On the Length Estimation of Digital Curves

Reinhard Klette[a], Vladimir Kovalevsky[b], and Ben Yip[a]

[a]CITR Tamaki, University of Auckland
Tamaki Campus, Building 731, Auckland, New Zealand

[b]Computer Science Department, University of Rostock
Albert-Einstein-Str 21, D-18059 Rostock, Germany

ABSTRACT

The paper details two linear-time algorithms, one for the partition of the boundary line of a digital region into digital straight segments, and one for calculating the minimum length polygon within a simple open boundary of a digital region. Both techniques allow the estimation of the length of digital curves or the perimeter of digital regions due to known multigrid convergence theorems. The algorithms are compared with respect to convergence speed and number of generated segments.

Keywords: Digital geometry, digital curves, cellular complexes, multigrid convergence, length estimation.

1. INTRODUCTION

Recent literature for image analysis still recommends different formulas combining numbers of local space configurations along the curve or along the boundary of a digital region for estimating the length of a digital curve, or of the perimeter of a digital planar object. One aim of the present paper is to draw (again) the attention to the fact that it is well-known for several years that these methods are imprecise[13,14,22,21] and, what is more important, their presicion cannot be improved by increasing the resolution of the digitization.

From the convergence point of view the estimation of a perimeter behaves essentially different as compared to the estimation of the area of a planar measurable region: the area may be estimated by means of counting the number of space elements such as grid points, or unit squares contained in the object.[17] This estimate converges to the true area for measurable regions in the Euclidean plane. Estimates of the length of a Jordan curve cannot be based on counts of space elements such as grid edges or grid diagonals. The scaled counts may not change at all when the grid resolution is increased, and may differ by up to 41% for different orientations of an otherwise identical curve before digitization.

This paper presents a comparison of two methods, abbreviated as **(i)** *DSS method* which is based on a partition of a digital curve into **d**igital **s**traight **s**egments,[3] and **(ii)** *MLP method* which is characterised by the calculation of the **m**inimum **l**ength **p**olygon in an open boundary of a digital region.[18] Both methods allow the calculation of curve length or perimeter, respectively, of digital curves or of simply-connected digital sets.

For both techniques it is known that the measured curve length converges towards the true value if an Euclidean convex region is digitized with increasing grid resolution.[8,19] These theorems comprise the theoretical fundamentals of this paper.

The paper presents a linear-time DSS algorithm[7] and a new linear-time MLP algorithm. For both methods further algorithms have been discussed elsewhere, for example three linear-time DSS algorithms[2] and basic ideas of MLP algorithms.[19,23] The paper also compares both methods with respect to speed of convergence and number of generated segments. The paper deals with closed-loop curves (perimeters of regions) only. However, the discussed algorithms may also be easily modified for open-loop curves.

2. FUNDAMENTALS

Digital regions are described in this paper by means of *abstract cellular complexes* (ACC's) $C = [E, B, dim]$. See, for example, 6,9,15 for definitions, theorems and results. E is the set of *elements* or *cells*, and B is an antisymmetric, irreflexive, and transitive binary relation called *bounding relation*. If $B(e', e'')$ then it is common to say that e' *bounds* e'', for $e', e'' \in E$. An integer valued function $dim(e)$ specifies the *dimension* of element $e \in E$. A *k-cell* is a cell of dimension k. A cell can only bound another cell of a greater dimension. If one cell bounds another one then they are called *incident with each other*. A cell is also incident with itself. Thus the incidence relation is symmetric, reflexive and not transitive. The transitive hull of the incidence relation is the *connectivity relation* thus defining the topological notion of *connected subsets* in ACC's.

We demonstrate in what follows that cellular complexes also allow to introduce grid continua[20] which have been used as the conceptual basis of the MLP method. The DSS method has been already discussed based on notations and results of ACC's.[6,8]

2.1. Boundary line and open boundary of a digital region

The intention of both methods is to be applied for the measurement of the length of an open-loop or closed-loop curve in image analysis. We consider closed-loop curves as boundaries of regions and open-loop curves as subsets of the boundaries. A *digital region* is a result of the digitization of an analog region. A *digital curve* is the boundary line (or a connected subset of a boundary line) of a digital region. For an exact specification of the notions of digital regions and their boundaries we start with citation of a few basic notions of the topology of ACC's.

A *subcomplex* $C' = (E', B', dim')$ of a given complex $C = (E, B, dim)$ is a complex whose set E' is a subset of E and the relation B' is the intersection of B with $E' \times E'$. The dimensions do not change, $dim'(e) = dim(e)$. Thus to define a subcomplex it suffices to define the corresponding subset E'. Therefore all subcomplexes of C may be regarded as subsets of E and it is always possible to apply operations of the set theory, such as union, intersection, complement etc. to subcomplexes of one and the same complex.

A subcomplex S of C is called *open in C* if for every cell e' of S all elements of C which are bounded by e' are also contained in S. We consider only finite sets. Therefore there exists the smallest open subset of C containing e, for each cell $e \in C$. It is called the *smallest open neighborhood* of e in C and will be denoted by $SON(e, C)$.

We consider a two-dimensional digital image as a *two-dimensional Cartesian ACC*. A two-dimensional Cartesian ACC[9] is a Cartesian product (cross product) of two one-dimensional ACC's which may be considered to be coordinate axes. It contains cells of dimensions 0, 1 and 2. We call the 2-cells *pixels*, the 1-cells *cracks* and the 0-cells *points*.

Graphically we represent pixels as squares, cracks as sides of squares and points as vertices of squares allowing us to visualise the bounding relation of an ACC: a crack bounds a pixel if the crack is represented as a side of the square representing the pixel, a point bounds a crack if the point is represented as an end point of the line segment representing the crack, and a point bounds a pixel if the point is represented as a vertex of the square representing the pixel. In a Cartesian complex the cells may be organized in rows and columns thus composing a regular grid. The grid may be drawn orthogonally. Note that this possible way (!) of representing Cartesian ACC's does not (!) imply that cells of an ACC would be subsets of the Euclidean plane. Cells of an ACC are not composed of any "smaller" objects, they are indivisible units of different categories where each category is defined by its dimension.

The main idea of using ACC's is to have a structure which contains a finite number of elements and therefore may be stored and processed by computers while possessing a topology in the classical sense[15] of this notion. An ACC fulfills this requirement while infinite topological spaces cannot be represented at elementary unit level by a computer.

A subcomplex S is *homogeneously two-dimensional* if any cell in S is either a 2-cell, or it bounds a 2-cell of S. A *digital region* is a connected, homogeneously two-dimensional subcomplex whose complement is also connected and homogeneously two-dimensional.

Let $S \subseteq C$ be a digital region. The *boundary* or *frontier $Fr(S, C)$ relative to C* is defined as in classical topology: it is the subset of all cells of C whose SON's intersect both S and its complement $C \setminus S$. The boundary of S may also be denoted as ∂S if there is no need to explicitly specify the complex C. It is easy to see that the SON of a pixel consists of the pixel itself. The SON of a crack contains the crack and the two pixels bounded by it. The SON

of a point contains the point itself and the four cracks and four pixels bounded by it. Thus the boundary of a digital region consists of cracks and points and contains no pixels.

The *topological interior* of a subcomplex $A \subseteq C$ is denoted by A°, defined by $A^\circ = A \setminus \partial A$. For any open subcomplex A it holds that $A = A^\circ$. The *topological closure* is denoted by $Cl(A)$, defined by $Cl(A) = A \cup \partial A$. A subcomplex A is *closed* if $A = Cl(A)$. Note that a digital region may be open, closed or neither open nor closed, and its boundary is not necessarily a subset of the digital region.

The problem under consideration consists in estimating the perimeter of a preimage of a digital region S while analyzing S. The estimate is expected to be "close" to the perimeter of the preimage of S which is assumed to be a region in the Euclidean plane having a Jordan curve as its boundary. To be more precise, the goal consists in ensuring multigrid convergence towards the true perimeter. *Multigrid digitization* is specified later on in Section 4 where a preimage is digitized into a digital region S.

The *boundary line* of a digital region S is defined as being a sequence of alternating cracks and points which covers all cells in ∂S and where each cell is always followed by a cell incident with it. The boundary line is a closed-loop polygonal curve. The boundary line may be subdivided into *digital straight line segments* (DSS's). The exact definition of a DSS is given in Section 2.2. The *DSS method* is defined by partitioning a boundary into digital straight line segments each of maximum length. Its result may vary with the chosen start position and orientation. See Fig. 1 for clockwise and counterclockwise partitions for of a digital region.

Typically, a DSS algorithm traces the boundary line crack by crack and subdivides it into maximum length digital straight segments. A linear-time off-line algorithm is contained in 3, and a linear-time on-line algorithm in 1. The algorithm presented in this paper follows 7. This algorithm detects for each maximum length DSS the coordinates of its end points and calculates the length of each DSS as the Euclidean distance between these points. The sum of the lengths of these DSS's is finally used as the *DSS estimate of the perimeter* of the given digital region.

Originally the MLP method[18–20] has been defined using the notion of a grid continuum. However, the notion of an open boundary of a digital region[8] is suitable as well. The *open boundary* of a region S is the set of all cracks and pixels whose topological closure intersects both the region S and the complement $C \setminus S$ (see Definition 4a in 12). This is in contrast to the boundary line which consists of points and cracks.

Two pixels e_1, e_2 are called *edge connected* if they are not identical and if there exists (exactly) one crack incident with both pixels. An open boundary is called *simple* if each 2-cell e in this open boundary is edge connected to

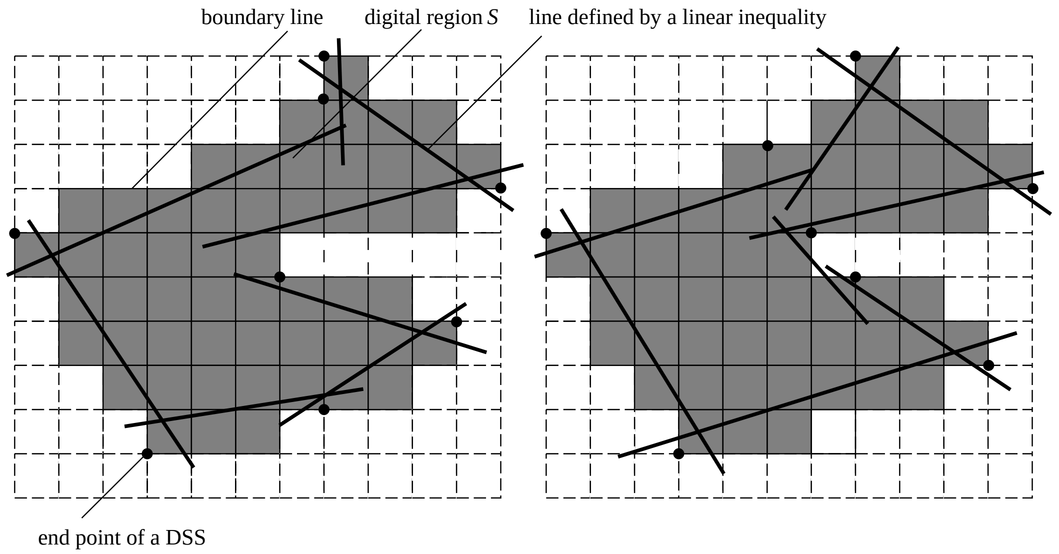

Figure 1. The partition of a boundary line into maximal DSS's may be calculated by identifying a sequence of DSS end points starting at the uppermost-leftmost point in S. Left: clockwise. Right: counterclockwise.

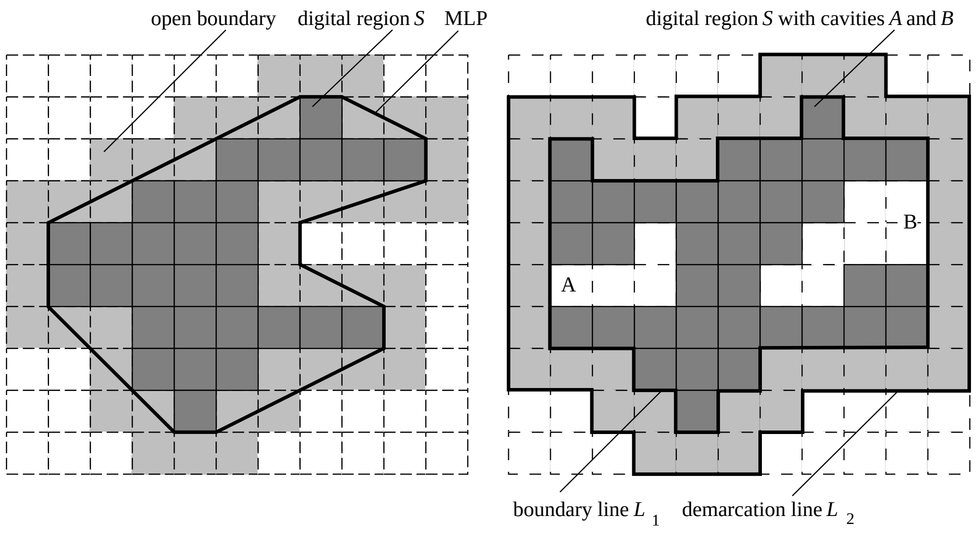

Figure 2. Left: An example of an open boundary and its minimum length polygon. Right: A digital region S with cavity A ("hole entrance of width 1"), cavity B (width 2) illustrating two types of problems which may arise, and possible boundary line L_1 and demarcation line L_2 of an expanded digital region.

exactly two further 2-cells in this open boundary. Cracks "between" two edge connected 2-cells of an open boundary are contained in this open boundary, and no further cracks.

A simple closed-loop one-dimensional grid continuum as defined in[18–20] corresponds to the closure of a simple open boundary. Any open boundary of a digital region contains (at least) one simple open boundary (Fig. 2, right). If the open boundary of a digital region S is homoeomorphic to the annulus, i.e. it is a closed loop, then it holds that all boundary cells of this open boundary may be traced by two sequences of alternating cracks and points, namely the boundary line L_1 of the given region and a second line L_2 which we call the *demarcation line* of the given region.

The *MLP method* consists of finding the shortest polygonal curve, called *the MLP*, lying completely in the closure of the open boundary of a digital region S and encircling the boundary line of S. Figure 2, left, illustrates the definition of the MLP.

The MLP of the closure of an open boundary of a digital region is uniquely defined[19] if this open boundary is homoeomorphic with the annulus. The length of this MLP is finally used as the *MLP estimate of the perimeter* of the digital region.

2.2. Digital straight segments

Digital straight lines[16] may be defined based on the notion of a half plane. For that reason we have to assign coordinates to all cells of the complex C. In general the theory of ACC's allows the specification of *topological coordinates* of cells[10]: Assume a one-dimensional cellular complex consisting of an alternating sequence of 0- and 1-cells. These cells are uniquely numbered by integers specifying topological coordinates for all 0- and 1-cells. The cross-product of two of such one-dimensional cellular complexes is isomorphic to a homogeneously two-dimensional complex C introducing tuples of integers as topological coordinates for all 0-, 1-, and 2-cells.

A *digital half-plane* in C is the set of all cells whose coordinates satisfy a given linear inequality. Note that a digital half plane satisfies the definition of a digital region. Thus the boundary line of a digital half plane is defined accordingly.

A *DSS* is a connected subset of the boundary line of a digital half-plane. It follows that a DSS contains no pixels and consists only of cracks and points as stated above for boundary lines of digital regions in general.

Note that this is not the only way for defining straight lines in finite spaces. In principle there are (at least) two kinds of DSS's[11] considered in the literature on digital geometry: the first one is the DSS as defined above consisting of cracks and points which is also called *boundary DSS*, and a second one, called *visual DSS*, is a sequence of pixels whose topological coordinates satisfy two linear inequalities. The latter kind of the DSS's is more widely represented in the literature since a DSS of this kind is easy to visualize on a computer screen while a boundary DSS requires some additional efforts for visualization.[11]

On the other hand, boundary DSS's describe boundary lines of regions and, being mathematical curves of zero width, they may easily be used in solutions of various problems in digital geometry. Boundary DSS's have been chosen for solving the problem of subdividing a given digital curve into DSS's of maximum length, where a *digital curve* is an alternating sequence of points and cracks. It may or may not be a closed loop.

We consider the digital curve as being *directed*. This means that every crack has one start point and one end point following a unique global orientation of the curve. The longest edge on the convex hull of the point set of a DSS is called its *base*. Any point of a DSS has a restricted distance from the base.[7] If the base lies on the right of the DSS (according to the orientation of the DSS) then it is called the *right base*, see Fig. 3. In this case the *left base* is defined to be that side of the convex hull which is parallel to the right base. The left base may degenerate into a single vertex only.

Similarly, if the longest side of the convex hull lies on the left of the DSS then it is called the *left base*, and the *right base* is then defined to be that side of the convex hull being parallel to the left base. Eventually it consists of a single vertex only.

We use a Cartesian coordinate system in the plane where grid points (0-cells) have tuples of integers as coordinates. All grid points (X, Y) of a DSS satisfy the following two inequalities[7]:

$$0 \leq V \cdot X - U \cdot Y + W \leq |U| + |V| - 1 \tag{1}$$

where $(U, V)^T$ is the *tangential vector* parallel to the right base of the DSS having mutually prime integer components and W is an integer of constant value for the DSS and chosen such that $V \cdot X - U \cdot Y + W$ be equal to 0 for any grid point (X, Y) lying on the right base. In what follows we denote the value of $V \cdot X - U \cdot Y + W$ by $H(X, Y)$. It also holds that if $H(X, Y)$ is equal to -1 which means that the point (X, Y) does not belong to the DSS with the given parameters U, V and W, then the parameters may be updated in such a way, that all the previous points of the DSS as well as the recent point (X, Y) belong to one DSS characterized by the updated parameters.

This updating method is implemented in the function $DSS\_Cont()$, see Fig. 5 below. The value of $H(X, Y) = -1$ corresponds to a "right outlier", i.e. to a point which lies to the right of the right base. Similarly, there might be a left outlier. In this case $H(X, Y)$ is equal to $|U| + |V|$. In the cases $H(X, Y) < -1$ or $H(X, Y) > |U| + |V|$ there is no possibility of parameter adjustment, i.e. there exists no DSS containing all previous points together with the current point (X, Y).

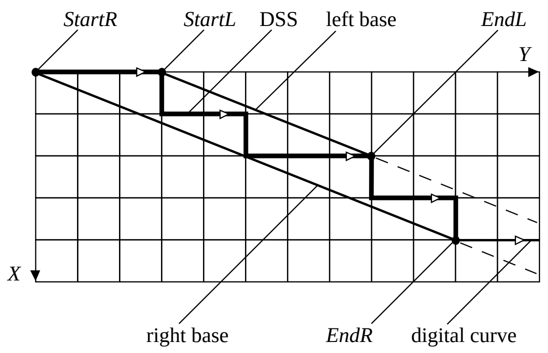

Figure 3. The features of a DSS used for recognition. *EndR* would be the recent point during curve tracing.

2.3. Minimum length polygon

For a polygonal line L in the Euclidean plane let P_L be the closed polygonal area circumscribed by the line L. For boundary line L_1 and demarcation line L_2 it holds $L_1 \subseteq P^{\circ}_{L_2}$. Assume that a polygonal curve is traced counterclockwise. A vertex of this polygonal curve is a *convex vertex* if the curve makes a left turn at this vertex, a *concave vertex* in case of a right turn, and a *colinear vertex* compared to both neighboring vertices otherwise. For such decisions it is common to use the determinant value

$$det(p_1, p_2, p_3) = (X_2 - X_1)(Y_3 - Y_1) + (Y_1 - Y_2)(X_3 - X_1) \tag{2}$$

for three-point sequences $p_i = (X_i, Y_i)$, $i = 1, 2, 3$, which is negative for left turns, positive for right turns, and equals zero for colinear situations assuming an XY-coordinate system as shown in Fig. 3. The vertices of the MLP in $G = P_{L_2} \setminus P^{\circ}_{L_1}$ belong either to the convex vertices of L_1 or to the concave vertices of L_2.[19]

Assume two polygonal Jordan curves L_1, L_2. We say that a ray $\overrightarrow{pq}$ *hits* L_1 *first* iff the ray starts in p, passes through q, intersects L_1 afterwards at a point r, and does not intersect L_2 between q and r. Similiarly we define a situation where ray $\overrightarrow{pq}$ *hits* L_2 *first*. A ray *intersects* a curve if there are points of the curve on the left as well as and on the right of the ray, and the intersection set of ray and curve is not empty. Note that the ray may be incident with a whole straight segment of the curve, i.e. point r could be any point of such a segment in this case. Now assume a boundary line L_1 and a demarcation line L_2 of an open boundary and let $p_0, p_1, \ldots, p_{n-1}$ be the MLP encircling L_1 in the closure of this open boundary. Then it holds[19] that the ray $\overrightarrow{p_i p}_{i+1(modn)}$ hits L_1 first if $p_{i+1(modn)} \in L_2$, or the ray $\overrightarrow{p_i p}_{i+1(modn)}$ hits L_2 first if $p_{i+1(modn)} \in L_1$, for $i = 0, 1, \ldots, n-1$.

3. ALGORITHMS

Both algorithms are described in detail to support reimplementations. The input sequence of grid points is assumed to follow a boundary line of a digital region. In other words, the coordinates of two subsequent points in the sequence with indices i and $i+1$ must 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, crack by crack, a boundary line of a given digitized region.[8] The *input* of both algorithms is a sequence of N grid points $P[1:N]$ on a boundary line of a digital region S, each point being described by a data structure $(P[i].X, P[i].Y)$ containing the integer coordinates of the ith point $P[i]$. Since a boundary line is a closed loop, $P[N]$ must be equal to $P[1]$. For both algorithms the *output* is a value $Perim$ which is the DSS estimate or the MLP estimate of the perimeter of a given digital region S.

3.1. DSS Algorithm

The task consists in tracing a given digital curve $P[1:N]$ crack by crack, and sequentially subdivide it into DSS's of maximum length. The *DSS algorithm* performs the following steps:

(i) initialize a recent DSS by a single crack,

(ii) consecutively during boundary line tracing test the current grid point 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 points, i.e. from the beginning of the recent DSS up to the preceding point, as well as the current point, 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. 4. The algorithm uses variables

1. *DIR*[1:2] for an array containing both crack directions allowed for the recent DSS,
2. $StartL$, $StartR$, $EndL$, $EndR$ for start and end points of the left and right bases (each point is described by a data structure $(intX, Y;)$ containing its coordinates),
3. $Vertex$ for the end point of the previous DSS described by a similar data structure,
4. $Tang$ for a tangential vector specifying the direction of the bases of the actual DSS and described by a similar data structure, while $Tang.X = U$ and $Tang.Y = V$,

```
program DSS_Perimeter (P[1:N]; Perim)
  Begin
    N_DSS:=-1;                          +++ not zero, index of first DSS
    Perim:=0.0;
    for i:=2 to N do
    begin
      dir:=F(P[i]-P[i-1]);              +++ direction of the last step specified by F( )
      If (i=2) then BRK:=1;             +++ for the initialization of parameters at start
      else BRK:=DSS_Cont(P[i], dir);    +++ call of function DSS_Cont( )
      If (BRK>0) then                   +++ an end point of a DSS is found
      begin
        StartL:=StartR:=P[i-1]; EndL:=EndR:=P[i];
        Tang:=P[i]-P[i-1];              +++ vector subtraction
        DIR[1]:=dir; DIR[2]:=-1;        +++ DIR[2] is yet unknown
        HR:=0;                          +++ HR stands for H(X,Y)
        Vertex:=P[i-1];
        N_DSS:=N_DSS+1;
        If (i>2) then
          Perim:=Perim+Distance(Vertex,P[i-1]);    +++ Euclidean distance
      end                               +++ of "If (BRK>0)"
    end                                 +++ of "for...do"
    N_DSS:=N_DSS+1;
    Perim:=Perim + Distance(Vertex,P[i]);
  End.                                  +++ of "DSS_Perimeter"
```

Figure 4. Estimation of the perimeter based on DSS segmentation of a boundary line.

5. *HR* for the integer value of $H(X,Y)$ of the left side of the linear inequality of that Euclidean half-plane whose boundary is incident with the right base of the DSS while *HR* is non-negative for the grid points of the DSS, and
6. *N_DSS* for the index (label) of the DSS's just detected, i for the index of the current point, *dir* for the direction of the last crack, and *BRK* as a flag for starting the next DSS if $BRK = 1$.

The integer function *DSS_Cont* repeatedly called in this algorithm *DSS_Perimeter* has as *input* the current point $P[i]$ described by the data structure $(P.X, P.Y)$ containing the coordinates of $P[i]$ and the direction *dir* of the last crack whose endpoint is $P[i]$. The *output* is equal to 0 if $P[i]$ belongs to the recent, eventually adjusted DSS, or it is equal to 1 if there exists no DSS containing the current point $P[i]$ and all previous points. The variables are as in algorithm *DSS_Perimeter*. An algorithm for the integer function *DSS_Cont* is sketched in Fig. 5.

3.2. MLP Algorithm

As already explained in Section 2.1, this method is based on finding the minimum length polygon which is contained in the closure of the open boundary of a given digital region S and which circumscribes the boundary line of the digital region. The task may also be specified as calculating an MLP in a compact polyhedral set of the Euclidean plane which has two disjoint *isothetic polygonal curves* (i.e. having only edges parallel to the coordinate axes) as its boundary, namely the boundary line L_1, the given digital curve $P[1 : N]$, as its "inner boundary" and the demarcation line L_2, a digital curve $Q[1 : M]$ calculated from $P[1 : N]$, as its "outer boundary". By preprocessing we have ensured that both curves define the boundary of a simple open boundary (a simple closed-loop one-dimensional grid continuum[20]), see right example in Fig. 2.

The two cited theorems in Section 2.3 provide the theoretical basis for the correctness of the following linear-time algorithm for the MLP problem. The *two-beetle algorithm* is as follows:

There are two beetles, BLACK and WHITE, see Fig. 6. Both may only move forward in one direction, say counterclockwise. The start vertex of the first race is the uppermost-leftmost vertex of the inner boundary which is also

```
function DSS_Cont (P, dir)
  Begin
    If (DIR[2]<0 AND dir=DIR[1]) then   +++ following the initial direction
    begin
      EndL:=EndR:=P;                    +++ for EndL and EndR 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 has been found
                                        +++ updating of the value of the left side of
    switch(dir)                         +++ the inequality of the right half-plane
    begin
      case 0: HR:=HR+Tang.Y; break;     +++ HR stands for H(X,Y)
      case 1: HR:=HR-Tang.X; break;
      case 2: HR:=HR-Tang.Y; break;
      case 3: HR:=HR+Tang.X; break;
    end                                 +++ of "switch"
    If (HR>0 AND HR<abs(Tang.X)+abs(Tang.Y)-1) then
      return 0;                         +++ allowed value of HR
    If (HR<-1 OR HR>abs(Tang.X)+abs(Tang.Y)) then
      return 1;                         +++ a non-repairable value
    If (HR=0) then                      +++ P is on the right base
    begin
      EndR:=P; return 0;
    end                                 +++ of "If (HR=0)"
    If (HR=abs(Tang.X)+abs(Tang.Y)-1) then   +++P is on the left base
    begin
      EndL:=P; return 0;
    end                                 +++ of "If (HR=abs...)"
    If (HR=-1) then                     +++ P is a right outlier, DSS is being adjusted:
    begin
      EndR:=P;   StartL:=EndL;  Tang:=P-StartR;         +++ vector subtraction
      HR=0; return 0;
    end                                 +++ of "If (HR=-1)"
    If (HR=abs(Tang.X)+abs(Tang.Y)) then
                                        +++ P is a left outlier, DSS is being adjusted:
    begin
      EndL:=P;   StartR:=EndR;  Tang:=P-StartL;         +++ vector subtraction
      HR=(P.X-StartR.X)*Tang.Y-(P.Y-StartR.Y)*Tang.X;  return 0;
    end                                 +++ of "If (HR=abs...)"
  End.                                  +++ of "DSS_Cont"
```

Figure 5. Decision whether the recent DSS may be extended or not. If "yes" then parameters may be adjusted if necessary.

the first vertex of the MLP. Only at that time BLACK is on the boundary line. After the start of the first race BLACK is only allowed to move into concave vertices of the demarcation line, and WHITE is only allowed to move into convex vertices of the boundary line, and for any new position of a beetle both straight lines "StartVertex to BLACK" and "StartVertex to WHITE" are not allowed to intersect neither the demarcation nor the boundary line. The beetles move alternately (as far as both are allowed to move) and stop if no further move is possible. WHITE begins the first race. All other races are started by the winner of the race before. If one beetle could not move at all during a race then the other one is the winner. If both moved then the winner is the one whose ray "StartVertex to

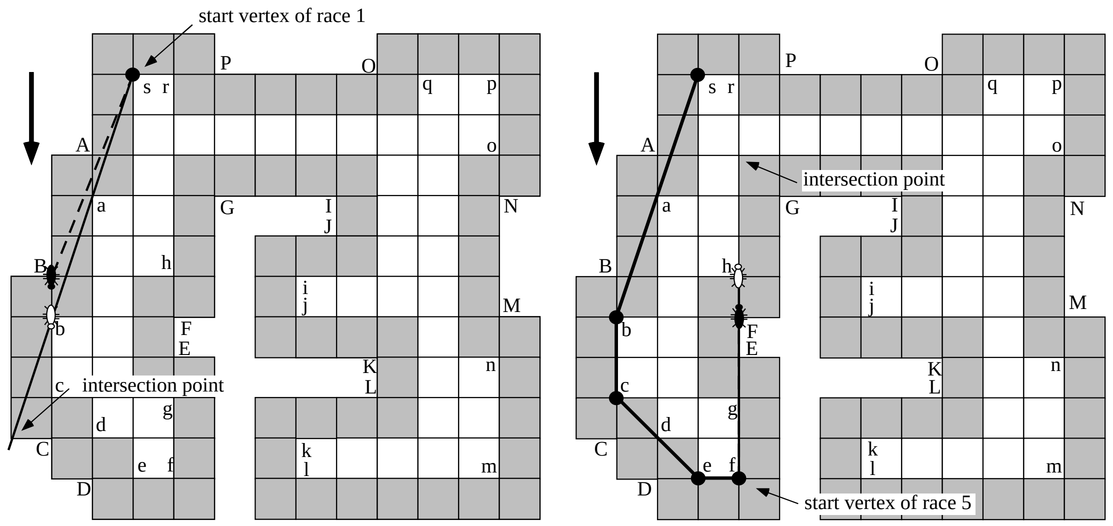

Figure 6. Left: Winner of the first race is WHITE. Right: Winner of the fifth race is BLACK. Its ray "StartVertex to BLACK" hits L_1 first at the indicated intersection point. The ray "StartVertex to WHITE" does not hit L_2 first.

beetle" hits the line of the other beetle first. The position of the winning beetle is the next vertex of the MLP and also its start vertex for the next race. The other beetle starts from its last accepted position in the race before. The sequence of races stops if WHITE reaches the first vertex of the MLP again.

For implementing this algorithm we have to specify **(i)** how to classify line vertices into convex, concave or colinear points, **(ii)** how to ensure that the straight segments "StartVertex to beetle" do not intersect L_1 or L_2, and **(iii)** how to decide whether a ray "StartVertex to beetle" hits L_1 or L_2 (the search interval is actually restricted to the final position of the other beetle and its next test position, i.e. the next concave or convex vertex).

Decision **(i)** is easy to implement using directional values $det(p_1, p_2, p_3)$ as specified in Equ. 2. The procedures for decisions **(ii)** and **(iii)** use variables:

1. *SV* denotes the start vertex of the race,
2. *W* is the last accepted position of WHITE, *TW* is the (next) test position of WHITE, *NW* is the number of steps of WHITE in this race so far, *Wmove* is a flag equals 1 if further WHITE moves may be expected in this race, and equals 0 if no further WHITE move is possible in this race,
3. *B*, *TB*, *NB*, *Bmove* are the corresponding variables for BLACK,
4. *BAY* and *PEN* are two flags for "bay" or "peninsula" situations, and
5. *NextMove* is either equal *Black* or *White*.

Furthermore, let $next\_convex(P, i)$ be the index of the next (modulo N) convex vertex on P following $P[i]$ not equal to i, and $next\_concave(Q, i)$ be the index of the next (modulo M) concave vertex on Q following $Q[i]$ and not equal to i. An algorithm for decision **(ii)** is shown in Fig. 3.2.

The test for the next BLACK position is analogous to this procedure and includes a possible detection of a "peninsula situation" leading to PEN := 1. The decision **(iii)** about the winning beetle is possible as follows:

```
IF ( NB = 0  OR  ( det(S,W,TB) = leftTurn  AND  det(S,W,B) NEQ leftTurn ) )
         THEN  Winner := White ELSE Winner := Black
```

The position of the winning beetle is the next vertex of the MLP and the new start position of the next race. Finally the MLP estimate of the perimeter is the length of the calculated minimum length polygon.

```
procedure NEXT_WHITE (P,SV,W,B,TB,TW,nextConvex,NB,NW,Wmove,BAY,PEN,NextMove)
  Begin
    nextConvex := next_convex(P,nextConvex);
    TW := P[nextConvex];                    +++ a convex vertex on P
    IF
        ( NB = 0  OR  det(SV,B,TW)  NEQ  rightTurn )
        AND
        ( NW = 0  OR  det(SV,W,TW)  NEQ  leftTurn )
        AND
        ( PEN = 0 OR  a 4-neighbor of TB in P has an index >= nextConvex )
    THEN
        begin
          W := TW;  NW := NW + 1            +++ next WHITE position found
          NextMove := Black;                +++ has to be confirmed by "Bmove=1"
        end
   ELSE
     begin
       IF ( NB > 0 AND det(SV,B,TW) = rightTurn ) THEN BAY := 1;   +++ "bay situation"
       Wmove := 0
     end                                    +++ no further WHITE move in this race
  End.                                      +++ of "NEXT_WHITE"
```

Figure 7. Test of *TW* to become the next WHITE position. If accepted then BLACK may move next if *Bmove*= 1, otherwise WHITE may try to continue to move.

4. DIGITIZATIONS AND MULTIGRID CONVERGENCE

The preimage is assumed to be a bounded compact set Θ in the Euclidean plane. A grid constant τ specifies the length of a grid edge of an orthogonal grid in this plane. For studying multigrid convergence of calculated features we decrease the value of τ specifying finer and finer grids. For the DSS method it is assumed that a closed unit square belongs to the digital region S_τ^{DSS} iff at least 50% of the area of this unit square belongs to Θ. For the MLP method it is assumed that a closed unit square belongs to the digital region S_τ^{MLP} iff this closed unit square is completely contained in the interior S° of Θ.

However, these ideal digitization approaches are difficult to implement. Practically we are using the following approximations in our experiments: a closed 2-cell belongs to S_τ^{DSS} iff its midpoint is in S (the *midpoint scheme*), and it belongs to S_τ^{MLP} iff all its four vertices are in S (the *four-vertex scheme*). For example, the digital regions in Fig. 1 and on the left in Fig. 2 have been generated assuming an identical region Θ in the real plane as a preimage.

Six different regions Θ have been used in our experiments: a circle, a square rotated by 45°, a square rotated by 22.5°, a halfmoon generated by two overlapping circles of identical size, the yin-part in the Chinese yinyang symbol,[23] and 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. All regions have been digitized for all grid resolutions between $n = 32$ and $n = 1024$, including these two values. For the resulting digital regions the DSS partition and the MLP were calculated allowing to compute the DSS estimate and the MLP estimate of the perimeter of region Θ.

The *error* is defined as the percentage of the absolute value of the difference between the true perimeter and the calculated perimeter to the true perimeter. The generated number of segments allows to specify another interesting *trade-off measure*. The product of error times number of segments informs about the efficiency of the convergence: if this product decreases faster or increases less for algorithm A in comparison to the second algorithm B then algorithm A is more efficient in achieving reduced errors without generating too many new segments. Figure 8 shows the resulting diagrams for the error (sliding mean of 64 values and standard deviation with respect to this sliding

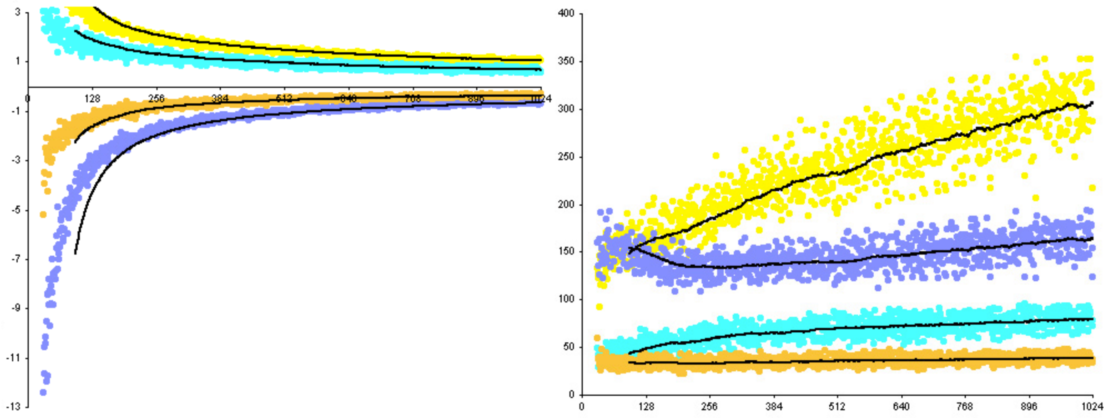

Figure 8. These diagrams illustrate the behaviour of both methods for the six selected regions digitized for grids of resolution 32, 33, ..., 1024. Left: Error diagram with scale from 0 % to 100%, the curves from top to bottom are: standard deviation of MLP error, standard deviation of DSS error, sliding mean of DSS error (times -1, just for grahical separation of this curve from the standard deviation curves), and sliding mean of MLP error (also times -1). Right: Trade-off diagram with scale "percent times number of segments", the curves from top to bottom are: standard deviation of MLP trade-off, sliding mean of MLP trade-off, standard deviation of DSS trade-off, and sliding mean of DSS trade-off.

mean) and for the trade-off values (again sliding mean of 64 values and corresponding standard deviation) combined for all six examples of regions.

5. CONCLUSIONS

The experiments have shown that in all cases there was "fast" convergence also for non-convex regions such as the halfmoon, the sinc-curve or the yin-part of the yinyang symbol. The DSS algorithm was with respect to run time behaviour considerably faster than the implemented MLP algorithm. The DSS partition also shows in general faster convergence and a better efficiency with respect to our trade-off measure. However, the latter property also means that the MLP is "somehow closer" to the boundary.

The application of the MLP method requires a special treatment of cases for "hole entrances of width 1 or 2", see Fig. 2. However, the existence of such cases may also be explained as “insufficient sampling” with respect to the sampling theorem ("at least two samples per wave length"). The MLP is uniquely defined.[19] The DSS partition may (slightly) change dependend upon start point and orientation.

The DSS method was considered by many authors for more than 20 years. The MLP method should still be studied more carefully with respect to optimized algorithms.

The midpoint digitization scheme could also be used for studying the MLP method because the four-vertex scheme might be a reason that MLP “typically” approximates curve length from below but DSS results “typically” oscillate (i.e. positive or negative differences) around the true value. However the sliding mean (say of 64 values) of the differences between calculated and true perimeter values is also permanently slightly negative for the DSS estimates for these six selected curves.

References

[1] 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.

[2] E. Creutzburg, A. Hübler, O. Sýkora: Geometric methods for on-line recognition of digital straight segments. *Computers and Artificial Intelligence* **7** (1988) 253–276.

[3] 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.

[4] E. Khalimsky: *Ordered Topological Spaces* (in Russian). Naukova Dumka, Kiev (1977).

[5] R. Klette, A. Rosenfeld, F. Sloboda (eds.): *Advances in Digital and Computational Geometry.* Springer, Singapore (1998).

[6] V. Kovalevsky: Finite topology as applied to image analysis. *CVGIP* **46** (1989) 141–161.

[7] V. Kovalevsky: New definition and fast recognition of digital straight segments and arcs. Proc. 10th ICPR, Atlantic City, June 16-21 (1990) 31–34.

[8] 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.

[9] 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.

[10] V. Kovalevsky: A new concept for digital geometry. in: *Shape in Picture* (Y.-L. O et al., eds.) Springer, Berlin (1994) 21–36.

[11] 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).

[12] V. Kovalevsky: A topological method of surface representation. in: *Discrete Geometry for Computer Imagery* (G. Bertrand, M. Couprir, L. Perroton, eds.), LNCS 1568, Springer, New York (1999).

[13] Z. Kulpa: On the properties of discrete circles, rings, and disks. *CGIP* **10** (1979) 348–365.

[14] 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.

[15] W. Rinow: *Lehrbuch der Topologie.* Deutscher Verlag der Wissenschaften, Berlin (1975).

[16] A. Rosenfeld: Digital straight line segments. *IEEE Trans. Comp.* **C-28** (1974) 1264–1269.

[17] W. Scherrer: Ein Satz über Gitter und Volumen. *Mathematische Annalen* **86** (1922) 99–107.

[18] 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.

[19] 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.

[20] 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.

[21] K. Voss: Integralgeometrie, digitale Kurven und Umfangsmessung. Technical Report, FSU Jena (1999), see http://pandora.inf.uni-jena.de/pframe.phtml/e/public.inc.

[22] M. Worring, A.W.M. Smeulders: Digitized circular arcs: Characterization and parameter estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence* **17** (1995) 587–598.

[23] 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.