TRIANGULATION of 3-D objects

J.D. Boissonnat O.D. Faugeras

INRIA Domaine de Voluceau-Rocquencourt
BP 105 78153 I e Chesnay Cedex FRANCE

Introduction

In this paper we describe an efficient way of building a polyhedral approximation of a set of point on a surface in 3-D space obtained by a laser range finder. The algorithm is a generalization of an existing algorithm for polygonal approximation of a curve in 2D space CI]. First we recast algorithm [1] in the setting of graph theory which allows us to generalize it to 3-D. Second we briefly explain how the surface graph which is the basis of the present algorithm can be obtained experimentally. Third we describe the main components of our algorithm.

Description of the two-dimensional algorithm

We will describe the planar algorithm for discrete simple closed curves in the sense of ['21. Let [2].A_i i=0,...,n be a set of points forming a curve C on a discrete grid. For reasons which will become evident later we will not assume that these points are ordered along the curve, that is that A_{i-1} and A-j-i (all indexes are taken modulo n) are the two neighbors of A_j. Nonetheless we assume that the structure of the set of points A_j is given to us as a graph G=(V,A) where the set of vertexes V is the set of points A_j and A the set of edges. A is a subset of V*V indicating neighboring vertexes that is to say neighbors on the curve.

The planar algorithm then goes as follows :

Algorithm 1 :

i) pick any two points P and Q on C which are not neighbors. The line PQ is the zero order approximation of the curve C.

ii) find the shortest path from P to Q in G. This separates the *curve* into two curves C_1 and C2 and the graph G into two disconnected subgraphs G_1 and Go.

iii) process G_1 and G_2 independently : find the most distant point P_i on C_j to the line PQ (i=I,2). The polygon PP_1QP_2 is the first order approximation to the curve C.

iv) we then recursively split the curves C_j in two by finding the shortest path in G_j from A to P_i and iterating step iii) on the corresponding subgraphs untils we reach an acceptable error level.

In practice of course the need to find shortest paths between vertices in a graph can be avoided in the planar case by the fact that points on a curve can be ordered as a sequence of neighbors. This unfortunately is not true in higher dimensions and this is why we have cast the well known algorithm 1 into a more general setting which in return will allow us to generalize it in a straightforward manner to the case of surfaces.

Description of the three-dimensional algorithm

These discrete surfaces have been obtained by scanning real objects whithout holes (in our case industrial parts) with a laser range finder built at INRIA. The object being positioned in front of the acquisition system, we "raster scan" the corresponding view by moving a laser beam from top to bottom and left to right of the object. The position in space of the laser spot is computed by triangulation. The laser range finder system is described elsewere [3].
Right now we have an accuracy of .75mm at distance of 5m for a depth of field of 75cm and make a measurement *every* 10ms. We then rotate the object in a controlled manner to "see" all of it. This provides us with a set of points A_j (i=0,...,n) in 3-D space every point having a number of neighbours. Some processing has to take place to obtain this information for all points in *every* view. This is described elsewhere [4].

We are now able to build from the set of points A_i and their neighbors the corresponding graph G=(V\A) as in the planar case.
From there we can design an algorithm which is almost the same Algorithm 1 and which "almost" works :

A1gorithm *2 :*

i) pick three points P,Q and R on the surface S which *are* not neighbors.
The plane PQR is the zero order approximation of the surface S.

ii) find the shortest most planar cycle through PQR. This forms a cycle in the graph G which we denote by (PQR). This cycle separates the surface into two surfaces S_1 and S_2 and the graph G into two disconnected subgraphs S_1 and Go (we arbitrarily associate the points of PQR) either to G_1 or to G_2).

iii) process G_1 and G_2 independently : find the most distant point P_i of S_i to the plane PQR (i-1,2).
The polyhedron (PP_1Q,QP_1R,RPiP,Pp2Q,QP2R,RPoP) is the first order approximation of the surface S.

iv) we then recursively split the surfaces

S_i i=1,2 in three as follows : find the cycle (PP_iQ) which does not contain R in G_i. This cycle is chosen in such a way that the path PP_i, for example, is the shortest in terms of a set of weights equal to the distances of each point to the bissector of planes PP_iR and PP_iQ. This splits G_i in G_{i_1} and G_{i_2}. Suppose G_{i_2} is the component of G_i that contains R. Then find the cycle (QP_iR) in G_{i_2}. This achieves the splitting in three : G_{i_1},G_{i_2} and G_{i_3} correspond to triangles PP_iQ, QP_iR and RP_iP, respectively (i=1,2).

Each subgraph (subsurface) is then processed independently in the same way as in iii) until we reach an acceptable error level. The net result is an approximation of the original surface S by a polyhedron whose faces are triangles. Oth, first and second order approximation of a sphere are shown in figure 1. Notice an undesirable fact about Algorithm 2, namely that once an edge of the polyhedron has been created, it can never disappear even though it may be quite a bad approximation to the surface. We therefore added the following refinement. Remember that to every triangular face PQR of the approximating polyhedron we can associate a subgraph G' of the original graph G or to put it another way a piece S' of the original surface S. To every face PQR we can also associate three neighboring faces PQP_1,QRP_2 and RPP_3 and their corresponding graphs G'_1,G'_2 and G'_3 as shown in figure 2. To explain the process by which we break edges in the polyhedron let us suppose that we find that the plane defined by the triangle PQR is not a good enough approximation of S'. Let M be the most distant on point on the surface S' to this plane. Let us also suppose that PQP_1 and QRP_2 are in the same situation and that we associate points M_1 and M_2 to them. Finally we will assume that triangle RPP_3 is a good approximation to S'_3. Algorithm 2 would proceed as indicated in Figure 3, Algorithm 3 proceeds as indicated in Figure 4. Notice that edges PQ and QR have disappeared in the new polyhedral representation. One crucial point is of course how we build the piece S''_1 of S (the subgraph G''_1 of G) that we associate to triangle PMM_1, for example. Very simply by considering $G'' = G' \cup G'_1$ which is a connected subgraph of G containing the points $PQRP_1MM_1$ we can find the cycle (PMM_1) in G'' which disconnects it in G''_1 and G''_2. G''_1 is the component which does not contain Q.
Oth, first and second order approximation of a sphere are shown in Figure 5.

Conclusion

We have developed an algorithm for approximation 3-D polyhedral surfaces with triangular faces. These surfaces are first digitized with a laser range finder yielding a set of points on the surface in 3-D space. A graph representation is then built which includes neighboring information for every point. A recursive split of the set of points is then obtained by using the graph representation of the surface to find paths between points on the surface. Since we know that graphs have compact machine representations and that there are efficient algorithms for finding a shortest path between two points in a graph we think that our technique is promising. We hope to have results on industrial parts at the time of the conference.

References

[1] Duda, R., and Hart, P., Pattern Classification and Scene Analysis, Wiley-Interscience, 1973.

[2] Rosenfeld, A., Picture Languages, Academic Press, 1979.

[3] Boissonnat, J.D., and Germain, F., "A new approach to the problem of acquiring randomly oriented workpieces out of a bin", these Proceedings.

[4] Boissonnat, J.D., and Faugeras, O.D., "How to build a graph representation of objects from the INRIA laser range finder", INRIA internal report.

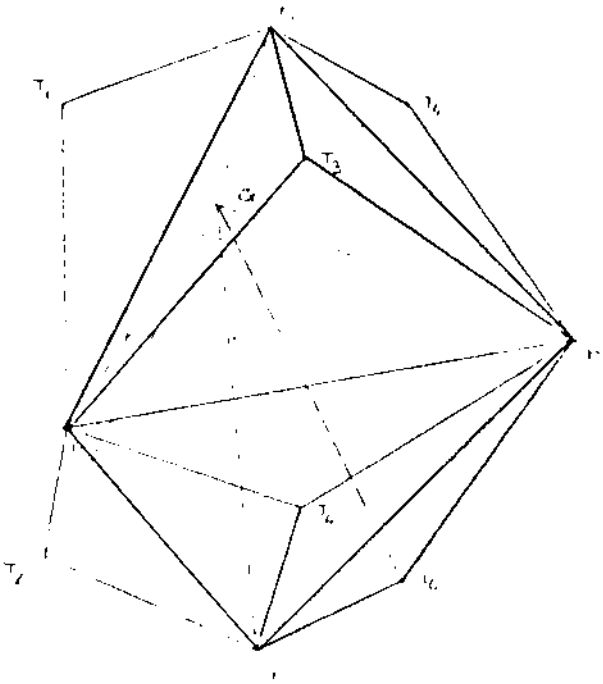

Figure 1 : Second order approximation of the sphere produced by Algorithm 2. All lines are edges.

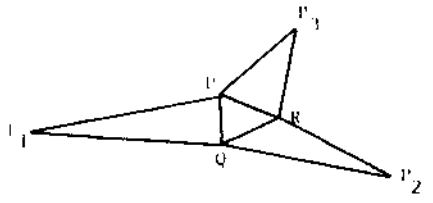

Figure 2.

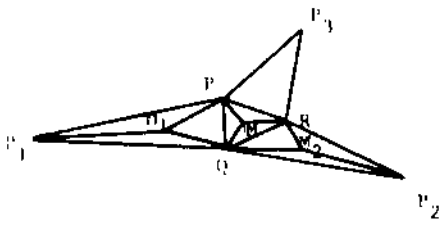

Figure 3 : The new triangulation produced by Algorithm 2.

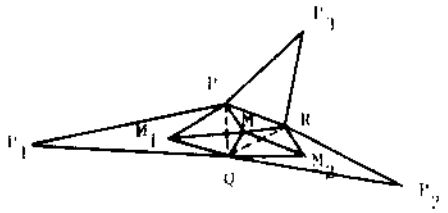

Figure 4 : The new triangulation produced by the modified version of Algorithm 2. The dotted lines correspond to edges that disappear.

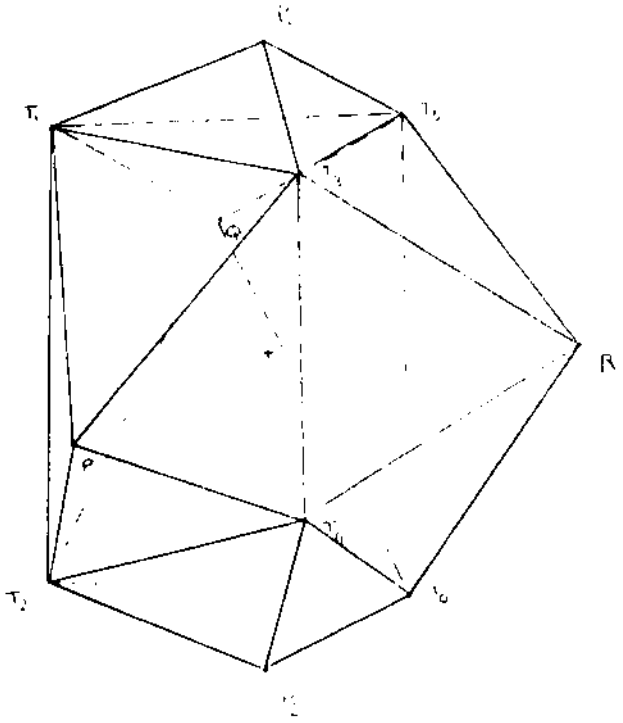

Figure 5 : Second order approximation of the sphere produced by the modified version of Algorithm 2. Notice that edges PQ,QR and RP have disappeared.