A New Algorithm for Triangulation from Cross Sections and Its Application to Surface Area Estimation

Petra Wiederhold, Mario Villafuerte

Departamento de Control Automático, Centro de Investigación y de Estudios Avanzados (CINVESTAV–IPN), Av. I.P.N. 2508, Col. San Pedro Zacatenco, Mexico 07000 D.F., Mexico

Received 8 February 2010; accepted 8 October 2010

ABSTRACT: This article proposes a new heuristic (locally determined) algorithm for the triangulation between point sequences representing cross-sectional contours of a surface. Such point sequence is required to be the set of vertices of a polygon representing a Jordan curve and approximating the contour. The algorithm which is applied directly to the whole point sequence, is very simple since it is based on comparisons between coordinates within a plane. The triangulation algorithm is applied to surface area estimation, where we use the polygonal approximations of cross sectional contours given by the sequences of MLP (minimal length polygon) vertices or of DSS (maximal digital straight segments) vertices. The test surfaces of known area are ellipsoids of revolution and hyperboloids of one sheet. © 2011 Wiley Periodicals, Inc. Int J Imaging Syst Technol, 21, 58–66, 2011; Published online in Wiley Online Library (wileyonlinelibrary.com). DOI 10.1002/ima.20267

Key words: triangulation from contours; triangulation from polygons; heuristic triangulation algorithm; surface area estimation

I. INTRODUCTION

Triangulation from cross-sectional contours is one of the most important approaches to reconstruction and representation of surfaces within digital image analysis. It consists in the generation of a triangular mesh, where each cross-sectional contour is represented as a polygon, and then the set of triangles spanned by each two sequences of polygon vertices representing consecutive contours, is constructed. The representation of a surface as triangular mesh has advantages with respect to the visualization and analysis of that surface, in particular for the estimation of the surface area. The supposition that the surface is given as a stack of contours, is justified by the practical argument that many image acquisition methods generate cross-sectional 2D images of a 3D object of interest (for example in computational tomography). Also, from a 3D digital object or surface already identified, slices parallel to some coordinate plane are easily obtained.

Surface area estimation is an interesting mathematical topic as well as an important task within the analysis of 3D objects in many application areas of digital image analysis. The surface area of a smooth surface of a 3D Euclidean object can be estimated by the sum of the areas of all polygons which are the faces of a polyhedron approximating the surface, where this polyhedron is determined from the corresponding 3D digital object. This has been proved for the polyhedron given as the relative convex hull, where the accuracy of the area estimator is improved by augmenting the digitization resolution (multi-grid convergence) in (Sloboda and Zatco, 2001). Nevertheless, there is no algorithm known so far, for the determination of the relative convex hull. Other polyhedra used in 3D image analysis for providing surface area estimators, are based on digital polygon segments, or on the marching cubes algorithm, see (Klette and Rosenfeld, 2004). In this article, we will estimate the surface area as the sum of areas of all triangles being faces of the polyhedron generated by triangulation from cross-sectional contours represented by minimal length polygon (MLP) and maximal digital straight segments (DSS) polygons.

The contributions of this article are as follows:

1. We propose a new heuristic triangulation algorithm to be performed for two ordered sequences of points which represent consecutive cross-sectional contours of the surface of a 3D digital object. Such point sequence is not required to be a digital (4-, 6-, or 8-) curve, but it is supposed to be the set of vertices of a polygon approximating a contour. That polygon is supposed to form a Jordan curve which is a cross-sectional curve of a surface lying in the Euclidean space. The polygon is required neither to be convex nor to have vertices with integer coordinates. The algorithm is applied directly to the whole point sequence. If the sequences have m and n points, respectively, the complexity of our algorithm is $O(m + n)$. The algorithm is based on simple proximity criteria which are checked by comparisons of coordinates within a plane.
2. We apply our algorithm to surface area estimation based on triangulation of sequences of points representing contours of serial sections. We propose in particular to use the sequences of MLP vertices or of DSS vertices of the contours. The (artificial) test surfaces of known area are ellipsoids of revolution and hyperboloids of one sheet, whose digital version is represented as a stack of Outer Jordan digitizations of cross-sectional curves.

Correspondence to: Petra Wiederhold; e-mail: biene@ctrl.cinvestav.mx

The novelties of this article with respect to its preliminary conference version (Wiederhold and Villafuerte, 2010) consist in a much more detailed presentation and analysis of the triangulation algorithm, including variants of the algorithm not presented before, and the augmentation of the surface area experiments by hyperboloids.

The only preprocessing of the sequences (other than the determination of a polygonal approximation of the contour) needed in our triangulation algorithm is that of determination of the initial edge. The initial edge problem is determined by particular properties of each practical application, as for example by the digitization resolution and by the distance between cross sections, as well as by the expected smoothness, size, and form of surfaces to be reconstructed. We applied known methods of initial edge determination based on proximity in our algorithm and experiments.

For our experiments on surface area estimation, we suppose a 3D digital object to be a bounded 6-connected subset of (voxels in) $\mathbb{Z}^3$; its digital surface is the set of all object voxels which have a 26-neighbor in the complement of the object. The 3D digital object is given initially as a stack of cross sections being 2D digital (4-connected) objects lying in parallel not necessarily equidistant planes. Equivalently, a digital surface is given as a stack of (Jordan) curves lying in such parallel planes and represented initially as digital 4-curves (actually, grid points are voxels when considered in $\mathbb{Z}^3$, but pixels when considered within the slicing plane). Nevertheless, as pointed out before, each contour considered as input data for our triangulation method is supposed to be represented by any ordered sequence of points, which generates a polygonal approximation of the digital contour. Our requirement on the initial digital 4-curve is only motivated by the necessity of determination of the MLP or of the DSS vertices, which in turn will be used for the surface area estimation experiments. Note that any ordered sequence of grid points forms a polygon in the Euclidean plane, generated by connecting subsequently the vertices of that sequence by straight line segments.

The article is organized as follows: In Section 2, triangulation from serial cross sections is reviewed. Section 3 presents and analyzes in detail our new triangulation algorithm and shows examples. In Section 4, we apply our algorithm to estimate the area of surfaces of known area (ellipsoids and hyperboloids), using MLP and DSS polygons as input. Section 5 contains conclusions.

II. TRIANGULATION FROM SERIAL CROSS SECTIONS

The aim of a triangulation algorithm for generating a surface from cross-sectional contours is to generate a triangular mesh given as the union of triangulated surface bands spanned by consecutive contours. The contours in general are not required to be represented by digital (for example, 4-, 6-, or 8-connected) curves. Commonly, each contour is represented by a sequence of points, which are vertices of a polygon approximating the digital contour. Using a ''good'' polygonal approximation of each contour, the polyhedral surface generated by triangulation is expected to be a ''good'' approximation of the digital surface.

Problems to be solved to satisfy the aim of triangulation from cross sections, include the identification of corresponding contours in two sections, and the design of methods for the triangulation from-one-to-many or from-many-to-many contours. In most triangulation algorithms which have been published, these problems are mainly solved by some preprocessing of the given contours, and the triangulation itself is performed only between two consecutive contours. Most triangulation algorithms, to guarantee the triangular mesh generated to represent a topologically correct surface (without self-intersections), require strong conditions on the contours, as convexity, similarity, alignment, or, they first decompose the contours in parts which satisfy such conditions (and where eventually, the correspondence problem for these parts has to be solved as well), see (Keppel, 1975; Ganapathy and Dennehy, 1982; Kehtarnavaz et al., 1988; Ekoule et al., 1991).

Optimal or global triangulation algorithms use global information about the given contours or satisfy global optimization criteria, as for example surface area minimization (Fuchs et al., 1977) or volume maximization (Keppel, 1975), where searching for determining an optimal path in a weighted graph is applied. The Delaunay triangulation is the base of other global algorithms (Wang et al., 2006).

Heuristic triangulation algorithms are to be applied to the input contours using only local information to generate the triangles (Christiansen and Sederberg, 1977; Wang and Aggarwal, 1986; Shirai, 1987; Ekoule et al., 1991). Those algorithms are characterized by simplicity and high computational efficiency. For two consecutive contours given by sequences of m and n points, respectively, a heuristic algorithm has time complexity $O(m + n)$. Nevertheless, the exact performance time depends on the criteria used for the decision about the next edge, and this complexity does not include preprocessing. For example, the local version (Ganapathy and Dennehy, 1982) of Keppel's algorithm (Keppel, 1975) needs the calculation of sophisticated formulae and iterative decompositions of the contours into convex and concave parts. A heuristic triangulation algorithm applied to two consecutive contours, starts with the selection of an initial pair of points from both contours, and subsequently, whenever a pair has been selected, the algorithm decides which the next pair is, based on some local criterion (Shirai, 1987). The end condition, for a wide class of surfaces, principally consists in having found again the initial pair of points. Each pair of points defines an edge (line segment), and each two edges consecutively detected (together with an edge of one of the two polygons given by the contours) form a triangle of the triangulation of the surface.

This article only focuses on the problem of the triangulation between two consecutive contours representing plane Jordan curves, given by ordered point sequences. Our algorithm can be easily adapted to handle point input sequences which represent pieces of curves (that is, Jordan arcs) instead of closed curves.

III. A NEW TRIANGULATION ALGORITHM

A. Assumptions and Notations. Suppose $\mathbb{R}^3$ or $\mathbb{Z}^3$ to be represented by a Cartesian coordinate system where the x-axis points to the right, the y-axis points to the front and the z-axis points upwards. Let the surface of a 3D object be sliced such that each cross section is parallel to the x–y-plane. Hence, in each such plane, the x-axis points to the right, and the y-axis points downwards, as it is common in many image analysis computer systems. In each slicing plane, only one point sequence representing a contour is considered as input to our triangulation algorithm.

The triangulation is performed for two point sequences representing two consecutive slices, without reference to the eventually existing result of the triangulation of pairs of sequences, previously treated. So, each pair of sequences is considered independently from the other ones.

Suppose $C^{(1)} = \left\langle p_1^{(1)}, p_2^{(1)}, ..., p_n^{(1)} \right\rangle$ and $C^{(2)} = \left\langle p_1^{(2)}, p_2^{(2)}, ..., p_m^{(2)} \right\rangle$ to be two ordered cyclic ($p_{n+1}^{(1)} = p_1^{(1)}$ and $p_{m+1}^{(2)} = p_1^{(2)}$) sequences of points representing consecutive contours. Let us denote the coordinates of these points by $p_k^{(1)} = (x_k^{(1)}, y_k^{(1)}, z_k^{(1)})$, $1 \le k \le n$, $p_l^{(2)} = (x_l^{(2)}, y_l^{(2)}, z_l^{(2)})$, $1 \le l \le m$.

Without restriction of generality, let $(p_1^{(1)}, p_1^{(2)})$ denote an initial edge of triangulation between $C^{(1)}$ and $C^{(2)}$. The subsequent formulations of our algorithm are valid under the following conditions:

i. $n \le m$, $n \ge 1$, $m \ge 3$.
ii. both sequences follow the contours in the mathematically negative (clock-wise) sense (when the plane containing the contour is viewed from the top);

For the degenerated case $n = 1$, we simply connect the unique point of $C^{(1)}$ with each point of $C^{(2)}$.

B. On the Initial Edge Problem. Intuitively, the initial edge for the triangulation between two consecutive cross-sectional contours should belong to a topologically correct approximation of the surface. That means, this edge should not cross through the volume of the 3D object enclosed by the surface. Since this article does not consider any particular application of 3D image analysis (so, we have no particular a priori knowledge about the surface), we apply a known method of initial edge determination, based on proximity:

For an arbitrary initial point p from sequence $C^{(1)}$ (for example $p_1^{(1)}$), the corresponding initial point q from sequence $C^{(2)}$ is selected as having the minimal (Euclidean) distance to p, among all points of $C^{(2)}$. This requires the inspection of the whole list $C^{(2)}$. This method provides good results whenever subsequent contours are similar and approximately aligned, which is certainly true for smooth surfaces of 3D objects and a sufficiently high digitization resolution. In experiments applying our triangulation algorithm to (artificial) nonsimilar nonaligned polygons, we use the additional requirement that both vertices of the initial edge should have a similar local curvature, in particular for example, that both vertices are convex.

C. End Condition. After having determined the initial edge, the next problem is how to continue from any current edge (the last edge having been determined) $(p_i^{(1)}, p_j^{(2)})$ to the next one. We name $p_i^{(1)}$ the current point on $C^{(1)}$ and $p_j^{(1)}$ the current point on $C^{(2)}$.

Before searching on the continuation of the triangulation, the end condition of the algorithm is checked. The end condition is satisfied whenever for the first time one of the two current points, say p, coincides with one of the vertices of the initial edge. (Note that the point sequences are cyclic; after the last point, the first point is considered again.) In this situation we can have two cases as follows:

- **End case a**. the other current point, say q, is the last point of its sequence. Then the algorithm is finished. We have only to add the last triangle to the triangulation: determined by the current edge, the initial edge, and the straight line segment connecting q with the first point of the sequence containing q.
- **End case b**. q is not the last point of its sequence. Then, denoting the points subsequently being successors of q within this sequence up to its last point by $q_1, q_2, \ldots, q_k$, p has to be connected with all these points $q_1, q_2, \ldots, q_k$ to form triangulation edges and triangles, and then, q_k satisfies the end case a.

D. Determination of the Next Edge. Now, let $(p_i^{(1)}, p_j^{(2)})$ be any current edge which does not satisfy the end condition. There are only two possibilities for the next edge:

- $(p_{i+1}^{(1)}, p_j^{(2)})$, we will say then that $C^{(1)}$ provides the next point, or
- $(p_i^{(1)}, p_{j+1}^{(2)})$, we will say then that $C^{(2)}$ provides the next point.

For the formulation of the algorithm, let us call a point of $C^{(1)}$ or $C^{(2)}$ a remaining point of the sequence if it has not yet been provided as a vertex of any triangulation edge.

The decision about the next edge is taken purely by comparisons involving the x- and y-coordinates of the contour points. So, triangulation is completely decided within the x–y-plane, and after this, the triangulation edges (and triangles) are put up into the 3D space. In a simplified manner, the following two types of main steps are performed:

Step A. If both sequences (projected onto the x–y-plane) advance towards the same direction, then that sequence which advances less, provides the next point.

Step B. Whenever the condition in step A is not satisfied, then $C^{(2)}$ provides the next point whenever the number of remaining points in $C^{(2)}$ is larger or equal than the number of remaining points of sequence $C^{(1)}$. Otherwise $C^{(1)}$ provides the next point.

Step B reflects a situation where the triangulation essentially changes its proceeding direction, and in this case the sequence which still has more remaining points, is preferred to provide the next vertex of triangulation edge.

The algorithm has four steps of type A, since it searches subsequently in four directions, in the following order: to the right, upwards, to the left, downwards. We interpret the idea of the condition in step A as follows: $C^{(1)}$ advances strictly, and $C^{(2)}$ advances or keeps constant, towards this direction. That means, starting from the current edge $(p_i^{(1)}, p_j^{(2)})$, the algorithm performs searching and decisions in the following order and manner (Recall that $n \le m$, the y-axis points downwards, and contours are traced clock-wise.):

TRIANGULATION ALGORITHM.

At the beginning of each step, check the end condition of the algorithm, in case it is satisfied, the algorithm is finished.
Step A1. ''to the right'':
If $x_{i+1}^{(1)} > x_i^{(1)}$ and $x_{j+1}^{(2)} \ge x_j^{(2)}$ then (if not, continue searching to A2):
if additionally $x_{i+1}^{(1)} \le x_{j+1}^{(1)}$ then $C^{(1)}$ provides the next point; otherwise $C^{(2)}$ provides the next point. Then go to A1.
Step A2. ''upwards'':
If $y_{i+1}^{(1)} < y_i^{(1)}$ and $y_{j+1}^{(2)} \le y_j^{(2)}$ then (if not, continue searching to A3):
if additionally $y_{i+1}^{(1)} \ge y_{j+1}^{(2)}$ then $C^{(1)}$ provides the next point; otherwise $C^{(2)}$ provides the next point. Then go to A1.
Step A3. ''to the left'':
If $x_{i+1}^{(1)} < x_i^{(1)}$ and $x_{j+1}^{(2)} \le x_j^{(2)}$ then (if not, continue searching to A4):
if additionally $x_{i+1}^{(1)} \ge x_{j+1}^{(2)}$ then $C^{(1)}$ provides the next point; otherwise $C^{(2)}$ provides the next point. Then go to A1.
Step A4. ''downwards'':
If $y_{i+1}^{(1)} > y_i^{(1)}$ and $y_{j+1}^{(2)} \ge y_j^{(2)}$ then (if not, continue to step B):
if additionally $y_{i+1}^{(1)} \le y_{j+1}^{(2)}$ then $C^{(1)}$ provides the next point; otherwise $C^{(2)}$ provides the next point. Then go to A1.
Step B. ''essential change of direction'':
The sequence having a larger (or equal) number of remaining points, provides the next point. Then go to A1.

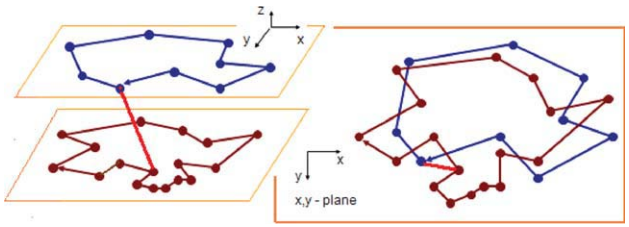

Figure 1. An initial triangulation edge has been selected between sequence 1 (upper plane) and sequence 2 (lower plane). Recall that all subsequent decisions are made by comparing positions of the points projected into the *x*–*y*-plane. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

In any case, from the steps A1–A4, B, a new current edge is obtained, which forms a new triangle, together with the triangulation edge previously detected and the straight line segment connecting the new current point and the previous point of that sequence which provided the new current point.

E. Adaptation of the Algorithm to be Applied to Pieces of Contours. Our algorithm can be easily adapted to be applied to ordered sequences $C^{(1)} = \{p_1^{(1)}, p_2^{(1)}, \cdots, p_n^{(1)}\}$ and $C^{(2)} = \{p_1^{(2)}, p_2^{(2)}, \cdots, p_m^{(2)}\}$ which are not cyclic and represent only pieces of consecutive contours. That means, the points of each sequence are interpreted as the vertices of a polygonal arc. Particularly, each sequence could be a k-connected digital arc, but this is not required.

Then the initial edge trivially is given by the pair $(p_1^{(1)}, p_1^{(2)})$, and the determination of each next edge algorithm is performed as explained before. The end condition is similar to the cyclic case, taking $(p_n^{(1)}, p_m^{(2)})$ instead of the initial edge.

F. Example. Figures 1–9 show an example of performance of the algorithm. On the left hand side of each figure, the input polygons (sequence 1 with nine points in the superior plane, sequence 2 with 17 points in the inferior plane) and triangulation edges (in green) are shown in 3D space, whereas on the right the two polygons and triangulation edges are represented as projected onto the *x*–*y*-plane.

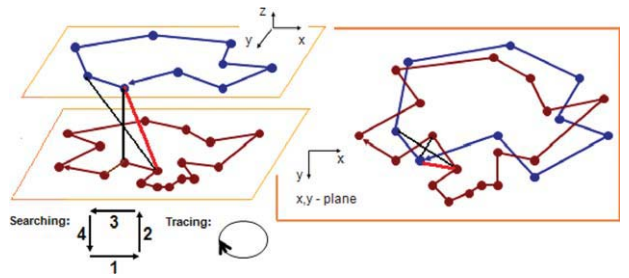

Figure 2. Additionally to the initial edge selected before, the next two possible triangulation edges are shown. The algorithm selects only one edge as follows: It is not true that both sequences advance to the right, so, the first condition of step A1 is not satisfied, we jump to step A2: both sequences advance upwards, but the next point of sequence 2 is positioned below the next point of sequence 1. Therefore, sequence 2 provides the next point, see Figure 3. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

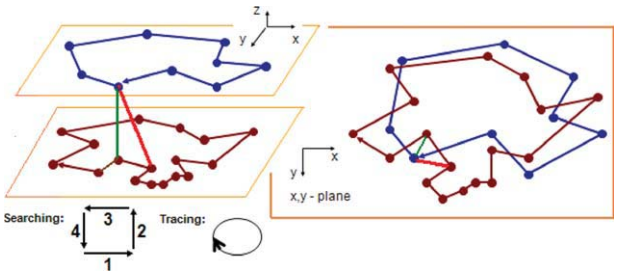

Figure 3. The new current triangulation edge selected in Figure 2 is shown. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

The decisions made by the algorithm are unique, so we would expect that taking any of the triangulation edges detected as the initial edge for repeating the triangulation process, the complete result would be the same. Nevertheless, observe again the example: When having detected the last edge (B,D), the algorithm simply add the last triangle to the triangular mesh of the surface band and stops. If we would continue the algorithm from the last edge (B,D), see Figures 9 and 10, the next edges detected would be (B,C) (which is not the initial edge (A,D)), and then (A,C), which is an edge in Figure 9. Hence the remaining triangulation would be the same as before. This makes also clear that starting with the initial edge (B,C), the triangulation result is unique in the sense that it can be re-generated exactly from any triangulation edge. We will name an initial edge with this property ideal initial edge. Figure 10 shows the solution of triangulation generated by the ideal edge (B,C). Note that any edge of this triangulation solution is an ideal edge.

Let us investigate the triangulation results obtained by other initial edges, which are not contained in the solution of Figure 10. It is easy to see that each of the edges drawn in Figure 11, taken as initial edge, generates firstly up to four edges which are not contained in Figure 10, but the remaining triangulation result coincides with Figure 10.

In Figure 12, we start with the ''bad'' initial edge (A,E), the next edges generated by the triangulation algorithm are (E,F), (F,H), (G,H). But (G,H) is contained in Figure 10, hence the remaining triangulation result will coincide with that of Figure 10, up to reaching edge (A,C) where the end condition is satisfied. Here, the algorithm will construct (A,J), (A,K) and the last triangle

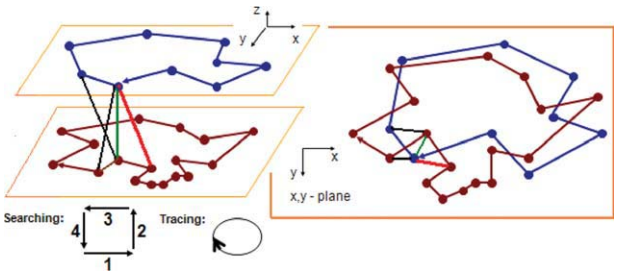

Figure 4. The next two possible triangulation edges are shown. The algorithm starts with step A1 whose first condition is not satisfied. The first condition of step A2 is satisfied since sequence 1 advances strictly upwards, and sequence 2 proceeds constant with respect to the *y*-coordinate, hence the next point of sequence 2 is positioned below the other next point. Sequence 2 provides the next point, see Figure 5. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

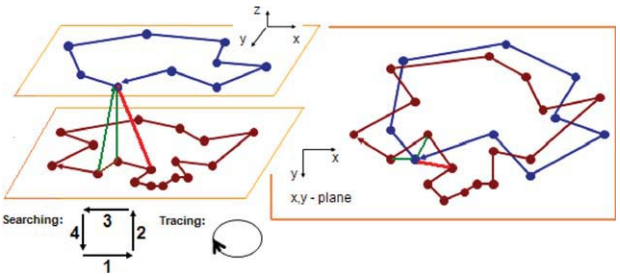

Figure 5. The new current triangulation edge has been added. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

given by (A,K,E) which intuitively presents an erroneous result caused by the ''bad'' initial edge. Nevertheless, if we would permit the algorithm to continue to construct only a few edges more, ignoring the end condition, until having constructed an edge which was generated once before, this error would be corrected by the algorithm itself. A very similar behavior can be observed in Figure 13 starting with the obviously erroneous initial edge (L,P), generating as next edges (L,O), (M,O), (N,O), (N,R), (B,R); and with this latter edge we reached the triangulation from Figure 10.

In consequence, our algorithm is able to present certain robustness with respect to a bad selection of the initial edge. It has to be investigated under which conditions to the contours, such robustness is guaranteed, and how could it be used to generate a ''good'' triangulation of the surface.

G. Variants of the Triangulation Algorithm. Our triangulation algorithm has been designed heuristically. Due to its simple underlying idea, it is similar to the well-known algorithm based on the shortest edge, where from the two possible next edges, the shortest one (due to the Euclidean metric) is selected. Nevertheless, our algorithm, rather than looking for the nearest next point, pretends to follow both sequences due to their direction of advancing, and the results of both algorithms are distinct.

The algorithm described in Section 3.4 presents only one possible particular interpretation of the idea of the Step A. In this interpretation, sequence $C^{(1)}$ acts as a ''leader'' for the proceeding direction: for example, if $C^{(1)}$ advances only to the right whereas $C^{(2)}$ advances only downwards, than the selection of the next point depends only on the direction ''to the right''. If $C^{(2)}$ would be the

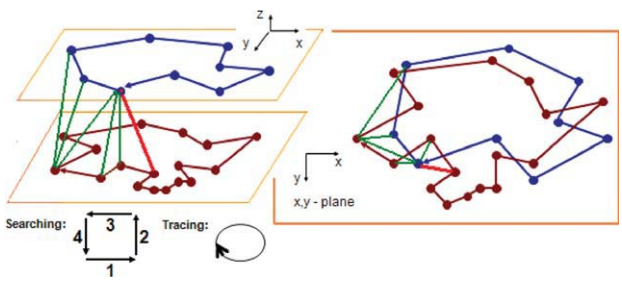

Figure 6. The next triangulation edges are shown, all selected based on the decision of step A1. Then both sequences advance to the right, later upwards, much later downwards, and so on, hence steps A1–A4 generate the next edges shown in Figure 7.[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

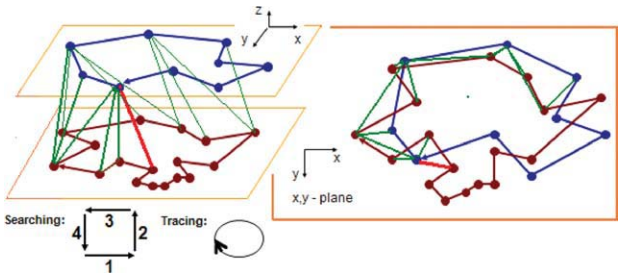

Figure 7. More triangulation edges have been added. Now we have the interesting situation that the first conditions of all steps A1–A4 are not satisfied (sequence 1 advances downwards to the left, but sequence 2 advances upwards to the right), hence step B is applied. Sequence 2 provides the next point, since it has eight remaining points (sequence 1 has four remaining points), see Figure 8. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

''leader'' in the triangulation process, the decision would depend on the direction ''downwards.''

Our algorithm traces both contours in the mathematically negative sense, and searches the proceeding directions in the order ''to the right—upwards—to the left—downwards''. The fact that we always start searching ''to the right'' causes some preference in the x-direction. It is important to note that any other order of searching directions, combined with any tracing sense of the contours, and selecting any of the two sequences as ''leader,'' would provide a triangulation algorithm, and the results in general would be all distinct.

H. Analysis and Comparison with Other Algorithms. By the construction of the triangulation by our algorithm, the result is always a triangular mesh, that means a set of triangles in 3D space, where each triangulation edge is the face of exactly two triangles. It is also clear that for each input point sequence (representing some cross-sectional contour), all edges defined by connecting these points consecutively (also the last one with the first one) by straight lines, are edges of the resulting triangulation.

Since our triangulation algorithm is locally determined, and the determination of the initial edge requires in the worst case (probably two times) the complete inspection of the longer sequence of points, the time complexity for the triangulation of the surface spanned between the two sequences $C^{(1)}$, $C^{(2)}$, is $O(n + m)$. It is important to note that after having determined the initial edge, the algorithm needs only one complete traversal of both sequences, and decisions are taken by comparing only x- and y-coordinates of 3D points rather than on calculations. For completing the triangulation

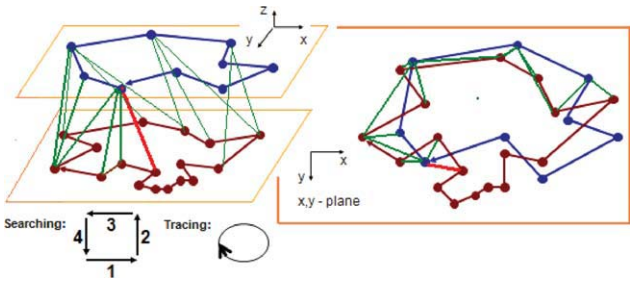

Figure 8. The new current triangulation edge selected in Figure 7 has been added. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

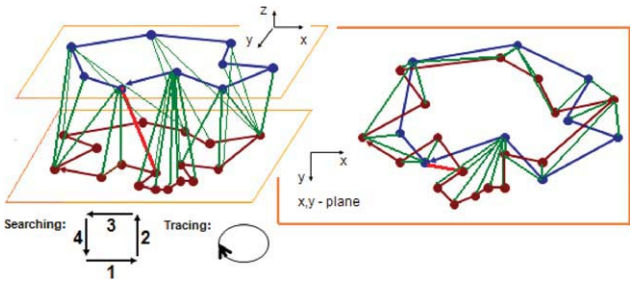

Figure 9. The complete result of triangulation. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

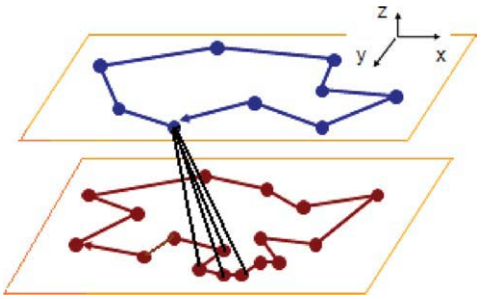

Figure 11. Various nonideal initial edges. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

of the whole surface, subsequently each pair of consecutive contours is triangulated.

The decisions of our algorithm taken on the base of relative positions of contour points, all projected into the *x*–*y*-plane, pretend that during the triangulation, no contour is ''advancing more rapidly than the other one.'' A similar idea is used in the triangulation algorithm in (Yu and Klette, 2001), where Yu and Klette triangulate pieces of cross sections of so called ''visible surfaces.'' Under the restrictive conditions to such surfaces, using our coordinate system and orientation of slices as supposed above, Yu/ Klette would consider a pair of consecutive contours as ''seen from the right'' (and projected on the *y*–*z*-plane), and later on as ''seen from the front, from the left, and from behind,'' to triangulate the surface band portion which can be ''seen'' from each of these orthogonal directions. Each such portion is a bounded horizontal strip (between two lines containing the contours points), on which triangulation always proceeds simply from one end to the other, that means, there is not any possibility of a ''proceeding direction change.'' The algorithm (Yu and Klette, 2001) does not provide immediately the triangulated surface band between the given contours, and the surface has to fulfill much more restrictions than for our algorithm.

Our algorithm is similar to the well-known shortest edge algorithm, but in general, the results are distinct. It is important to note that the shortest-edge algorithm can fail to generate a ''reasonable'' triangulation result, in particular for cases of nonsimilar or nonconvex contours. Figure 14 shows an example of situations where the shortest edge algorithm have been reported to fail (Ekoule et al., 1991), and where our algorithm generates a correct result.

The fact that the result of our triangulation algorithm lies on a topologically well-defined surface (without self-intersections), in general is not guaranteed, and this article does not study the conditions to ensure this. Experimentally we observed that the nonalignment of the input contours seems to be a more important reason than the nonsimilarity, for obtaining a topologically wrong triangulation result.

Note that we are interested in particular in the triangulation from pairs of MLP or DSS polygons. Hence, we cannot suppose for example that for each polygon, its vertices are approximately uniformly distributed on the corresponding Jordan curves, as it is required for several triangulation algorithms. We will have the situation that pieces of large local curvature variations of a contour contribute many points to the sequence, whereas pieces of small such variations are presented by fewer points, and almost linear parts of the contour are represented only by its end points belonging to the sequence.

IV. EXPERIMENTS FOR SURFACE AREA ESTIMATION

Let us consider some examples of smooth surfaces with known surface area, which were digitized and sliced into parallel cross sections, equidistant with distance equal to the digitization resolution, to generate a stack of 4-contours. This is achieved by a rounding procedure, which determines all intersection points of the Euclidean surface with the grid lines within each slicing plane (parallel to some coordinates plane), and then, the coordinates are rounded ''in direction from the interior towards the exterior of the surface.'' All 4-contours together form the Outer Jordan digitization of the object enclosed by the original surface. We generate all 4-contours as ordered sequences of points with the same (clock wise) traversal direction.

From these 4-contours, we construct two types of polygons to be used as input data for our triangulation algorithm:

MLP: The minimal length polygon, MLP, is a polygon uniquely defined for any simple Jordan curve γ in the plane and for any fixed digitization resolution, under the digitization scheme by equidistantly

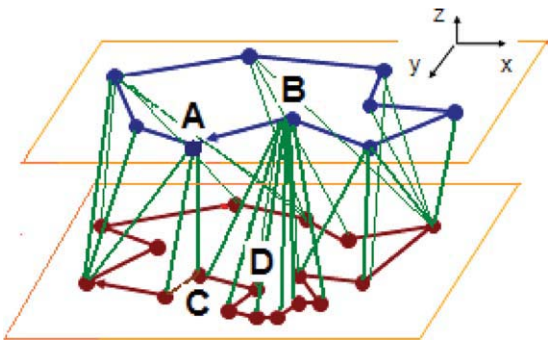

Figure 10. The result of triangulation starting with the initial edge (B,C). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

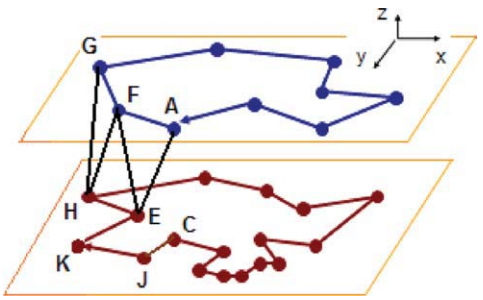

Figure 12. Initial edge (A,E) and subsequent edges. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

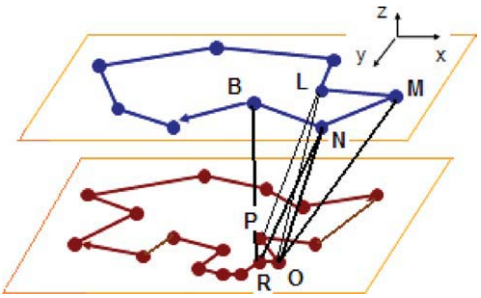

Figure 13. Wrong initial edge (L,P) and sub-sequent edges reaching the ideal initial edge (B,R). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

distributed pixels which equivalently are interpreted as squares. Let A denote the interior of this (closed) Jordan curve. The outer Jordan digitization of A is the set of squares which intersect A, and the inner Jordan digitization of A is the set of squares which are contained in A, we denote by Out(A) and Inn(A), respectively. If the digitization resolution is high enough, the difference set Out(A), Inn(A) is a cyclic sequence of squares in which each square shares an edge with exactly two other squares. The point set union of all squares belonging to this sequence is (a subset of the Euclidean plane and) a compact annular set whose inner frontier and outer frontier both are polygons (point set unions of digital 4-curves), which contains a unique polygonal curve having minimal length among all curves contained in Out(A) and circumscribing Inn(A), as proved in (Sloboda and Stoer, 1994). This polygonal curve is named the MLP of γ.

DSS: The polygon given by maximal digital straight lines, DSS, is determined by starting on a vertex of any given digital 4- or 8-curve, and searching on such segments of the ordered sequence of curve points, which represent the digitization of Euclidean straight line segments and are of maximal length. The result is a polygon whose vertices belong to the original digital curve. This polygon is not uniquely defined since the procedure of its determination in general depends on the starting point. For the theoretical bases on MLP and DSS, see Klette and Rosenfeld (2004).

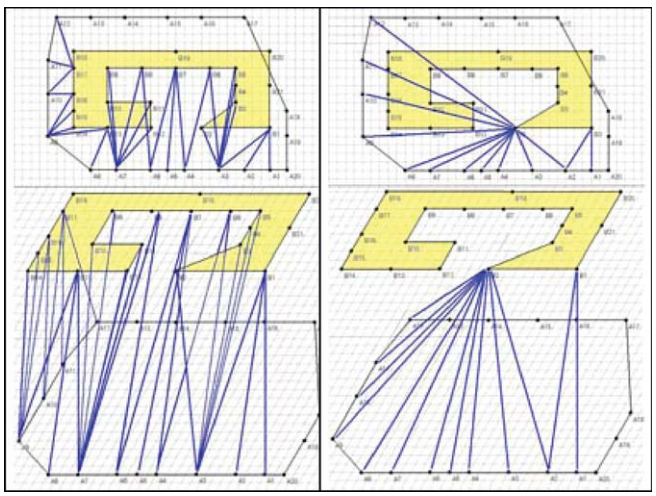

Figure 14. On the left: First triangulation edges determined by our algorithm (projected into the x–y-plane and in 3D space), On the right: the corresponding result of the shortest edge algorithm. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

We selected the MLP and the DSS polygon for representing the cross-sectional contours because these are polygonal approximations not only of the digital contours but also of the cross-sectional (Euclidean) Jordan curves, which have the interesting property of multi-grid convergence (Klette and Rosenfeld, 2004). For increasing digitization resolution, the sequence of estimated curve length numbers given as the sum of Euclidean lengths of all edges of the polygon, converges to the true length of the Jordan curve. This guarantees a good quality of the curve length estimation obtained from the MLP or from the DSS, which gives us some expectation of similarity between the surface area estimated from the complete triangulated surface, and the true area of the Euclidean surface, whenever the digitization resolution is sufficiently high.

By the definition, the MLP of a digital curve has to be determined by interpreting the given curve as the Inner Jordan digitization of some Jordan curve, and then constructing the corresponding Outer Jordan digitization by simple dilatation by one square, as it is common for known MLP algorithms. Both digitizations have contours which are 4-curves, and the MLP vertices have to be selected from the set of vertices of both contours. This justifies our digitization method generating four-curves, applied to the cross-sectional curves. There is no known MLP algorithm which could be applied directly to 8-curves, actually the theoretical base for the MLP would require to be extended to design such an algorithm.

We construct the 4-curve being the Outer Jordan digitization of each cross-sectional curve, and then we adapt the MLP algorithm due to (Klette and Yip, 2000), interpreting the input 4-curve as the outer frontier of the annular set for which the MLP is determined. From the 4-curves already constructed, we use the DSS algorithm of Kovalevsky (Kovalevsky, 1990) for obtaining a DSS polygon for each digitized cross section.

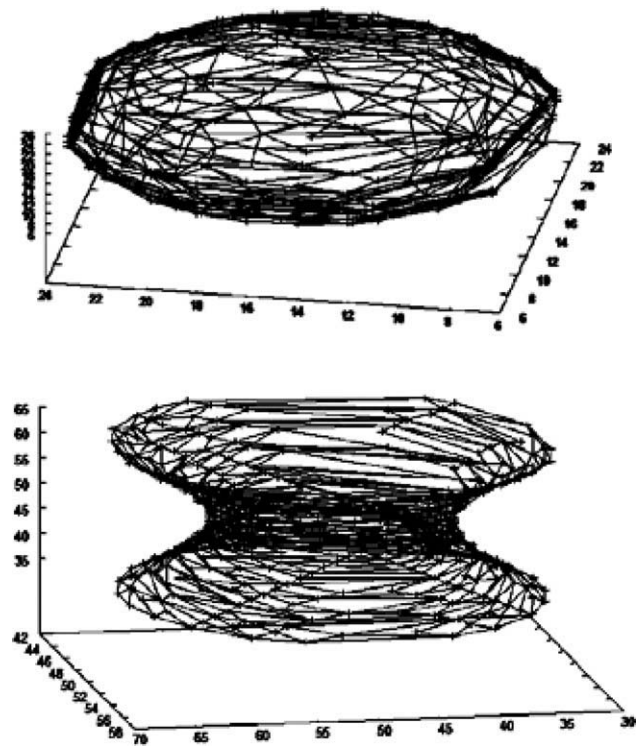

Figure 15. Ellipsoid and truncated hyperboloid, triangulated from DSS cross-sections.

Table I. Surface area estimation for spheres

Radius r	True Area	Area-DSS	Error-DSS	Area-MLP	Error-MLP
15	2827	3305	0.17	2798	0.01
20	5027	5982	0.19	5224	0.04
25	7854	8560	0.089	7879	0.003
30	11,310	12,480	0.103	11,390	0.007
35	15,390	16,910	0.0987	15,550	0.0103
40	20,110	21,670	0.0775	20,290	0.0089
45	25,450	27,520	0.0821	25,670	0.0094
50	31,420	33,620	0.0700	31,800	0.0120
55	38,010	40,710	0.0710	38,510	0.0131
60	45,240	48,490	0.0718	45,890	0.0143
65	53,090	56,700	0.0679	53,820	0.0137
70	61,580	65,990	0.0716	62,410	0.0147
75	70,690	75,360	0.0660	71,710	0.0144
80	80,420	85,980	0.0691	81,670	0.0155

In the experiments presented in this article, the surface is smooth such that consecutive cross sections are aligned and similar, and the digitization resolution is sufficient to represent all details of the object. This justifies the following simple heuristic to decide the initial vertex for the DSS algorithm in each contour: For the first contour, this initial vertex is selected arbitrary, but for each other contour, the initial vertex is selected to be near to the initial vertex of that contour which is the predecessor in the stack of cross sections. Searching for such near other point is performed locally.

As examples, consider ellipsoids represented by their normal forms $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$ (Center point coinciding with the origin of the Cartesian coordinate system, all axes aligned with the coordinate axes). In the case of an ellipsoid of revolution (spheroid), where two of the parameters a, b, c coincide, there are exact formulae for the surface area (Tee, 2005). Supposing $b = c$, and denoting $\delta = 1 - b^2/a^2$, the surface area is given by

$$A = 2\pi b(a \cdot \frac{\arcsin\sqrt{\delta}}{\sqrt{\delta}} + b), \text{ for the case } a > b \text{ (prolate spheroid)},$$

$$A = 4\pi a^2 \text{ for the case } a = b \text{ (sphere)},$$

$$A = 2\pi b(a \cdot \frac{\arcsin h\sqrt{-\delta}}{\sqrt{-\delta}} + b), \text{for the case } a < b \text{ (oblate spheroid)}.$$

The surface area of a bounded portion of a hyperboloid of one sheet is obtained by revolution of a hyperbola given by $\frac{y^2}{b^2}-\frac{x^2}{a^2}=1$, or equivalently $y = f(x) = b\sqrt{1+\frac{x^2}{a^2}}$. Applying the well-known method for calculating a surface as revolution, $A = 2\pi \int_\alpha^\beta |f(x)| \sqrt{1+(f'(x))^2}dx$, we obtain the surface area as $A = 2\pi \int_\alpha^\beta |b\sqrt{1+\frac{x^2}{a^2}}| \sqrt{1+(\frac{b^2}{a^4}x^2(1+\frac{x^2}{a^2})^{-1})}dx$.

Tables I–III show the results of experiments of estimation of the areas of a sphere with increasing radii, of ellipsoids of revolution for some distinct parameters a, b, and of hyperboloids of revolution with increasing parameter $a = b$. The True area is the area of the Euclidean surface, Area-DSS and Area-MLP denote the surface area estimated as the area of the triangulation generated from DSS cross sections or MLP cross sections, respectively; error-DSS and error-MLP denote the corresponding relative errors between the estimated and the true areas. In Table II, a and b are the two distinct parameters of an ellipsoid of revolution.

In the three experiments, the MLP based surface area estimation presents smaller relative errors than the estimations based on the DSS polygons. Whereas for the sphere, the error for DSS for the largest radius considered is still 6.9 percent, it is only 1.55 percent for the MLP estimation. For the hyperboloid with increasing parameters, the DSS estimation reaches an error lower than 5.8 percent, but the MLP estimation has then an error already of only 0.25 percent. For the hyperboloid, both estimations show a limited multi-grid convergence behavior. Nevertheless, the sizes of all 3D objects enclosed by the surfaces are still small. The DSS based estimations always overestimate the area which is clearly a result of the Outer Jordan digitization applied to the cross-sectional curves.

V. CONCLUSIONS

In this article, we proposed a new heuristic triangulation algorithm to be performed for two ordered sequences of points, which are polygons forming Jordan curves and represent consecutive cross sections of a surface. The 3D object enclosed by the surface to be triangulated, is supposed to be a compact simply connected subset of $\mathbb{R}^3$, or its digitization, without bifurcations.

Our algorithm is based on local proximity criteria which are checked by comparisons of coordinates exclusively within a plane to which both sequences are projected. Hence, the algorithm is performed first within this plane, and later, the coordinates of resulting triangulation edges (and triangles) are put back to 3D space. A similar idea is used for methods based on the Delaunay triangulation constraint to a so-called parametric domain (Wang et al., 2005). Our algorithm is alternative and complementary to the known shortest edge algorithm.

Although we formulated the algorithm for slicing planes parallel to some coordinate plane, the algorithm in principal does not need slices to be parallel, and permits nonconvex, nonsimilar, or nonaligned input contours. Nevertheless, the analysis of conditions which guarantee that the algorithm produces topologically correct triangulated surfaces, are not treated in this paper.

The conditions on the contours under which the length of the triangulation edges is bounded by certain number (for example, in dependence from the maximal length of edges of the input polygons

Table II. Surface area estimation for ellipsoids of revolution

Parameter a	Parameter b	True Area	Area-DSS	Error-DSS	Area-MLP	Error-MLP
10	6	663.17	796.42	0.2008	597.64	0.0988
15	8	1300.82	1527	0.1739	1245	0.0429
20	10	2147.84	2318	0.0792	2073	0.0348
25	13	3509.78	3839	0.0938	3638	0.0365
30	15	4832.65	5099	0.0551	4872	0.0081
35	17	6364.80	6625	0.0409	6194	0.0269

Table III. Surface area estimation for hyperboloids

Parameter $a = b$	True Area	Area-DSS	Error-DSS	Area-MLP	Error-MLP
15	7625	8781	0.1516	7600	0.0033
20	8753	9710	0.1094	8789	0.0042
25	9986	11,020	0.1037	10,020	0.0036
30	11,291	12,300	0.0894	11,350	0.0053
35	12,650	13,700	0.08296	12,730	0.0063
40	14,050	15,170	0.0797	14,090	0.0028
45	15,480	16,500	0.0659	15,540	0.0039
50	16,933	17,970	0.0612	16,970	0.0022
55	18,404	19,590	0.0644	18,460	0.0030
60	19,890	21,040	0.0578	19,920	0.0015
65	21,387	22,730	0.0628	21,410	0.0011
70	22,893	24,340	0.0632	22,900	0.0003
76	24,711	26,150	0.0582	24,660	0.0002
79	25,623	27,100	0.0576	25,560	0.0025

and the Hausdorff distance between the two polygons), are under investigation.

The z-coordinates of the points never are used in the algorithm, so, the assumption that the input sequences represent contours lying in exactly parallel planes could be weakened.

We applied our triangulation algorithm to the generation of triangulated surfaces from Euclidean surfaces of known area, to estimate this area by the sum of the areas of all generated triangles. We present experiments with ellipsoids of revolution and hyperboloids of one sheet, where the cross-sectional curves are represented by MLP and DSS polygons. In our experiments, the surface area estimated from cross-sectional MLP's showed more accuracy that the estimation obtained from DSS polygons.

Our triangulation from DSS polygons or from MLP cross sections satisfies the definition of a global polyhedrization method due to definition 8.7 of (Klette and Rosenfeld, 2004), since the decision for each digital surface vertex whether it will be a vertex of the triangulation or not, does not depend on a uniformly bounded neighborhood of this vertex; the DSS polygon or MLP are determined without such restriction within each slice (contour). Nevertheless, our triangle generation is strongly restricted to be performed within two consecutive slices, fact which applies to the method presented in (Yu and Klette, 2001) as well. In order to achieve global polyhedrization (and hence, multigrid convergence of the surface area estimator), intuitively, our triangles have to be given the chance to ''grow over various slices.'' Another possibility is the application of postprocessing to the resulting polyhedron, where triangles of similar surface normal are unified to form some new polygon. This is a known method used for the improvement of approximations and representations of surfaces (Wang et al., 2006), which of course can substantially increase the time complexity of the whole surface area estimation algorithm.

ACKNOWLEDGMENT

A preliminary version of this article was presented at the 13th International Workshop on Combinatorial Image Analysis (Wiederhold and Villafuerte, 2010). This research was supported by CONACYT Mexico, Grant CB-2009-01-128744. The authors express their gratitude to the three reviewers for their constructive and helpful remarks.

REFERENCES

H.N. Christiansen and T.W. Sederberg, Conversion of complex contour line definitions into polygonal mosaics, Computer Graphics: A Quarterly Report of SIGGRAPH-ACM, vol. 12, 1977, pp. 693–702.

A.B. Ekoule, F.C. Peyrin, and C.L. Odet, A triangulation algorithm from arbitrary shaped multiple planar contours, ACM Trans Graphics 10 (1991), 182–199.

H. Fuchs, Z.M. Kedem, and S.P. Uselton, Optimal surface reconstruction from planar contours, Commun ACM 20 (1977), 693–702.

S. Ganapathy and T.G. Dennehy, A new general triangulation method for planar contours, Comput Graph 16 (1982), 69–75.

N. Kehtarnavaz, L.R. Simar, and R.J.P. de Figueiredo, A syntactic/semantic technique for surface reconstruction from cross-sectional contours (Note), CVGIP 42 (1988), 399–409.

E. Keppel, Approximating complex surfaces by triangulation of contour lines, IBM J Res Dev 19 (1975), 2–11.

R. Klette and A. Rosenfeld, Digital geometry—geometric methods for digital picture analysis, Morgan Kaufman Publ., Elsevier, CA, U.S.A., 2004.

R. Klette and B. Yip, The length of digital curves, Mach Graph Vis 9 (2000), 673–703.

V.A. Kovalevsky, New definition and fast recognition of digital straight segments and arcs, Proc. Int. Conf. on Pattern Recognition (ICPR-10), IEEE Computer Society Vol. II, 1990, pp. 31–34.

Y. Shirai, Three-dimensional computer vision, Springer, Berlin Heidelberg, 1987.

F. Sloboda and J. Stoer, On piecewise linear approximation of planar Jordan curves, J Comput Appl Math 55 (1994), 369–383.

F. Sloboda and B. Zatco,'' On approximation of Jordan Surfaces in 3D,'' In Digital and image geometry, LNCS ,Bertrand et al. (Editors), 2001, vol. 2243, pp. 365–386.

G.J. Tee, Surface area and capacity of ellipsoids in n dimensions, NZ J Math 34 (2005), 165–198.

D. Wang, O. Hassan, K. Morgan, and N. Weatherill, Efficient surface reconstruction from contours based on two-dimensional Delaunay triangulation, Int J Numer Meth Eng 65 (2006), 734–751.

Y.F. Wang and J.K. Aggarwal, Surface reconstruction and representation of 3-D scenes, Pattern Recognit 19 (1986), 197–207.

P. Wiederhold and M. Villafuerte, ''Triangulation of cross-sectional digital straights segments and minimum length polygons for surface area estimation,'' In Progress in combinatorial image analysis, P. Wiederhold and R.P. Barneva (Editors), Research Publishing Services, Singapore, 2010, pp. 79–92.

L. Yu and R. Klette, An approximative calculation of relative convex hulls for surface area estimation, Proc. Image Vision Computing New Zealand (Int. Conf. IVCNZ'01, Nov. 2001, Dunedin, New Zealand), 2001, pp. 69–74.