Technical Report

Pattern Recognition and Image Processing Group
Institute of Automation
Vienna University of Technology
Treitlstr. 3/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-8161
Fax: +43 (1) 5054668
E-mail: mll@prip.tuwien.ac.at

PRIP-TR-50 June 16, 1998

Three-Dimensional Object Reconstruction form Layered Spatial Data

Michael Dangl

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 3D 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 of the object.

CONTENTS

1 Introduction 3
2 Reconstruction of Geometric Objects 4
2.1 Polygon Triangulation 5
2.2 Triangulation between Cross Sections 6
2.2.1 Determination of Triangulation Start Vertices 6
2.2.2 Selection of the Next Vertex 7
2.3 Marching Cubes Triangulation 8
3 Mesh Optimization 10
4 Results 12
5 Conclusion 12
6 Acknowledgements 14
7 References 15
Appendix A (Implementation Details) 17
A.1 Implementation 17
A.2 Usage of Tasks 17
A.2.1 Angle Criterion Triangulator 17
A.2.2 Marching Cubes Triangulator 18
A.2.3 Normal Optimizer 19
A.2.4 3D Geometry File Converter 19
A.3 Description of Profile File Format 20
Language Definition 20
A.4 Description of the Mesh File Format 21
Language Definition 21
A.5 Installation 21

LIST OF FIGURES

FIGURE 1 – DIGITAL IMAGE PROCESSING PIPELINE 3
FIGURE 2 – WORKFLOW OF THE RECONSTRUCTION PROCESS 4
FIGURE 3 – DEFINITION OF SCANNER MESH (A), CROSS SECTION (B) AND CONTOUR (C) 4
FIGURE 4 – A: THE SCANNING DEVICE SAMPLES THE SURFACE OF THE OBJECTS, NO POLYGON WITH INTERSECTION B: WOULD RESULT IN INTERSECTING POLYGON, BUT IMPOSSIBLE BECAUSE THE SCANNING DEVICE CANNOT SAMPLE THROUGH THE OBJECT 5
FIGURE 5 – STEPS DURING POLYGON TRIANGULATION. IN EACH STEP, A NEW TRIANGLE (DARK SHADED) IS CREATED AROUND THE POLYGON NODE SHOWING THE SMALLEST INNER ANGLE (BOLDED). 6
FIGURE 6 – SELECTION OF THE INITIAL PAIR FOR CROSS SECTION TRIANGULATION 7
FIGURE 7 - FINDING NEXT VERTEX 7
FIGURE 8 – SELECTION OF NEXT VERTEX 8
FIGURE 9 – MULTI-CONTOUR SCANNER MESH 8
FIGURE 10 – STEPS DURING MARCHING SQUARES TRIANGULATION 9
FIGURE 11 – LOOKUP TABLE FOR MARCHING SQUARES 9
FIGURE 12 – EDGE INTERPOLATION INSIDE A MARCHING SQUARE (BOLDED EDGES INDICATE MEMBERSHIP OF CONTOUR LINE) 9
FIGURE 13 – MARCHING CUBE CASES 10
FIGURE 14 – ORIGINAL DRAGON MESH WITH 57.000 TRIANGLES AND ONE LOD APPROXIMATION WITH 15.000 TRIANGLES 11
FIGURE 15 – RECONTRUCTION RESULTS 13
FIGURE 16 - ANGLE CRITERIA TRIANGULATOR USER INTERFACE 17
FIGURE 17 - MARCHING CUBES TRIANGULATOR USER INTERFACE 18
FIGURE 18 - NORMAL OPTIMIZER USER INTERFACE 19
FIGURE 19 – 3D FILE FORMAT CONVERTER USER INTERFACE 20

1 INTRODUCTION

Regarding the digital image analysis process (Figure 1), this work is settled around the "Object Model" step. In this stage, after having improved image quality, extracted image features and segmented regions and specific parts we want to extract models of the objects depicted within the digital image. From the point of the 3D Vision field of research, this can be seen as a generation of three-dimensional geometric representations of the image models. Spatial data retrieved from a 3D scanning system is one possible way of sampling a surface, using approximated points organized in cross sections. The field of application is wide spread and covers areas such as medical scans (i.e. computer tomography (CT) and magnect resonance (MR) scans) or the industrial product domain where mechanical parts are designed by series of cross sections.

Along with proper visualization techniques, crucial improvements at many tasks are achieved where a 3D representation facilitates work (i.e. 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). Concrete fields of application are: three-dimensional surfaces of the anatomy that offer a valuable medical tool. They help physicians to understand the complex human anatomy, provide a fundamental base for organizing medical operations, and support the detection of anomalies inside the human body. Another field of interest covers the surveillance of specific features of objects. For example, while surveying human skulls, the depth of the orbital cavity might be of interest which is simplified by a sophisticated 3D visualization. An increasing number of companies advertise for and sell their (haptic) products in electronic media. About ten percent of the Fortune-500 companies use or develop interactive virtual worlds for several economic purposes, like customer support, advertising, presentations or education ([19]). Consider the impact in both, the customer's and the producer's benefit if products are presented as an accurate 3D model on, for example, the Internet. Furthermore, in many branches of education, an interactive spatial model may stimulate the learning process. Generally, everywhere real illustrative material is not available or affordable, a qualitative electronic visualization might replace the existence of a real object.

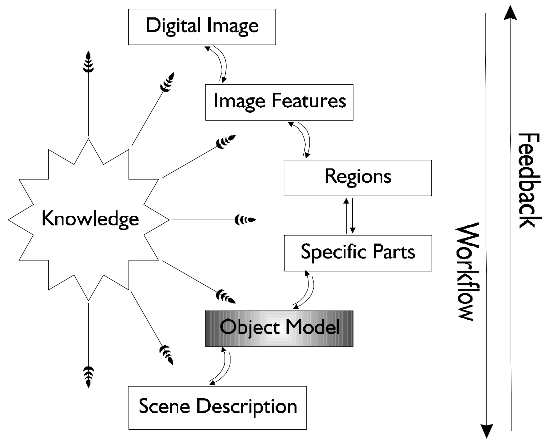


Figure 1 – Digital Image Processing Pipeline

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 profiles (or cross sections) describe the contour of an object at a certain level. In simple cases, sections are ordered by the altitude on any 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).

The goal of the presented algorithms is to reconstruct a three dimensional representation out of 2D projections of real objects. The process of gaining spatial data is based on the “shape from structured light” method, described in [16]. 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 [12] and [10]. This approach is separated into two parts according to the workflow of the reconstruction process, depicted in Figure 2.

In Section 2, the surface of three-dimensional objects is geometrically reconstructed by triangulation of the input (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 ([8], [9] and [7]), since they profit from the characteristics of the input data, i.e. it's organization in layers.

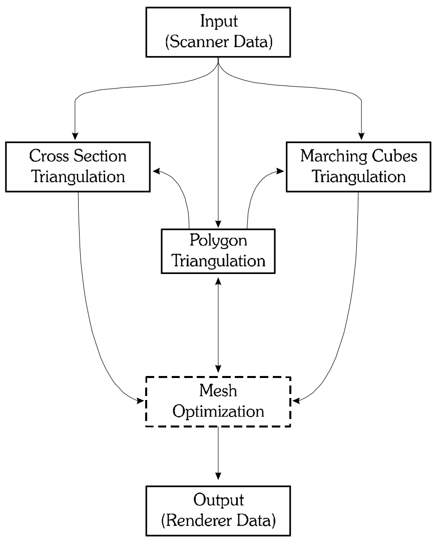


Figure 2 – Workflow of the Reconstruction Process

Section 3 describes the (optional) mesh optimization step during the reconstruction process. For this task, 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 ([15]) and level of detail (LOD) approximations ([13], [2]), this approach allows an efficient optimization of the model while giving the possibility to control the committed error.

2 RECONSTRUCTION OF GEOMETRIC OBJECTS

This section shows an approach for geometrically reconstructing an object (triangle mesh) out of scattered spatial data points from the surface of the object delivered by a sequential scanning system. 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 (Figure 3):

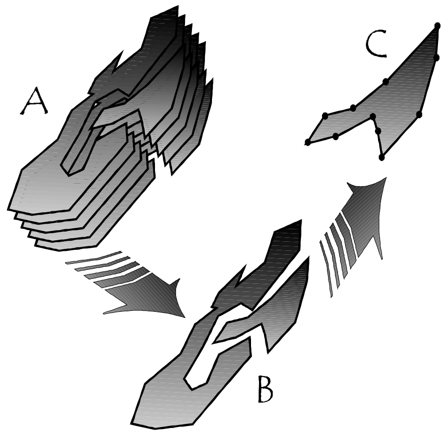


Figure 3 – Definition of Scanner Mesh (A), Cross Section (B) and Contour (C)

A *scanner mesh* (Figure 3A) 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.

Next, a *cross section* (Figure 3B) 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.

Finally, a *contour* (Figure 3C) 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.

In Section 2.1, a method for triangulating planar polygons is described. This algorithm is used to fulfill three different tasks:

- Reconstruction of the surface of the top and bottom level cross section of a scanner mesh, if a closed surface of the object is desired.
- The descretizing step of the marching cubes (the algorithm for testing whether a points lies inside or outside a polygon) can be speed up if the polygon is separated into (convex) triangles (Section 2.3).
- While optimizing the mesh, we will need a triangulation algorithm the transform a polygon back into a triangulated representation (section 3).

Section 2.2 presents a method for the reconstruction of the surface between to adjacent profiles, based on a local angle criterion. Finally, Section 2.3, describes the marching cubes triangulation algorithm and its ability to process multi-contour objects.

2.1 Polygon Triangulation

As illustrated in Figure 2, polygon triangulation is of central concern during the reconstruction process. Several strategies serving this purpose are described for example in [14]. For selecting an appropriate algorithm, let us first consider the demands of our field of 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 4A), 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 4B). 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.

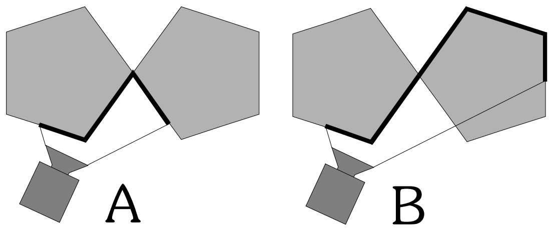


Figure 4 – A: the scanning device samples the surface of the objects, no polygon with intersection
B: Would result in intersecting polygon, but impossible because the scanning device cannot sample through the object

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 5 subsequently depicts the whole triangulation process for a simple polygon. In each step, the node with the smallest inner angle (bolded) is selected and spans up a new triangle (dark shaded).

Considering the cost of this algorithm, calculation time for *N+2*(N-4)* inner angles (where *N* is the total number of vertices in the polygon, *N>3*). *N* inner angles are initially calculated. Additionally, *2*(N-4)* calculations are needed because after each creation of a new triangle, 2 inner angles must be recalculated (as long as the polygon contains more than *4* vertices). Furthermore, the overhead for searching *N-3* times the smallest angles in a set of $\frac{n*(n-1)}{2}$ vertices is needed (*n=N(-1)4*, the number of current nodes forming the polygon, *N>4*).

Figure 15D 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 a drawback of this algorithm: the occurrence of long triangles with an acute angle. This influences the visualization of the geometry since such triangles may produce aliasing effects during rendering (shading). To overcome this, the polygon re-tiling strategy proposed in [17] may be used.

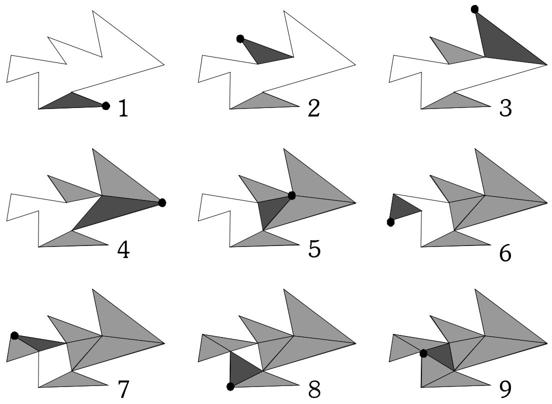


Figure 5 – Steps during polygon triangulation. In each step, a new triangle (dark shaded) is created around the polygon node showing the smallest inner angle (bolded).

2.2 Triangulation between Cross Sections

When examining cross section triangulation, the actual problem can be formulated as follows [8]: for the bottom and top level cross sections, reconstruct their surface by applying the polygon triangulation algorithm described in Section 2.1. 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: first, an initial pair of vertices (P_i and Q_i) has to be found in order to define a start for the triangulation process, described in Section 2.2.1. Second, 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), shown in Section 2.2.2.

2.2.1 Determination of Triangulation Start Vertices

The selection criterion for the two vertices forming the initial pair of the triangulation algorithm must not be underestimated, since the quality of the whole triangulation process depends on these starting points (an inconvenient pair will result in distorted triangles in the triangle fan). A convenient initial pair should be similar from various points of view. That is, they have to show similar features in respect with their polygons. Candidate features are the position of the vertex, the direction of the edges connecting a vertex with its neighboring vertices or the direction of a vertex to the centroid of its polygon.

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 6). Actually, both, the direction and the distance, have to be considered when determining the initial pair.

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*V_j is a maximum. Then it is guaranteed to find an initial pair with similar features, i.e. peaks and direction of edges to neighboring polygon nodes. The actual selection works as follows: first, in one contour the polygon node with the largest distance to the centroid is determined (for example Q_i in Figure 6).

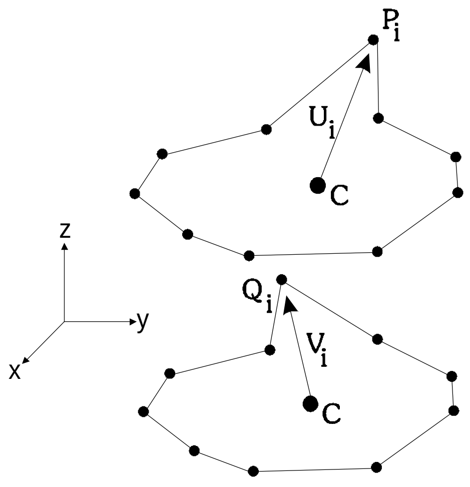


Figure 6 – Selection of the initial pair for cross section triangulation

Afterwards, in order to find the member of the initial pair in the adjacent contour, we obviously cannot search through all of the nodes in the latter contour, since the maximum might lie in a position where the resulting triangle may be unusable distorted. Therefore the direction of vector V_i is taken into account and for the search for the maximum of the inner product, only nodes lying in a similar direction as Q_i are regarded.

2.2.2 Selection of the Next Vertex

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 $P_iQ_iP_{i+1}$ is generated and the resulting pair for the next triangulation step is P_{i+1} and Q_i.

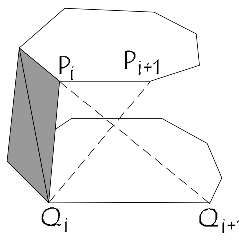


Figure 7 - Finding next vertex

An intuitive criterion for this selection might be the length between the next pair, either $P_{i+1}Q_i$ or P_iQ_{i+1}. Choosing the triangle with the minimum area seems to be more natural since the original surface is approximated smoother rather than with larger triangles. As for the situation in Figure 7, $P_{i+1}Q_i$ is less than P_iQ_{i+1} and $P_i\ Q_i\ P_{i+1}$ is designated to be processed next, leaving ($P_{i+1,}Q_l$) the next pair.

However, a criterion based on length only, might fail to reproduce a natural-looking surface, especially when the horizontal position of the vertices is quite different. Regarding the triangulation process in Figure 8, where the current pair is P_i and Q_i and the *x* coordinate of point P_i is larger than the one of Q_i. Now the length of P_{i+1} and Q_i might be shorter than P_i and Q_{i+1} and $P_iQ_iP_{i+1}$ would be selected as the next triangle. Nevertheless, because of the deviation of the direction between P_iP_{i+1} and Q_iQ_{i+1}, a more natural way for approximating the original surface is the selection of Q_{i+1} instead of P_{i+1}, although the resulting triangle's area is larger. In fact, despite the length of the next pair, the difference of direction is taken into account. Let the difference in direction between P_iP_{i+1} and $Q_{i-1}Q_i$ be denoted by α_i and the one between $P_{i-1}P_i$ and Q_iQ_{i+1} be denoted by β_i.

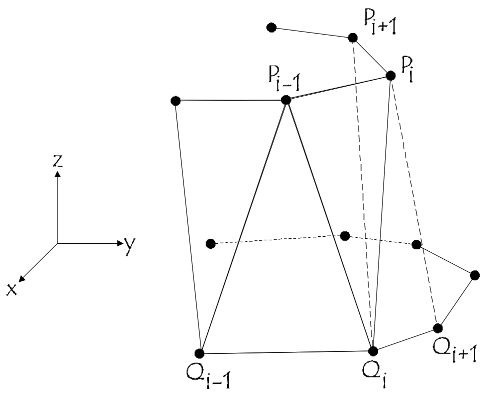


Figure 8 – Selection of next vertex

Now, the selection algorithm in the case of $P_{i+1}Q_i < P_iQ_{i+1}$ determines the next pair as follows:

If	$\cos \alpha_i > t$	select P_{i+1}
Otherwise, if	$\cos \beta_i > t$	select Q_{i+1}
	else	select P_{i+1}

where *t* is a threshold.

With the threshold parameter *t* we are able to handle the difference between the curvature of the lower and upper cross section and care for peaks in concave polygons. The results show a more natural approximation of the surface although triangles might become larger.

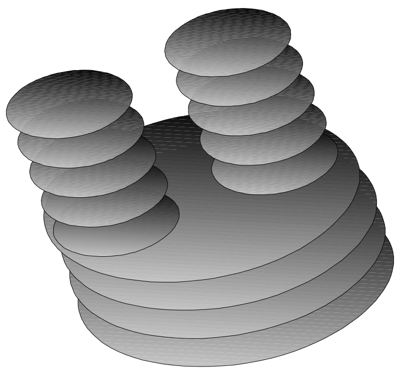


Figure 9 – Multi-contour Scanner Mesh

So far, this algorithm produces accurate triangular meshes. Nevertheless, consider the scanner mesh in Figure 9. Obviously, we lack information about the inner vertices of the lower contour, which build the surface along with the inner vertices of the upper (smaller) contours. That is, we need a different approach to overcome this problem with unequal multi-contoured objects.

2.3 Marching Cubes Triangulation

As outlined above, one drawback concerning the triangulation with the angle criterion approach, is its inability to handle multi-contoured structures in an easy manner. Unfortunately, since real world objects are single-contour primitives rarely, this throws up the need for complex handling of such structures, e.g. combining simple objects to shape a more complex model. The Marching Cubes algorithm, introduced by William E. Lorensen and Harvey E. Cline ([11]), was designed to extract surface information from a 3D field of values and to overcome these difficulties. In order to show the method of work, first a two-dimensional equivalent of this approach is presented, which is also know as "Marching Squares" and might be used for triangulation of polygons, and afterwards the procedure is expanded into 3D.

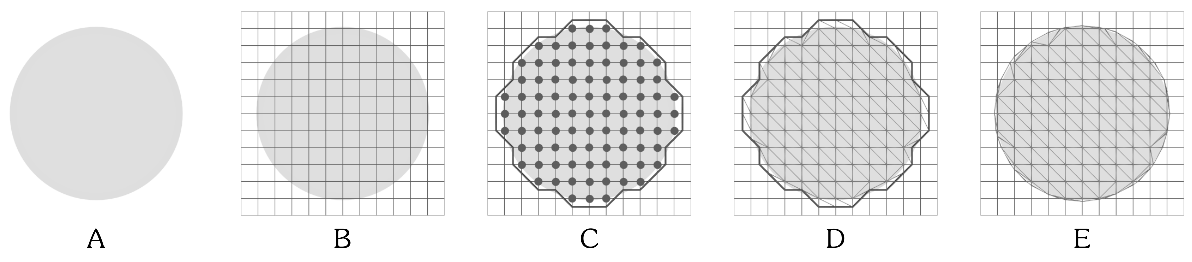


Figure 10 – Steps during Marching Squares Triangulation

Given a planar polygon (Figure 10A), the object's coordinate system is discretized with a regular grid. This is called the resolution of the Marching Squares algorithm or the number of steps the square will "march" along each axis through space (Figure 10B). Afterwards, each corner of every grid element is examined whether it lies inside or outside the polygon (Figure 10C).

Now, from a lookup-table the algorithm extracts the information for each of the 2^4 possible cases for the corner matching the number and position of faces (if any) which represents the underlying polygon best. However, these 16 possibilities can be reduced by rotation, mirroring and inversion to a set of four possible cases, shown in Figure 11. Finally, the source polygon becomes triangulated like given in Figure 10D. Nevertheless, this triangle set looks like a too-rough approximation of the original circle rather than a sophisticated model. That is, because so far, a Discrete Marching Square algorithm was applied. For some tasks, this may be a satisfying result because of execution time considerations.

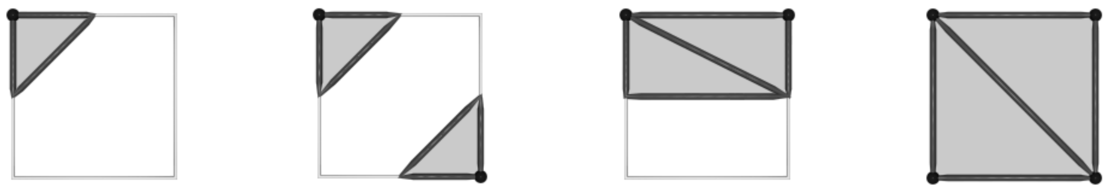


Figure 11 – Lookup Table for Marching Squares

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 face can be gained by applying interpolation inside the lookup square. Consider square A in Figure 12. 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 (square B).

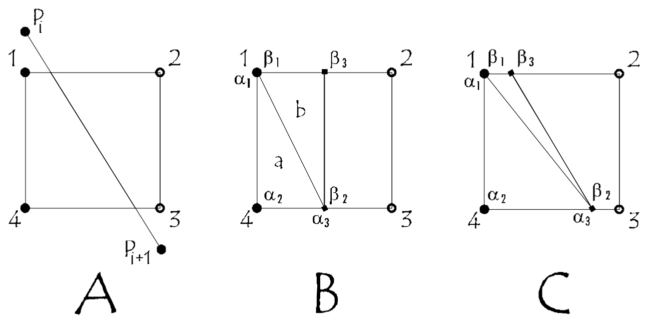


Figure 12 – Edge interpolation inside a Marching Square (bolded edges indicate membership of contour line)

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 (square C in Figure 12). Figure 10E shows a triangle-mesh gained by the interpolating Marching Squares algorithm applied to the polygon in Figure 10A.

Now, having worked out the basic mechanism in two dimensions we can consider a third dimension. Obviously, the former squares turn into cubes, giving rise to 2^8 possible cases for inside/outside corner combinations, which can be reduced to 15 base classes, shown in Figure 13. Here, 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. In some cases, like case 2 in Figure 13, there is no existing reference point for the *z* axis and therefore the point dividing the edge in the middle is used.

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

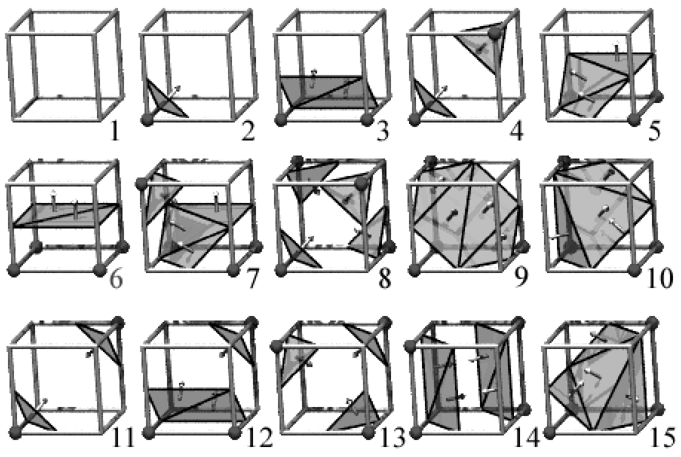


Figure 13 – Marching Cube Cases

3 *MESH OPTIMIZATION*

Modeling and scanning systems commonly give rise to triangle meshes of high complexity. On the other hand, triangulation algorithms, such as the Marching Cubes algorithm, 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. Since the purpose of meshes reconstructed from scans typically lies in the field of observation and surveillance, the must allow to be rendered, rotated or scaled (nearly) in real time to allow efficient work. 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. These propotions arose in the field of computer graphic where they are employed to allow real time rendering of a scene. Some recent work was done by [2], [6], [5] and [1].

However, towards our field of application of the resulting model, 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 14, A + B): they might change both, the geometrical (A) and topological (B) features of the model. As 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. In fact, a triangle reduction function has to be found that preserves both, the non-manifold property and the contour of the input model.

In [18], 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.

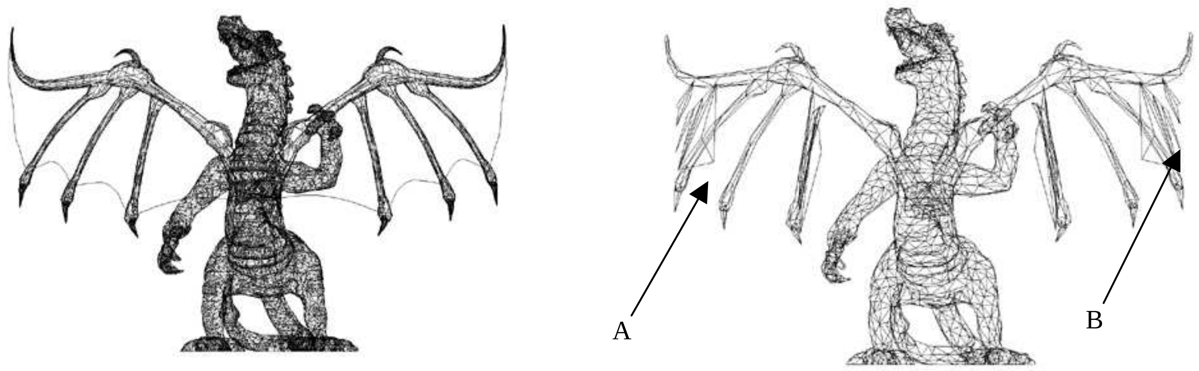


Figure 14 – Original Dragon Mesh with 57.000 triangles and one LOD approximation with 15.000 triangles

Generally, this approach splits into four steps:

- **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 α.

  The threshold α parameter determines the curvature of a region. Using a small value for α, the region is nearly planar. According to an increasing value of α, the region will be more curved and the final approximation will become rough. If α equals zero, the optimized model only will erase redundancy in the original mesh, i.e. only coplanar triangles will merge to form a region, resulting in the most accurate simplified mesh.

- **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 [3], can be calculated as follows: Let m be the number of nodes in a 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

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

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

- **Deleting Vertices from the 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.

- **Retriangulation:** finally, retriangulation of these free polygons with the algorithm presented in Section 2.1 transforms all regions back to a homogenous, triangulated surface.

4 *RESULTS*

Figure 15A+D 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.

Picture B and E in Figure 15 demonstrates the cross section algorithm described in Section 2.2. The input data are two adjacent cross sections of an antique vase gained by a rotational scanning procedure.

Figure 15C shows the wireframe model of a workpiece containing a hole in order to demonstrate the ability of the marching cubes algorithm to handle multi-contoured objects. Since this reconstruction method usually delivers an over-tessellated (redundant) surface description, the optimization algorithm presented in Section 3 reduces the number of triangles from 6.240 vertices/12.480 faces to 94 vertices/188 faces with a normal vector threshold of 5 degrees (Figure 15F).

Since one special issue of our approach is the three-dimensional reconstruction of archeological finds ([12] and [10]), Figure 15G-L presents images of 3D reconstruction of such objects.

	#vertices	#faces	reconstruction time[†]	optimization time
Figure 15 G&J	13,854	27,729	2.35 sec	n/a
Figure 15 H&K	37,110	74,220	3.87 sec	n/a
Figure 15 I&L	6,185	12,372	n/a	12.23 sec

5 *CONCLUSION*

The scanner mesh reconstruction algorithms presented in this work show different behavior concerning ability of handling input data, model accuracy, and performance. The actual goal of reconstruction, i.e. the field of application of the model, determines the usage of a proper algorithm. Cross section triangulation offers a fast, accurate surface reconstruction on its input data but lacks the ability to handle multi-contoured objects properly. The Marching Cubes Algorithm provides a fast, straightforward treatment of its input data, based on a local decision criterion ("greedy" algorithm). It can be applied to parallel execution easily and need no assumptions about the objects. Problems occur relating to enormous number reconstructed triangles and its less precise reconstructed surface, due to discretizing and interpolation. The former problem can be solved by a mesh optimizer, like the Normal Optimizer, which simplifies an object while preserving its contour (conservative settings) or allows to control the error committed to the contour of the object. Additionally, to gain platform-independed models, our implementation delivers VRML models of the reconstructed objects. For any further work, we could consider taking the object's texture into account, while triangulating and optimizing the input cross sections.

[†] Performance measurements relate to a SGI O2 180Mhz R5000 /w 64 MB RAM

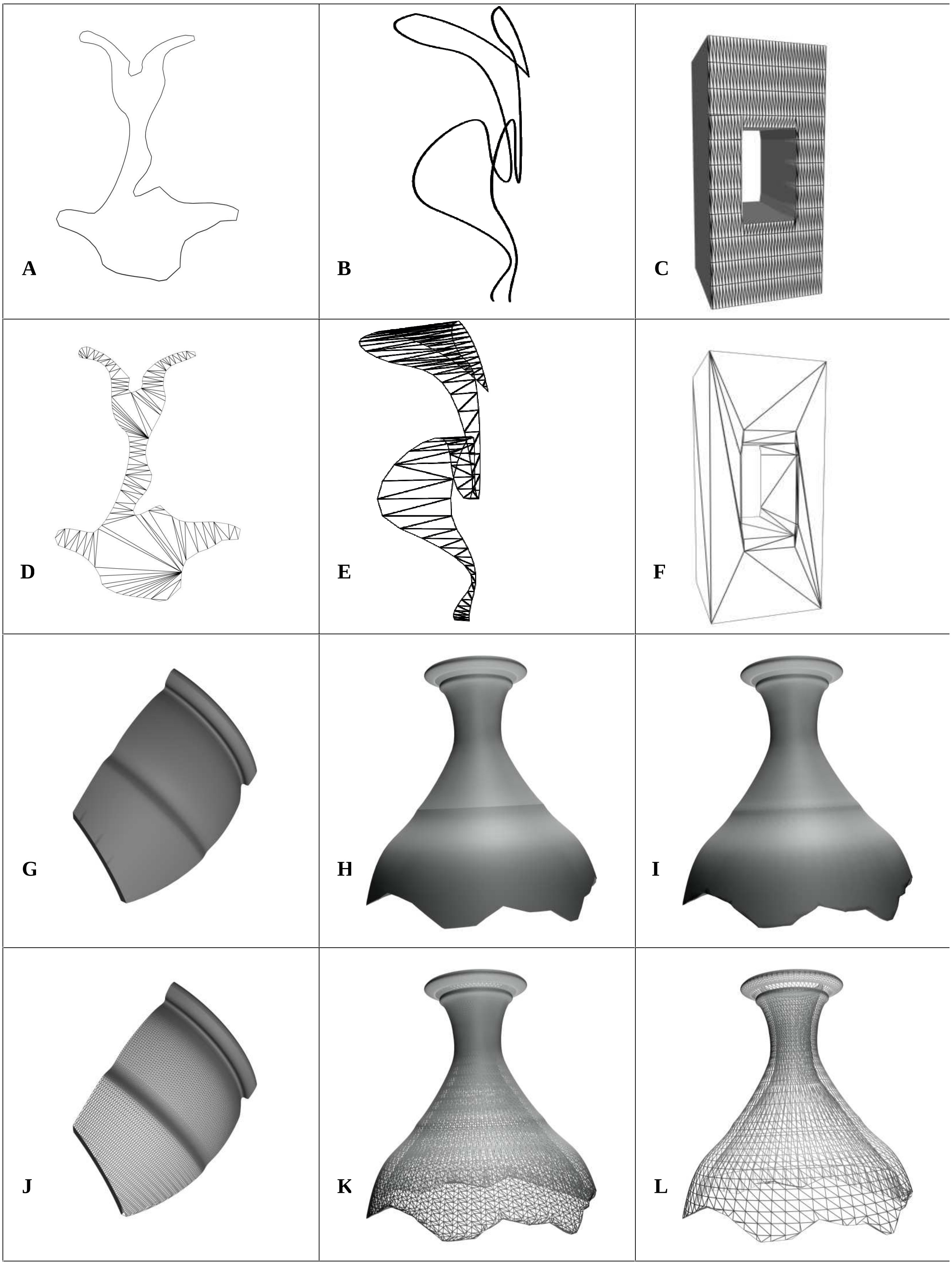


Figure 15 – Recontruction results

6 ACKNOWLEDGEMENTS

This work would not have been possible without the guidance, ideas, advice, and support of

Robert Sablatnig,
for steady supervision and reviewing tasks and moral support,

Christian Menard,
for initially making this work possible and supervision,

Horst Bischof,
for technical guidance with Khoros and Postscript and providing awakening coffee,

and many other peoples at the Pattern Recognition and Image Processing Group and colleagues at the Vienna University of Technology for their endless discussions and refreshing ideas.

7 REFERENCES

[1] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. "Simplification Envelopes". In *Computer Graphics (SIGGRAPH '96 Proceedings, Annual Conference Series)*, pages 119-128, August 1996.

[2] M. Garland and P. S. Heckbert. "Surface Simplification Using Quadric Error Metrics", In *Computer Graphics (SIGGRAPH '97 Proceedings, Annual Conference Series)*, pages 209-214, August 1997.

[3] F. S. Hill. *Computer Graphics*. Macmillian Publishing Company, NJ, 1990.

[4] D.G. Holmes and D.D. Snyder. "The generation of unstructured triangular meshes using Delaunay triangulation". In *Proceedings of the Second Conference on Grid Generation in Computational Fluid Dynamics*, Pineridge Press, Swansea, 1988.

[5] H. Hoppe. "Efficient Implementation of Progressive Meshes". To appear in *Computers Graphics*, 1998.

[6] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Mesh Optimization". In *Computer Graphics (SIGGRAPH '93 Proceedings, Annual Conference Series)*, pages 19-26, August 1993.

[7] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Surface Reconstruction from Unorganized Points". In *Computer Graphics (SIGGRAPH '91 Proceedings, Annual Conference Series)*, pages 242-249, 1991.

[8] A. Klingert and W. Straub. "A Robust and Efficient Triangulation Algorithm for Contours on Parallel Planes". In *Report 11/92*. Institut für Betriebs- und Dialogsysteme. Universität Karlsruhe, 1992.

[9] J.-O. Lachaud and A. Montanvert. "Volumetric Segmentation using Hierarchical Representation and Triangulated Surface". In Research Report No 95-37, Ecole Normale Supérieure de Lyon, 1995.

[10] C. Liska. "Robuste und Adaptive 3D-Bildgewinnung mittels Laserlicht". To appear in *Technical Report 51*. Institute for Automation, Pattern Recognition and Image Processing Group. Technical University Vienna, 1998.

[11] W. E. Lorensen and H. E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In *Computer Graphics (SIGGRAPH '87 Proceedings, Annual Conference Series),* pages 163-167, July 1987.

[12] C. Menard and R. Sablatnig, "Pictorial and 3d Data Acquisition of Archaeological Finds - the Basis for Automatic Classification", In H. Kamermans and K. Fennema, (Eds): *Interfacing the Past, Computer Applications and Quantitative Methods in Archaeology - CAA95, Leiden*, No. 28, pages 413-424, Analecta Praehistorica Leidensia, 1996.

[13] J. Popovic and H. Hoppe. "Progressive Simplicial Complexes". In *Computer Graphics (SIGGRAPH '97 Proceedings, Annual Conference Series)*, pages 217-224, August 1997.

[14] F. P. Preperata and M. I. Shamos. *Computational Geometry*. Springer Verlag. New York, 1985

[15] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. "Decimation of Triangle Meshes". In *Computer Graphics (SIGGRAPH '92 Proceedings, Annual Conference Series)*, pages 65-70, July 1992

[16] Y. Shirai. "Three-Dimensional Description and Representation". In *Three-Dimensional Computer Vision*. Springer Verlag, pages 189-199, 1987.

[17] G. Turk. "Re-Tiling Polygonal Surfaces". In *Computer Graphics (SIGGRAPH '92 Proceedings, Annual Conference Series),* pages 55-64, July 1992.

[18] F. Wilmer, U. Tiede and K. H. Höhne. "Reduktion der Oberflächenbeschreibung durch Anpassung an die Objektform". In S. Fuchs and R. Hoffmann (Eds.): *Mustererkennung 1992, Informatik aktuell*, pages 430-436. Springer Verlag, 1992.

[19] T. Sperlich. “In 3D um die Welt”. In *c’t – magazin für computer technik*, pages 48-49. Heise Verlag, 1998.

Appendix A (Implementation Details)

A.1 Implementation

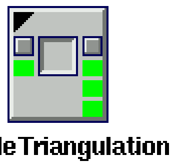
All tasks (Angle Criteria Triangulator, Marching Cubes Triangulator, Normal Optimizer, and 3DConverter) have been implemented and tested under the Khoros/Cantata Visual Programming Environment ver. 2.2 running on SGI Irix 6.4 and Linux 2.0.33 using C++ and the standard template library (STL). As a base for all three tasks, a Khoros C++ library (Geom3DLib) has been created providing a class hierarchy for generic geometric objects (meshes, cross sections, contours, vertices, faced meshes), input and output constraints (file readers and writers) and administrative purposes (performance measurements). Based on this library, each task defines its own task class, which realizes its very specific algorithm. For any future work dealing with 3D geometry this library should be a solid base to start with and thanks of the object-oriented paradigm any functionality can be easily added.

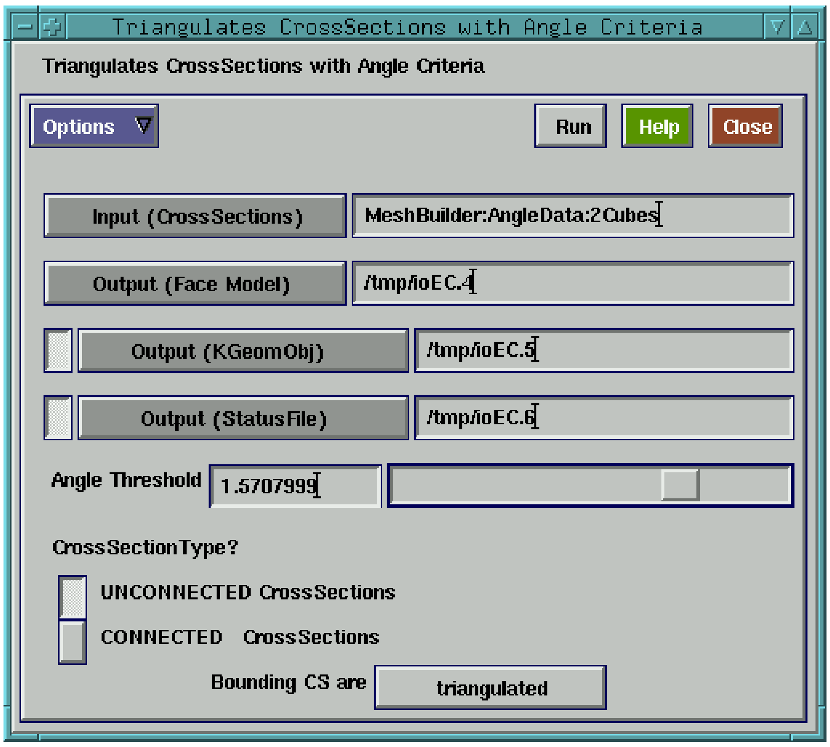
A.2 Usage of Tasks

All tasks are located in the "MeshBuilder" menu item in the Khoros "Glyphs" menu. There you can find three subcategories ("Triangulation", "Optimization" and "Converters") for tasks realizing triangulation, mesh optimization and model file format conversion algorithms. For any sample model files and sample Khoros workspaces be sure to see the "Meshbuilder" alias.

A.2.1 Angle Criterion Triangulator

This task implements the cross section triangulation algorithm described in Section 2.2.





Triangulates CrossSections with Angle Criteria

Input (CrossSections) MeshBuilder:AngleData:2Cubes

Output (Face Model) /tmp/ioEC.4

Output (KGeomObj) /tmp/ioEC.5

Output (StatusFile) /tmp/ioEC.6

Angle Threshold 1.5707999

CrossSectionType?

UNCONNECTED CrossSections

CONNECTED CrossSections

Bounding CS are triangulated

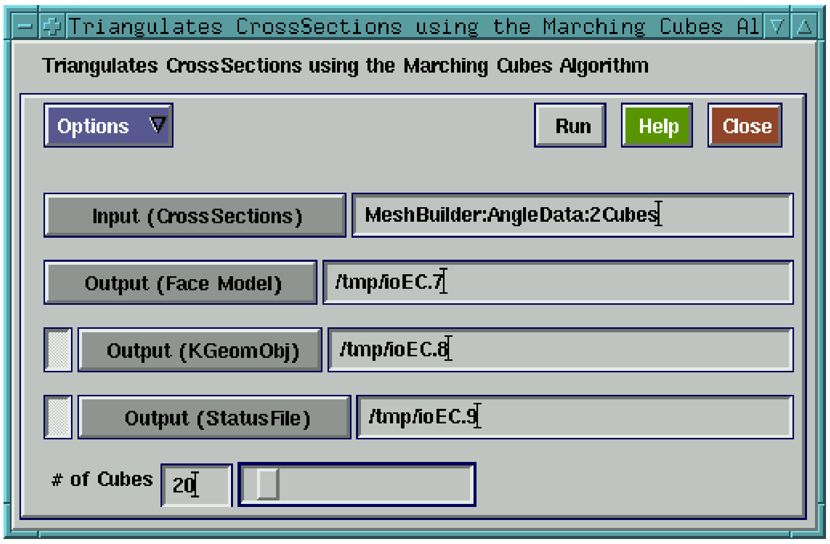
Figure 16 - Angle Criteria Triangulator User Interface

Description of task parameter:

Name	Description
Input (Cross Section)	Input data to the triangulation process (profile file format (see A.3))
Output (Face Model)	Output data of the triangulation process (mesh file format (see A.4))
Output (KgeomObj)	Output data of the triangulation process (Khoros geometry file format) for direct use as input for the Khoros Renderer
Output (StatusFile)	Output of triangulation statistics (time consumption, number of faces, vertices)
Angle Threshold	Angle Threshold parameter described in 2.2.2
CrossSection Type	Determines whether contours are assumed to be open ("unconnected CS") or closed ("connected CS") polygons
Bounding CS are	"triangulated" – bottom and top level cross section will be triangulated "non-triangulated" – bottom and top level cross section will remain untriangulated

A.2.2 Marching Cubes Triangulator

This task implements the Marching Cubes Algorithm described in Section 2.3.



Triangulates CrossSections using the Marching Cubes Algorithm

Input (CrossSections) MeshBuilder:AngleData:2Cubes

Output (Face Model) /tmp/ioEC.7

Output (KGeomObj) /tmp/ioEC.8

Output (StatusFile) /tmp/ioEC.9

# of Cubes 20

Figure 17 - Marching Cubes Triangulator User Interface

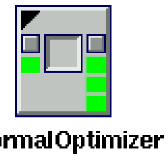
Description of task parameter:

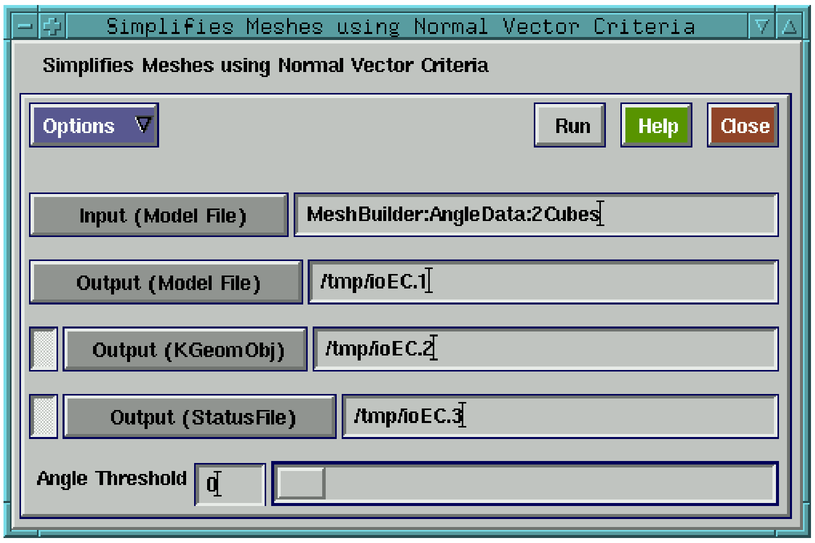
Name	Description
Input (Cross Section)	Input data to the triangulation process (profile file format (see A.3))
Output (Face Model)	Output data of the triangulation process (mesh file format (see A.4))
Output (KgeomObj)	Output data of the triangulation process (Khoros geometry file format) for direct use as input for the Khoros Renderer
Output (StatusFile)	Output of triangulation statistics (time consumption, number of faces, vertices)
# of Cubes	Resolution of the March Cubes Raster

The "# of Cubes" parameter determines the number of raster elements in x and y direction according to the object's bounding box. The raster element width in z direction is implicitly given by the distance between two adjacent cross sections.

A.2.3 Normal Optimizer

This task implements the Mesh Optimization Algorithm, using a normal vector criterion, described in Section 3.





Simplifies Meshes using Normal Vector Criteria

Input (Model File) MeshBuilder:AngleData:2Cubes

Output (Model File) /tmp/ioEC.1

Output (KGeomObj) /tmp/ioEC.2

Output (StatusFile) /tmp/ioEC.3

Angle Threshold 0

Figure 18 - Normal Optimizer User Interface

Description of task parameter:

Name	Description
Input (Cross Section)	Input data to the triangulation process (profile file format (see A.3))
Output (Face Model)	Output data of the triangulation process (mesh file format (see A.4))
Output (KgeomObj)	Output data of the triangulation process (Khoros geometry file format) for direct use as input for the Khoros Renderer
Output (StatusFile)	Output of triangulation statistics (time consumption, number of faces, vertices)
Angle Threshold	Normal Vector Angle Threshold described in

A.2.4 3D Geometry File Converter

This task is to prove conversion of 3D geometry file formats to facilitates work with different rendering and visualization systems.

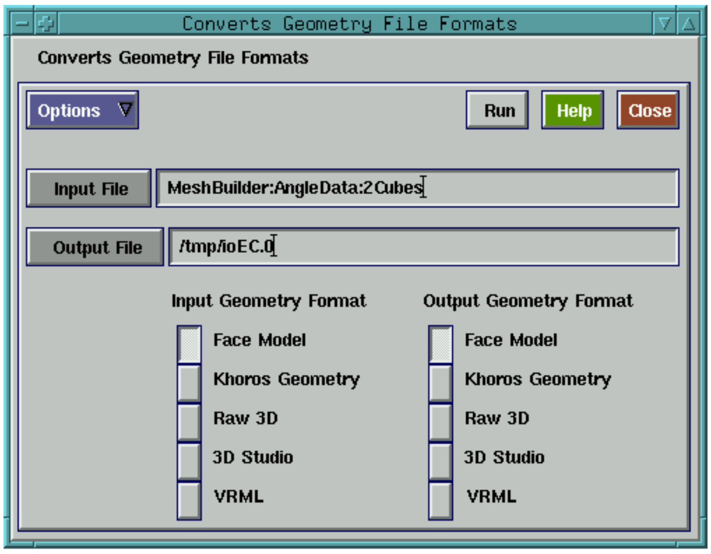
Description of task parameter:

Name	Description
Input File	Input data to the triangulation process (profile file format (see A.3))
Output File	Output data of the triangulation process (mesh file format (see A.4))
Input Geometry Format	Specifies the 3D Input File Format
Output Geometry Format	Specifies the 3D Output File Format

So far, five different 3D geometry file formats are supported:

- Face Model Mesh File, described in Section A.4
- Khoros internal 3D geometry file format
- Simple Raw 3D format, i.e. list of nine coordinates of a triangle per line
- 3D Studio mesh file format
- VRML 2.0 format

Geometry Converter


Converts Geometry File Formats

Converts Geometry File Formats

Input File: MeshBuilder:AngleData:2Cubes

Output File: /tmp/ioEC.0

Input Geometry Format	Output Geometry Format
Face Model	Face Model
Khoros Geometry	Khoros Geometry
Raw 3D	Raw 3D
3D Studio	3D Studio
VRML	VRML

Figure 19 – 3D File Format Converter User Interface

A.3 Description of Profile File Format

A profile file contains all 3D data which is needed by the triangulation algorithm to reconstruct a triangulated 3D mesh from the object which was scanned in. The 3D data is separated by sections (i.e. the cross section keyword "section") where a section represents a cross section of the scanned object. A section must contain at least one contour (otherwise there are no vertices in the contour the beginning of the first contour is assumed by the cross section keyword and any additional contours are separated by the contour delimiter "contour"). A contour (polygon) is defined by the vertices it contains and they have to appear either in clock wise or counter clock wise ordering. Comments are indicated by the "#" symbol which assigns the rest of the line to be a comment. Comments may occur at any time after the file header string.

Language Definition

The profile file format grammer (PFFG) is defined as follows:

```
PFFG :=
        (
          { PROFILEFILE, FILEHEADER, FILEBODY, COMMENT, SECTION, CONTOUR,
            VERTICES, VERTEX, TEXT, COORD, LETTER, DIGIT },
          { "pro", "#", "section", "contour", "." },
          ℘,
          PROFILEFILE
        )

℘  :=
        {
          PROFILEFILE ⇒ FILEHEADER FILEBODY
          FILEHEADER  ⇒ "pro"
          FILEBODY    ⇒ [COMMENT] SECTION {SECTION}
          COMMENT     ⇒ "#" TEXT [COMMENT]
          SECTION     ⇒ "section" [COMMENT] VERTICES [CONTOUR]
          CONTOUR     ⇒ "contour" [COMMENT] VERTICES
          VERTICES    ⇒ VERTEX [COMMENT] {VERTICES}
          VERTEX      ⇒ COORD COORD COORD
          TEXT        ⇒ {LETTER | DIGIT}
          COORD       ⇒ DIGIT {DIGIT} ["." {DIGIT}]
          LETTER      ⇒ a-z | A-Z
          DIGIT       ⇒ 0-9
}
```

A.4 Description of the Mesh File Format

A mesh file is generated by one of the triangulation tasks and serves as input of the mesh optimization task. This special mesh file format is needed since, for the mesh Optimizer task to be more efficient, it quickly has to restore neighborhood relations between faces which cannot be satisfied be the Khoros Geometry Object. Furthermore, this common simple mesh file format can be used to import the model geometry into any other 3D-modeling program.

First, the file header "mesh" determines the type of file, followed by the list of vertices of the mesh, which are determined by three coordinate values and finally the list of faces is denoted. A face is simply defined by three indices of vertices. Again, comments may occur at any time after the file header string.

Language Definition

The mesh file format grammar (MFFG) is defined as follows:

```
MFFG :=
       (
         { MESHFILE, FILEHEADER, FILEBODY, FACE, FACES, COMMENT, VERTICES, VERTEX, TEXT,
           INDEX, COORD, LETTER, DIGIT },
         { "mesh", "#", "v", "f", "." },
         ℘,
         MESHFILE
       )

℘ :=
       {
         MESHFILE   ⇒ FILEHEADER FILEBODY
         FILEHEADER ⇒ "mesh"
         FILEBODY   ⇒ [COMMENT] VERTICES FACES
         COMMENT    ⇒ "#" TEXT [COMMENT]
         VERTICES   ⇒ VERTEX [COMMENT] {VERTICES}
         VERTEX     ⇒ "v" COORD COORD COORD
         FACES      ⇒ FACE [COMMENT] {FACES}
         FACE       ⇒ "f" INDEX INDEX INDEX
         TEXT       ⇒ {LETTER | DIGIT}
         INDEX      ⇒ DIGIT {DIGIT}
         COORD      ⇒ DIGIT {DIGIT} ["." {DIGIT}]
         LETTER     ⇒ a-z | A-Z
         DIGIT      ⇒ 0-9
       }
```

A.5 Installation

On any Unix system, providing a ANSI C/C++ compatible C++ compiler (CC or gcc/g++) and an ANSI implementation of the STL, running a Khoros 2.2 environment, simply unzip and untar the distribution file and add a toolbox reference to the top level mount point of the untared archive. Afterwards, recreate Imakefiles and Makfiles and do a "make install".