Three-Dimensional Object Reconstruction from Layered Spatial Data

Michael Dangl and Robert Sablatnig
Vienna University of Technology,
Institute of Computer Aided Automation,
Pattern Recognition and Image Processing Group
Treitlstr. 3, 183-2, A-1040 Wien
fax: ++43-(0)1-58801 18392
{mll, sab}@prip.tuwien.ac.at, http://www.prip.tuwien.ac.at

Abstract

In this work, an approach for reconstructing 3D objects from layered spatial data points is presented. These data points are sampled at the surface of a world object using a 3D scanning system. In the first part, two algorithms for reconstructing triangle meshes from scanned objects are compared against each other showing different behavior concerning the quality and the number of reproduced triangles of the approximated model. Second, in order to allow fast visualization of the geometry, the mesh is optimized in respect of its number of triangles in a way that preserves the contour and the shape of the reconstructed object since reconstruction accuracy is more important than rendering quality in Computer Vision.

Key words: Surface Reconstruction, 3D-Acquisition, Marching Cubes, Cross-Section Triangulation, Mesh Optimization

1 Introduction

This work deals with the geometric three-dimensional reconstruction of objects in a scene. A scanning system delivers spatial data points along contours of cross-sections. These profile- (or cross-) sections describe the contour at a certain level. In simple cases, sections are ordered by the altitude on an axis; or they are distinguished by the degree of rotation during a rotational scanning process. In general, they represent the points of intersection from any plane with one (or several) object(s) [16].

The field of application is wide spread and covers areas such as medical scans (i.e. computer tomography (CT) and magnet resonance (MR) scans) or the industrial product domain where mechanical parts are designed by series of cross-sections. Along with proper visualization techniques, including texturing, color coding of various parts showing different features, the possibility of rotation or of hiding covering parts of the object's surface to gain an insight into hidden structures, crucial improvements are achieved where a 3D representation facilitates work. Generally, everywhere real illustrative material is not available or affordable a qualitative electronic visualization might replace the existence of a real object.

The process of gaining spatial data is based on the "shape from structured light" method, described in [1], [9], and [8]. Laser lines are projected onto the object, afterwards these lines are detected in the image, and finally, vertices are transformed back to spatial data using triangulation. As for our environment, this procedure is described in [15] and [13].

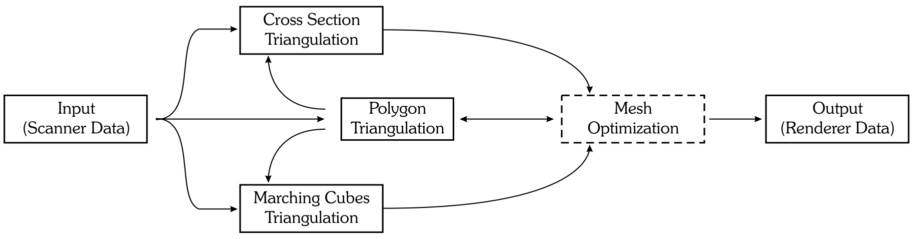

Figure 1: Work-flow of the reconstruction process.

Our approach and this paper are structured according to the work-flow of the reconstruction process, depicted in Figure 1. In Section 2, the surface of three-dimensional objects is geometrically reconstructed by triangulation of the input polygons (scanner data). Two algorithms, cross-section and marching cubes triangulation, are compared against each other, showing different behavior concerning the quality and the amount of triangles of the reconstructed objects. These two approaches were chosen among others ([10], [11] and [6]), since they profit from the characteristics of the input data, i.e. its organization in layers. Section 3 describes the mesh optimization step during reconstruction. For this step, an algorithm is chosen which allows control of the geometric deformation of the model while reducing the number of triangles based on a gradient criterion. Compared with other simplification algorithms ([20]) and level of detail (LOD) approximations ([17] and [3]), this approach allows an efficient optimization of the model while giving the possibility to control the committed error. Section 4 shows results computed out of synthetic and real data and the paper concludes with a discussion of the results and give an outlook on future work.

2 Reconstruction of Geometric Objects

Supplied with the knowledge of the origin and the characteristics of the sampled data points that define the surface of the scanned object in layers, a general definition of a scanner mesh can be given as follows:

- A *scanner mesh* (Figure 2a) is built up by a series of cross-sections. The route on which the scanning plane was moved over the object implicitly determines the order of the cross-sections. For some scanning procedures, this order is simply given along an axis; with respect to rotational cross- sections, the order of the series is equivalent to the degree of rotation.

- A *cross-section* (Figure 2b) combines a set of planar contours retrieved by the intersection between the scanning plane and the object at a certain level. To keep the approach as general as possible, this level of abstraction is needed to allow the representation of multi-contoured objects or scenes containing several objects.
- A *contour* (Figure 2c) is a planar polygon describing the shape of one object's contour at a certain level where the nodes of the polygon are ordered either clock wise or counter clock wise.

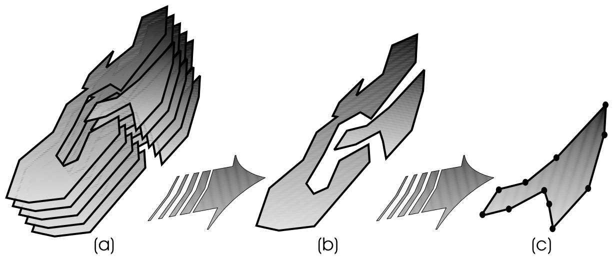

Figure 2: (a) definition of the scanner mesh, (b) cross-section, (c) contour.

2.1 Polygon Triangulation

As illustrated in Figure 1, polygon triangulation is of central concern during the reconstruction process. Several strategies serving this purpose are described for example in [18]. For selecting an appropriate algorithm, let us first consider the demands of our application. No a-priori knowledge about the nature of the polygon can be assumed, that is, the algorithm has to work on any kind of polygon (convex and concave). Nevertheless, regarding the mode of operation of the scanning process, it is assured that such a polygon is free of intersection (Figure 3a), since, in order to produce a self-intersecting polygon, the scanning sensor must be able to sample the surface through the body of the object (Figure 3b). Furthermore, we do not allow the creation of new vertices on the polygon contour or inside the polygon to keep the visualization cost as low as possible.

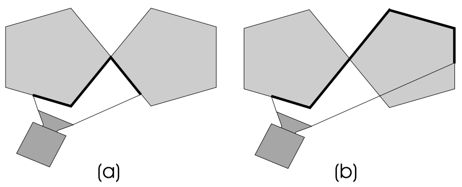

Figure 3: (a) the scanning device samples the surface of the objects, no polygon with intersection, (b) self-intersecting polygon, but impossible because the scanning device cannot sample through the object.

Our heuristic approach is called "smallest inner angle first" which holds these demands. The algorithm systematically cuts off triangles from the polygon choosing the point with the smallest inner angle to be the one to be removed from the polygon contour after building a triangle with its predecessor and successor node. Since the polygon is free of intersections, neither additional faces deform the contour of the model, nor vertices are dropped without being a member of a face. Figure 4 subsequently depicts the whole triangulation process for a simple polygon. In each step, the node with the smallest inner angle (bold) is selected and spans up a new triangle (dark shaded).

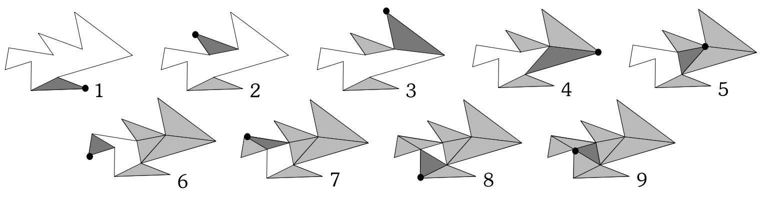

Figure 4: Steps during polygon triangulation. In each step, a new triangle (dark shaded) is created around the polygon node having the smallest inner angle (bold).

Figure 9d shows the result of the polygon triangulation algorithm applied on the contour polygon of the cerebrospinal fluid segmented from a CT scan, where we can notice one drawback of this algorithm: the occurrence of long triangles with an acute angle. To overcome this, the polygon re-tiling strategy proposed in [22] may be used.

2.2 Triangulation between Cross-Sections

When examining cross-section triangulation, the actual problem can be formulated as follows [10]: for the bottom and top level cross- sections, reconstruct their surface by applying the polygon triangulation algorithm. Afterwards, for any pair of adjacent cross-sections, find a set of triangles (triangle-fan) which approximate the surface between these polygons with a minimal error according to the real surface of the object. Furthermore, the latter problem can be split up into the following two sub-problems:

- An initial pair of vertices (P_i,Q_i) has to be found in order to define a start for the triangulation process.
- For a given pair of vertices P_j and Q_j, the next point P_{j+1} or Q_{j+1} must be determined to define a triangle of the surface, together with P_i and Q_i.

The selection criterion for the two vertices forming the initial pair of the triangulation algorithm must not be underestimated, since the quality of the triangulation process depends on these starting points (an inconvenient pair will result in distorted triangles in the triangle fan). A convenient initial pair should show similar features in respect with their polygons.

The direction from the centroid C to a point is a stable feature if the distance from the point to the centroid is large (Figure 5a). Let u_i and v_i be vectors from the centroid to a point of the polygon's contour in the lower and upper cross-section. Now, the initial pair (P_i,Q_j) is selected such that the inner product of the vectors $u_i \times v_j$ is a maximum. This guarantees an initial pair with similar features, i.e. peaks and

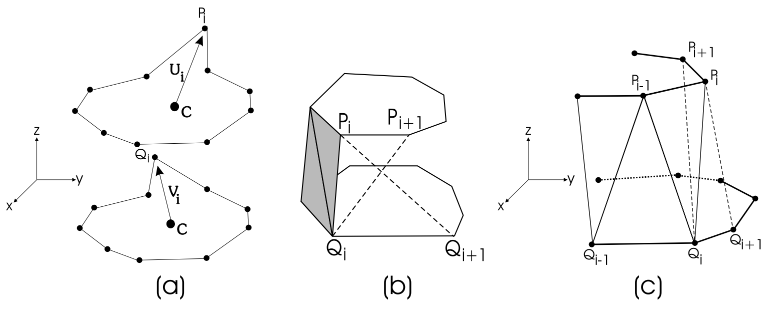

Figure 5: (a) selection of the initial pair for cross-section triangulation, (b) finding the next vertex, (c) selection of the next vertex.

direction of edges to neighboring polygon nodes. The direction of vector v_i is taken into account, for the search for the maximum of the inner product, only nodes similar to Q_i are regarded.

Assume that the triangulation process has stopped after the last step leaving P_i and Q_i to be the points that are used for building the next triangle in the object's surface. Now either of the two vertices P_{i+1} or Q_{i+1} could be selected as the third point in the triangle. If P_{i+1} would be selected, the triangle (P_i, Q_i, P_{i+1}) is generated and the resulting pair for the next triangulation step is (P_{i+1}, Q_i). An intuitive criterion for this selection might be the length between the vertices, either $\overline{P_{i+1}Q_i}$ or $\overline{P_iQ_{i+1}}$. Choosing the triangle with the minimum area seems to be more natural since the original surface is approximated smoother. As for the situation in Figure 5b, $\overline{P_{i+1}Q_i}$ is shorter than $\overline{P_iQ_{i+1}}$ and (P_i, Q_i, P_{i+1}) is designated to be processed next, leaving (P_{i+1}, Q_i) the next pair.

However, a criterion based on length only, fails to reproduce a natural-looking surface, if the horizontal position of the vertices is quite different (see Figure 5c). The current pair is (P_i, Q_i) and the x coordinate of P_i is larger than the x coordinate of Q_i. Now $\overline{P_{i+1}Q_i}$ might be shorter than $\overline{P_iQ_{i+1}}$ and (P_i, Q_i, P_{i+1}) would be selected as the next triangle. Despite the length, the difference of direction is taken into account, because it is a more natural way for approximating the original surface. Let the difference in direction between $\overline{P_iP_{i+1}}$ and $\overline{Q_{i-1}Q_i}$ be denoted by α_i and the one between $\overline{P_{i-1}P_i}$ and $\overline{Q_iQ_{i+1}}$ be denoted by β_i. Now, the selection algorithm in the case of $\overline{P_{i+1}Q_i} < \overline{P_i, Q_{i+1}}$ determines the next pair as follows:

$$\begin{array}{lll} \text{If} & \cos\alpha_i > t & \text{select } P_{i+1} \\ \text{otherwise if} & \cos\beta_i > t & \text{select } Q_{i+1} \\ & \text{else} & \text{select } P_{i+1} \end{array}$$

where t is a threshold to handle the difference between the curvature of the lower and upper cross-section. The results show a more natural approximation of the surface although triangles might become larger.

2.3 Marching Cubes Triangulation

One drawback concerning the triangulation with the angle criterion approach, is its inability to handle multi-contoured structures. Since real world objects are single-contour primitives only rarely, complex handling of such structures, e.g. combining simple objects to shape a more complex model, is needed. The *Marching Cubes* algorithm, introduced by William E. Lorensen and Harvey E. Cline [14], was designed to extract surface information from a 3D field of values and to overcome these difficulties.

In order to show the method, first a two-dimensional equivalent of this approach is presented, know as *Marching Squares* and afterwards the procedure is expanded into 3D. Given a planar polygon (Figure 6a), the object's coordinate system is discretized with a regular grid, called the resolution of the Marching Squares algorithm (Figure 6b). Afterwards, each corner of every grid element is examined whether it lies inside or outside the polygon (Figure 6c). Now, from a lookup-table the algorithm extracts the information for each of the 24 possible cases for the corner matching the number and position of faces (if any) which represents the underlying polygon best. Finally, the source polygon becomes triangulated like given in Figure 6d. Nevertheless, this triangle set looks like a too-rough approximation of the original circle.

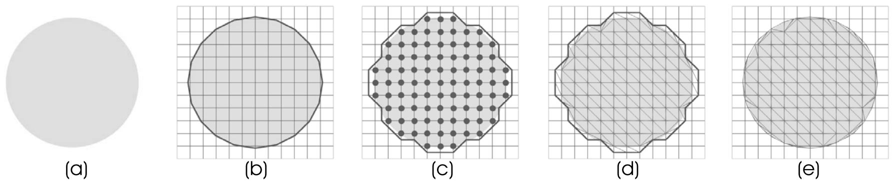

Figure 6: (a)-(d) steps during marching squares triangulation, (e) interpolation result.

One obvious method to improve the accuracy of the model is to increase the resolution of the raster, but this will lead to an exponential increase of triangles. A higher quality mesh with the same number of faces can be gained by applying interpolation inside the lookup square. Consider the square in Figure 7a. Corners 1 and 4 lie inside the object's contour, 2 and 3 do not. As a first step, the "marching square" suggests to generate face a and b which describe the actual shape of the polygon in the current raster square unsatisfactory (Figure 7b).

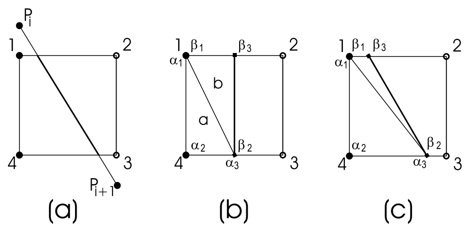

Figure 7: Edge interpolation inside a marching square (bold edges indicate membership of contour line - see text for details).

Since we know the real coordinates of P_i and P_{i+1} of the original polygon, we are able to interpolate the coordinates of the vertices β_2 and β_3 of face b and α_3 of face a along the square's edges in order to obtain a more accurate approximation (Figure 7c). Figure 6e shows a triangle-mesh gained by the interpolating Marching Squares algorithm applied to the polygon Figure 6a.

Now, having worked out the basic mechanism in two dimensions, we can consider the third dimension. Obviously, the former squares turn into cubes, giving rise to 28 possible cases for inside/outside corner combinations, which can be reduced to 15 base classes. The cube "marches" through object space returning a surface approximating set of triangles. Again, interpolation of vertex coordinates improves the quality of the model increasingly. If possible, interpolation is done along all three axis.

The most powerful feature of the Marching Cube algorithm is its easiness to handle complex models, although the results are less accurate than using cross-section triangulation. The most aggravating drawback is the occurrence of huge sets of triangles.

3 Mesh Optimization

Triangulation algorithms perform a straightforward process on their input data and tend to result in (typically heavily) over-tessellated meshes. Such meshes throw up problems concerning rendering, storing and transmission due to the enormous number of triangles they are built of. Therefore, decimation of triangles in a triangular mesh is required. One approach to speed up rendering is to replace a complex object by a level of detail (LOD) approximation (see [3], [7], [5] and [2] for examples).

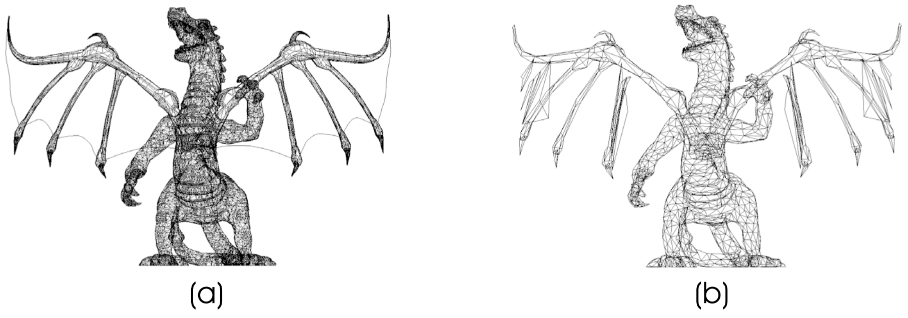

Figure 8: (a) original dragon mesh with 54.000 triangles, (b) LOD approximation with 10.000 triangles.

However, towards our field of application where it is crucial to gain a precise representation rather than guarantee a high and/or fixed frame rate while rendering, these LOD approximations show a severe drawback (Figure 8): they might change both, the geometrical (Figure 8a) and topological (Figure 8b) features of the model. For the geometrical aspect, no reliable measure for the resulting contour change can be given. The topological transformations can result in converting non-manifold meshes into manifold counterparts. A triangle reduction function has to be found that preserves both, the non-manifold property and the contour of the input model. In [23], a method for reducing the surface description of triangulated models by adaptation to the object contour is presented. The error between the original and the simplified mesh is measured by the deviation of the spatial curvature. This allows approximations from a model where only redundancy is removed, i.e. triangles with equal normal vectors are merged, up to meshes showing a similar contour like their original.The approach splits into four steps:

1. **Creation of Free Polygons**: During this step, a region growing process is performed. For each triangle of the object's, which is not already a member of any region, examine its neighbors and add them to the same region, if the following two conditions hold:

   - Each triangle of a region must have a common edge with another triangle of the same region.
   - For each pair of triangles out of one region, the angle between their normal vectors is less than a specified threshold a.

   Threshold a determines the curvature of a region. Using a small a, the region is nearly planar. According to an increasing value of a, the region will be more curved and the final approximation will become rough. If $a = 0$, the optimized model will erase redundancy in the original mesh, i.e. coplanar triangles will merge to form a region.

2. **Projection of Polygons**: In order to make re-triangulation possible in the last step of optimization, projecting a region onto a plane is required. The projection plane is defined by an average normal vector, which, according to [4], can be calculated as follows: Let m be the number of nodes in polygon P, $V_i = (x_i, y_i, z_i)$ be a node and j the successor index of i, $j = i + 1, i < m - 1$ and $j = 0$ with $i = m - 1$. The average normal vector of P is defined by

$$N_x = \sum_{i=1}^{m}(y_i - y_j)(z_i + z_j),\ N_y = \sum_{i=1}^{m}(z_i - z_j)(x_i + x_j),\ N_z = \sum_{i=1}^{m}(x_i - x_j)(y_i + y_j) \quad (1)$$

   If the projected representation is not free of intersection, the region is left out of consideration and cannot be optimized.

3. **Deleting Vertices from Polygon Contour**: Another possibility for optimization is the elimination of vertices from the polygon contour that form a line with their predecessor and successor nodes. By deleting such a point, the model is decimated by one face. If for all m regions containing a point P the condition above is valid, the optimization saves m triangles and additionally decreases the number of vertices of the mesh by one.

4. **Retriangulation**: Finally, retriangulation of these free polygons with the algorithm presented in Section 2.2 transforms all regions back to a homogenous, triangulated surface.

4 Results

Figure 9a and Figure 9d represent the result from the polygon triangulation algorithm from Section 2.1 applied to a polygon of the cerebrospinal fluid segmented from a CT scan image. The original polygon consists of 154 polygon nodes, the triangulated mesh contains of 152 triangles, since no new vertices are generated. Figure 9b and Figure 9e demonstrate the cross-section algorithm described in Section 2.2. The