume **3811**, Denver, ??-?? July (1999) ??–??.

11. R. Klette, J. Žunić: Multigrid convergence of calculated features in image analysis. CITR-TR 51, CITR Tamaki, Auckland University, October 1999.
12. R. Klette, F. Wu: Evaluation of surface calculation algorithms for 3D volumes. CARS'1999, Paris, June 23-26, 1999 , Elsevier, Intern. Congress Series **1165** (1999) ??–??.
13. V. Kovalevsky: Finite topology as applied to image analysis. *CVGIP* **46** (1989) 141–161.
14. V. Kovalevsky: New definition and fast recognition of digital straight segments and arcs. Proc. 10th ICPR, Atlantic City, June 16-21 (1990) 31–34.
15. V. Kovalevsky, S. Fuchs: Theoretical and experimental analysis of the accuracy of perimeter estimates. in: *Robust Computer Vision* (W. Förstner, S. Ruwiedel, eds.), Wichmann, Karlsruhe (1992) 218–242.
16. V. Kovalevsky: Finite topology and image analysis. in: *Image Mathematics and Image Processing* (P. Hawkes, ed.), Advances in Electronics and Electron. Physics **84**, Academic Press (1992) 197–259.
17. V. Kovalevsky: A new concept for digital geometry. in: *Shape in Picture* (Y.-L. O et al., eds.) Springer, Berlin (1994) 21–36.
18. V. Kovalevsky: Applications of digital straight segments to economical image encoding. in: *Discrete Geometry for Computer Imagery* (E. Ahronovitz, Ch. Fiorio, eds.), LNCS 1347, Springer, New York (1997).
19. Z. Kulpa: On the properties of discrete circles, rings, and disks. *CGIP* **10** (1979) 348–365.
20. S.R. Kulkarni, S.K. Mitter, T.J. Richardson, J.N. Tsitsiklis: Local versus nonlocal computation of length of digitized curves. *IEEE Trans. Pattern Analysis and Machine Intelligence* **16** (1994) 711–718.
21. W. Rinow: *Topologie.* Deutscher Verlag der Wissenschaften, Berlin (1975).
22. A. Rosenfeld: Digital straight line segments. *IEEE Trans. Comp.* **C-23** (1974) 1264–1269.
23. F. Sloboda, B. Zaťko, P. Ferianc: Minimum perimeter polygon and its application. in: *Theoretical Foundations of Computer Vision* (R. Klette, W.G. Kropatsch, eds.), Mathematical Research **69**, Akademie Verlag, Berlin (1992) 59-70.
24. F. Sloboda, B. Zaťko, J. Stoer: On approximation of planar one-dimensional continua. in: *Advances in Digital and Computational Geometry*, (R. Klette, A. Rosenfeld and F. Sloboda, eds.) Springer, Singapore (1998) 113–160.
25. F. Sloboda, B. Zaťko, R. Klette: On the topology of grid continua. Proc. *Vision Geometry VII*, SPIE Volume **3454**, San Diego, 20-22 July (1998) 52–63.
26. C.-H. Teh, R.T. Chin: On digital approximation of moment invariants. *Comp. Vis. Graph. Image Proc..* **33** (1986) 318-326.
27. A.W. Tucker: An abstract approach to manifolds. *Annals of Math.* **34** (1933) 191–243.
28. K. Voss: Digitalisierungseffekte in der automatischen Bildverarbeitung. *EIK* **11** (1975) 469–477.
29. K. Voss: *Discrete Images, Objects, and Functions in* Z^n. Springer, Berlin (1993).
30. M. Worring, A.W.M. Smeulders: Digitized circular arcs: Characterization and parameter estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence* **17** (1995) 587–598.
31. N. Yang, R. Klette: Linear time calculation of 2D shortest polygonal Jordan curves. in: *Image and Vision Computing New Zealand* (R. Klette, G. Gimel'farb, R. Kakarala, eds.), CITR Tamaki, Auckland (1998) 180–185.