The Visual Computer

Towards shape representation using trihedral mesh projections

Lluís Ros[1], Kokichi Sugihara[2], Federico Thomas[1]

[1] Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens Artigas 4-6, 08028 Barcelona, Spain
E-mail: llros@iri.upc.es, fthomas@iri.upc.es
[2] Department of Mathematical Informatics, University of Tokyo. 7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
E-mail: sugihara@simplex.t.u-tokyo.ac.jp

Published online: 5 February 2003

This paper explores the possibility of approximating a surface by a trihedral polygonal mesh plus some triangles at strategic places. The presented approximation has attractive properties. It turns out that the Z-coordinates of the vertices are completely governed by the Z-coordinates assigned to four selected ones. This allows describing the spatial polygonal mesh with just its 2D projection plus the heights of four vertices. As a consequence, these projections essentially capture the "spatial meaning" of the given surface, in the sense that, whatever spatial interpretations are drawn from them, they all exhibit essentially the same shape.

Key words: Shape representation – Polygonization – Trihedral mesh – Reconstruction from projections – AND/OR graphs

Correspondence to: L. Ros

1 Introduction

A *polygonal mesh* is a piecewise linear 2-manifold made up with planar polygonal patches, glued along the edges, and possibly containing holes. A *polygonization method* is an algorithm able to construct a polygonal mesh approximating a given surface. The literature on polygonization methods, mainly on triangulations, is vast (see for example [3, 18], or [6] for a survey on triangulations and algorithms to simplify them). In general, the main goal is to obtain meshes that are close to the surface within a known error, as a way to understand and represent the surface shape [12]. Other goals have been to increase the speed of polygonization and the ability of the polygonizer to satisfy some constraints in the solution (e.g., one might request the most accurate approximation using a given number of line segments or triangles).

In general, a polygonal mesh cannot be reconstructed from its projection onto a plane because infinitely many meshes generate exactly the same projection. For example, for the triangular mesh projection in Fig. 1, there are many different reconstructions, as illustrated. The first two seem to have no meaning; but, actually, there is a rather "hidden" meaningful reconstruction of Nefertiti's face available. Can we obtain a spatial mesh approximating this face so that its projection still keeps its spatial meaning?

Certainly, there is a class of meshes whose projections fully determine the spatial shape once the heights of four vertices are given. We call these projections *unequivocal* because their reconstructions represent essentially the same object. For example, the projection in Fig. 2a unequivocally represents a truncated tetrahedron, as seen in Fig. 2d–f. Observe that it suffices to set the heights of P, Q, T and R to determine those of S and U, using the fact that all cofacial vertices must be coplanar and, hence, S must lie on the face-plane $RPQS$, and U on $SQTU$. In general, if all vertices of a polyhedron have exactly three incident faces, the heights of a vertex and its three neighbours are sufficient to determine the heights of the rest, as Sect. 2 explains. One of our goals is then to approximate any given surface with a polygonal mesh yielding unequivocal projections that uniquely identify the spatial shape once the heights of four vertices (selected in this way) are given. Section 2 presents the trihedral polygonal mesh, the model we use to this end, and shows how its projections are unequivocal in the sense given above.

The Visual Computer (2003) 19:139–150
Digital Object Identifier (DOI) 10.1007/s00371-002-0176-9

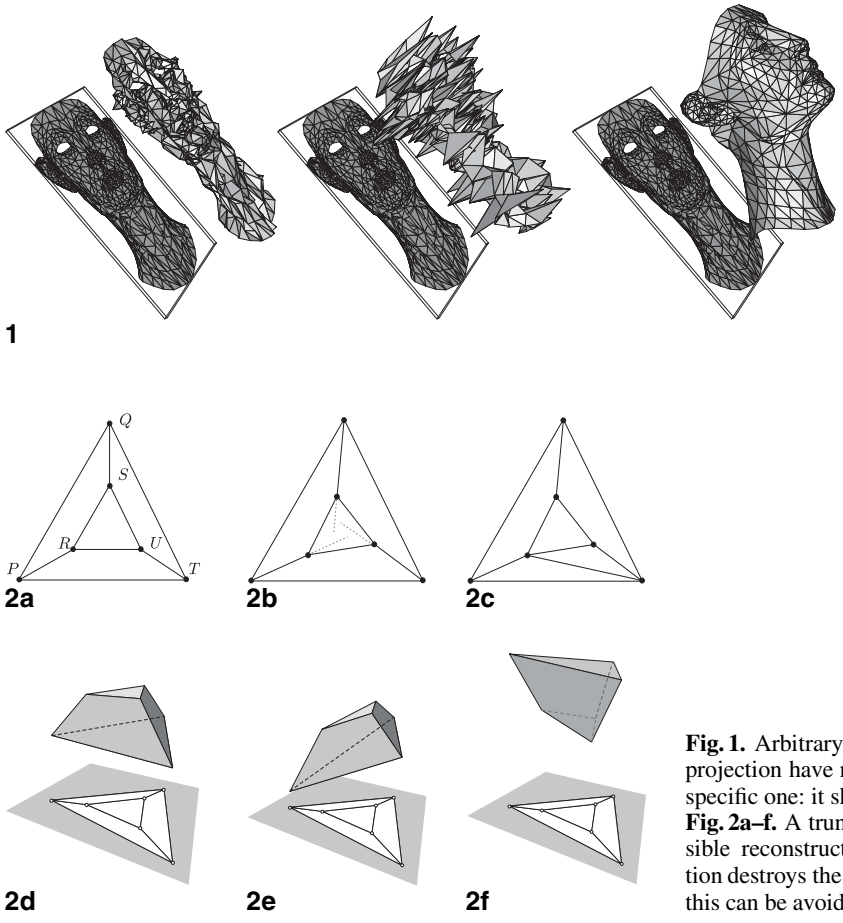

Fig. 1. Arbitrary reconstructions of this triangulated projection have no spatial meaning except for a very specific one: it shows Nefertiti's face
Fig. 2a–f. A truncated tetrahedron (**a**) and three possible reconstructions (**d–f**). The slightest perturbation destroys the correctness of the projection (**b**), but this can be avoided adding new triangular faces (**c**)

Nevertheless, we need to go beyond this goal if this representation is to be useful. Consider what happens if the (x, y) vertex positions in Fig. 2a are slightly altered (Fig. 2b). The new projection no longer represents a correct truncated tetrahedron, for to be so, the edges joining the two triangular faces, when extended, should be concurrent at the apex of the (imaginary) original tetrahedron. Equivalently, note that once P, Q, R and T are given, the height of U is overconstrained, for it can be calculated from *both* the coplanarity of $SQTU$ or that of $RPTU$. For generic vertex positions, the two values of this height do not necessarily coincide, and the only spatial reconstruction that keeps cofacial vertices coplanar is a *trivial* one, with *all* vertices lying on a single plane [10, 11]. This makes the four provided heights inconsistent between each other. In sum, the consistency of the four heights only holds at very specific positions of the vertices and inevitable discretization errors will make this representation useless. This problem is common in computer vision [13] and computer graphics [15, 17], and mathematical characterizations of generically consistent projections are given in [14, 16]. The way we use to make this representation robust against these errors follows from this observation: if the height of a vertex in a projection is overconstrained because the vertex lies on several planes that fix it, we just introduce new triangular faces around it for preventing this to occur (Fig. 2c). Section 3 gives a fast algorithm to this end, derived from this observation, using the so-called T/TT-transformations. Section 4 describes a complementary optimization step that properly places these transformations to minimize the reconstruction errors by reducing the problem to

a cyclic AND/OR graph search. We finally conclude in Sect. 5.

2 Trihedral polygonal meshes

Trihedral meshes, i.e., those where all vertices have exactly three incident faces, produce unequivocal projections. Indeed, Fig. 3 shows that in them, after fixing the planes of two adjacent faces, we have enough data to derive the heights of the remaining vertices. Clearly, the heights of the bold vertices fix the shadowed face-planes and the heights of other vertices on these face-planes. At this point, any other surrounding face has three vertices whose heights are known, and so, its plane can be fixed too. The same argument can be iteratively applied and the result is a *height propagation* reaching all vertices in the projection.

In the schematic representation of this height propagation (Fig. 3) every face f receives three incoming arrows from the three vertices that fix it. The derivation of heights for the rest of vertices on f is indicated with outgoing arrows from f. The result is a tree-shaped structure spanning all vertices and faces. In this tree, a path from any of the initial four vertices to any other vertex will be hereafter referred to as a *propagation wave*. Note that height propagations where a face is fixed from three (almost) collinear vertices must be avoided. Section 4 gives a way to compute propagations eluding these collinearities.

A trihedral mesh approximating a convex or concave surface can be readily obtained by distributing a set of random points all over the surface and computing its tangent planes at these points. This leads to a plane arrangement whose upper envelope, if the surface is convex, or lower envelope, if it is concave, provides a good mesh approximation of the surface. Since the tangent plane orientations are random, any three of such planes meet in a single point, and hence the mesh is trihedral.

Alternatively, a trihedral mesh approximation of a piece of concave or convex surface can be obtained by starting with a rough mesh approximation and iteratively applying a bevel-cutting [5] and/or a corner-cutting [2] operation to attain the desired approximation.

The situation becomes more complex when concavities, convexities and saddle-like shapes are simultaneously present, as neither of the previous strategies

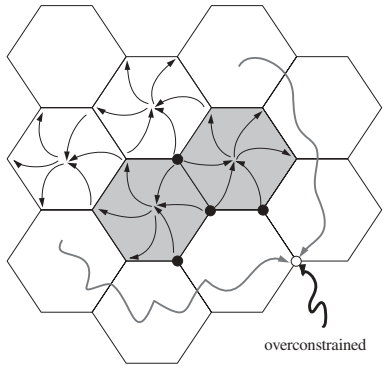

Fig. 3. A height propagation starting at four prespecified (*bold*) vertices. Several vertices can have an overconstrained height

can be directly applied. A straightforward solution to find a trihedral mesh approximation of a general surface consists of computing the three-dimensional Voronoi diagram of a set of points sampling the surface. The algorithm encompasses the following steps, where $\mathcal{S}$ denotes the input surface to be polygonized:

1. Generate points on $\mathcal{S}$ at random with a density proportional to the curvature of $\mathcal{S}$.
2. Replace each point p produced in step 1 by a pair of *generating points*, p' and p'', placing them on the normal to $\mathcal{S}$ at p, one in each side of $\mathcal{S}$, at the same distance from p. This distance must be sufficiently small so that the line-segment connecting p' and p'' penetrates the surface exactly once.
3. Construct the three-dimensional Voronoi diagram induced by all generating points. Each such point will yield one polyhedral Voronoi cell in the diagram, possibly unbounded.
4. For each polygonal face separating two cells, if the two cells correspond to points on different sides of $\mathcal{S}$, select this face as part of the desired output polygonization. Collecting such faces, we get a polygonization with only trihedral or tetrahedral vertices, with three and four incident faces, respectively.
5. Use one of the local operations shown in the bottom row of Fig. 4 to replace every tetrahedral vertex with trihedral ones. Output the resulting polygonization.

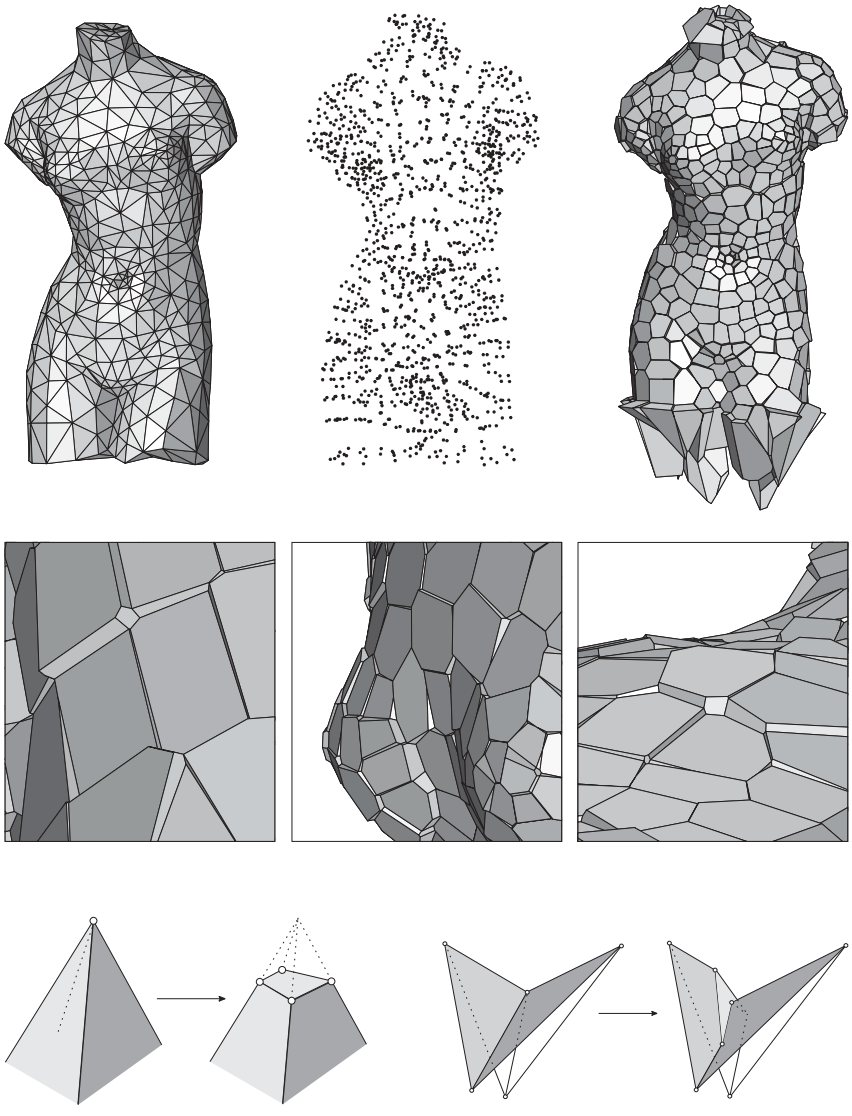

Fig. 4. *Top row*: a triangulated model of a Venus (*left*) is symmetrically sampled (*center*) to obtain a polygonization with only trihedral and tetrahedral vertices (*right*). *Center row*: zoom into three selected areas. *Bottom row*: local operations used to remove tetrahedral vertices

We easily see that this algorithm produces a trihedral mesh. Since we select faces from the Voronoi diagram of points in general position, the polygonization obtained in step four either contains trihedral or tetrahedral vertices. (Any five points in general position will not be co-spherical, and hence, the vertices of the Voronoi diagram will have at most four incident Voronoi cells.) Any tetrahedral vertex of this polygonization can be finally removed by using one of the two operations shown in the bottom row of Fig. 4. If there is a plane that separates the vertex from its neighbours, we simply truncate the vertex (bottom left of Fig. 4), otherwise it is a saddle point and we replace it with four trihedral vertices (bottom right of Fig. 4).

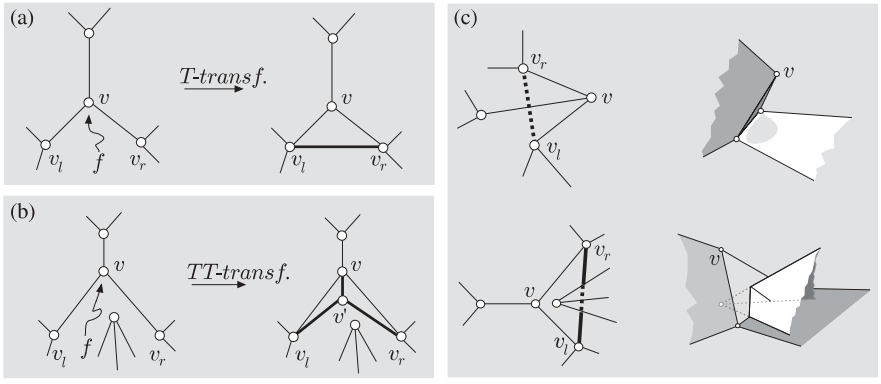

Fig. 5. a,b T and TT-transformations. **c** Overhanged and self-intersecting reconstructions induced by T-transformations at locally non-convex faces. In all figures, newly added edges are shown in thick lines

This strategy has been partially implemented until step four, using the Qhull package to compute the required 3D Voronoi diagram [9]. Strictly, Qhull computes convex hulls, but it can also be used to derive Delaunay triangulations, halfspace intersections about a point, Voronoi diagrams, furthest-site Delaunay triangulations, and furthest-site Voronoi diagrams in 2D, 3D, and higher dimensions. It implements the Quickhull algorithm for computing convex hulls [1] and is able to handle rounding errors inherent to the use of floating point arithmetic. The top row of Fig. 4 shows the results of the strategy when applied to a triangulated surface of a Venus sculpture. The input sculpture is shown (left) together with the two layers of generating points produced in step 2 (center), and the polygonization generated in step 4 (right). The middle row of Fig. 4 shows three enlargements of selected areas. One can come up with other "trihedrization" strategies with possibly better results, mainly concerning the smoothness of the approximation, but this does not modify the discussions that follow.

3 T and TT-transformations

In a trihedral mesh a projection is overconstrained because any of its vertices lies on three faces, and potentially up to three propagation waves can determine a height at the same time (Fig. 3). However, as done in Fig. 2c, this can be avoided by adding triangular faces. To this end, we first compute an arbitrary height propagation spanning all vertices and check which of them receives more than one wave. We then take one overconstrained vertex v at a time and prevent all *but one* waves from reaching v as follows. To stop the wave getting v from face f, we apply either of these two transformations (Fig. 5a,b):

- A *T-transformation*, which places a new edge joining the two neighbouring vertices of v in f, say v_l and v_r.
- A *TT-transformation*, which places a new vertex v' on f, near v, and the three new edges (v', v), (v', v_l) and (v', v_r).

After either transformation, f is split into two or three faces respectively, so that it cannot constrain the height of v anymore. Also, the added triangles are innocuous because all heights can still be determined from the four initial ones.

Which transformation is preferred depends on the geometry of face f around vertex v. If all points inside the triangle $v_l v_r v$ belong to f, we say that f is *locally convex* at v. So, for situations where f is locally convex at v, simplicity prevails and T-transformations are enough (Fig. 5a). When local non-convexities are present (Fig. 5b), T-transformations would yield occluded or partially occluded crossing edges whose spatial reconstructions have overhanged parts or self-intersecting faces (Fig. 5c). Here, TT-transformations are preferred for they can avoid this.

An observation complements the strategy. In an overconstrained vertex v, either two or three in-

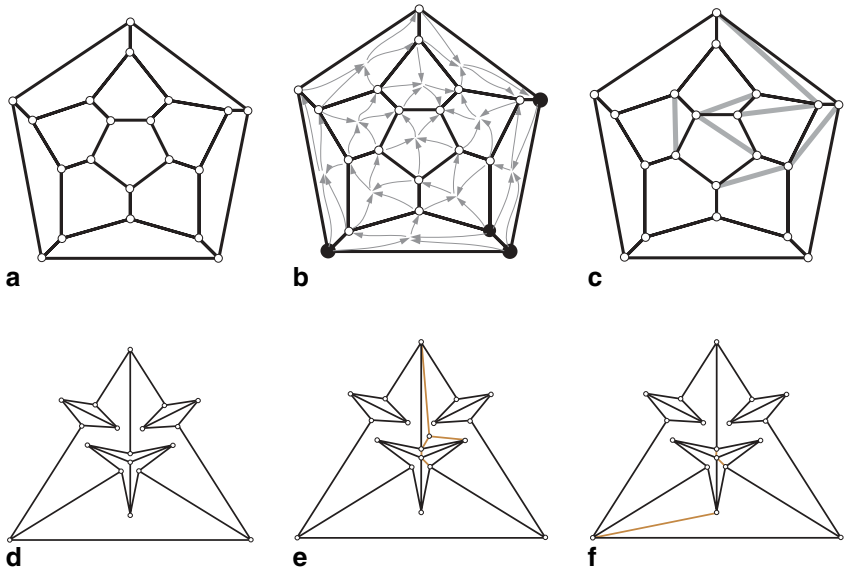

Fig. 6a–f. A projected dodecahedron (**a**) together with a height propagation (**b**) and the T-transformations it yields (**c**). A protruded tetrahedron (**d**) and two possible corrections: **e** involving TT-transformations, and **f**, involving only T-transformations

coming propagation waves arrive. If *no more* than one of them comes through a locally non-convex face, then we can always drop the incidence constraint in this vertex by only using T-transformations: we just leave the eventual "bad" wave to determine the height of v and stop the others with T-transformations. This completes the description of a one-sweep algorithm removing overdetermination. As an example, Fig. 6a–c shows a projected dodecahedron before and after applying T-transformations.

In general, when the approximated surface is uniformly convex, or uniformly concave, all faces of the resulting trihedral polygonal mesh will be locally convex, and hence T-transformations will suffice. However, even when local non-convexities exist at the faces, there still might be some height propagations where only T-transformations suffice. In Fig. 6e, for example, an algorithm computing an arbitrary propagation can be forced to use TT-transformations, whereas with a proper search, a robust projection is obtained only with T-transformations (Fig. 6f). But one certainly finds projections where TT-transformations are strictly necessary [10, Sect. 8.4].

4 Optimal propagations and cyclic AND/OR graphs

The algorithm in the preceeding section corrects the incidence structure by finding an arbitrary height propagation and inserting a T- or a TT-transformation whenever a vertex height is determined by two or more faces. However, arbitrary propagations might travel along "degenerate paths" where the planes for some of the faces are determined by three aligned (or almost aligned) vertices. Clearly, these *degenerate propagations* must be avoided if we want to minimize the errors during the reconstruction of the spatial shape from the initial set of four heights.

The following experiment exemplifies this point. For the projection of a trihedral strip in Fig. 7a, knowing the heights of points 1, 2, 3 and 4 produces a height propagation since, for $i = 1, 3, 5, 7, \ldots$, the 3D locations of points i, $i+1$ and $i+2$ determine the plane where the points $i+4$ and $i+5$ lie and, thus, their height on the spatial reconstruction. However, this plane becomes numerically ill-determined as the angle α between the points $i, i+1$ and $i+2$ approaches 180° and, when this happens, small errors

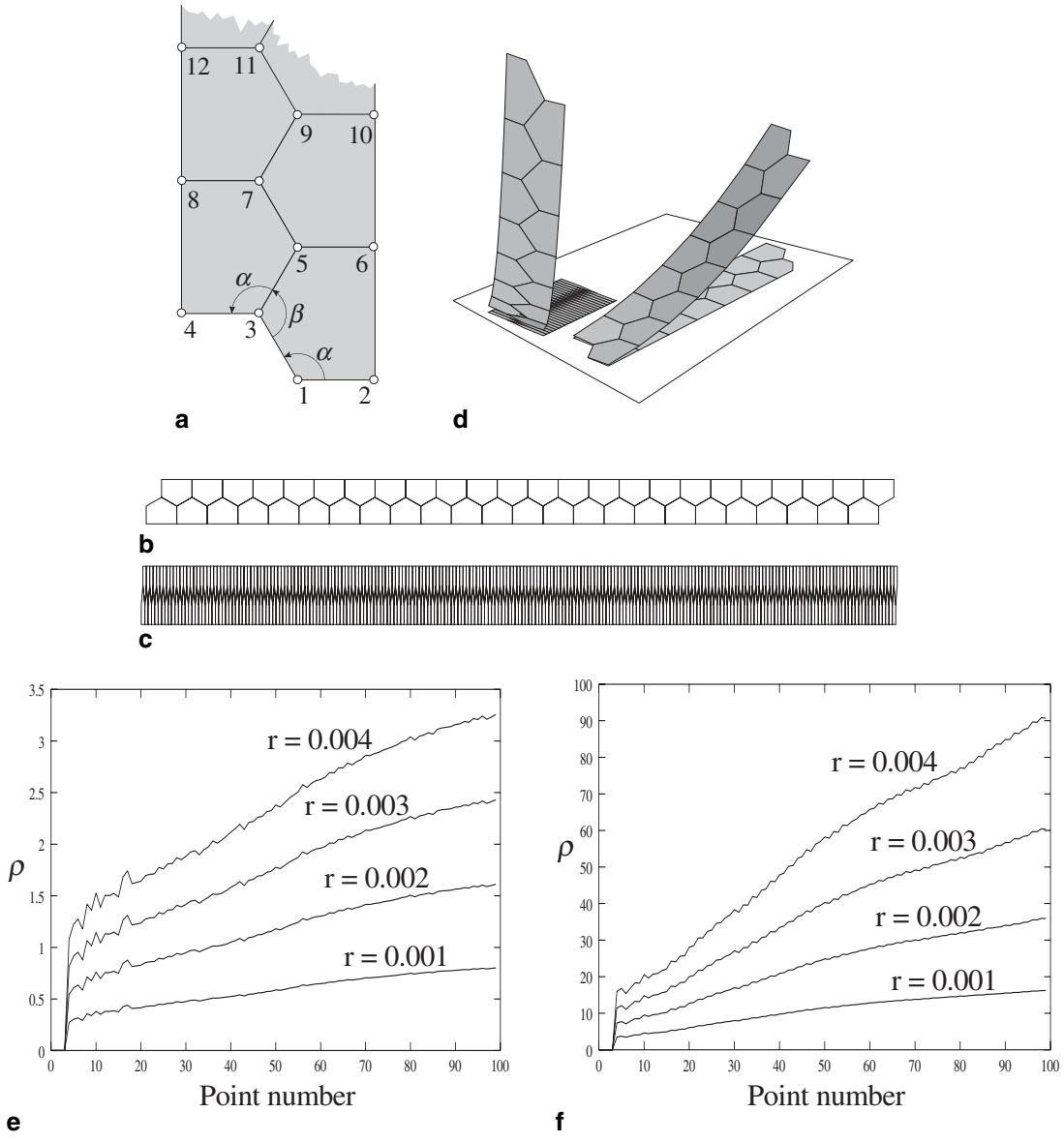

Fig. 7a–f. Error propagations. Given the projection of a trihedral strip (**a**), the Z coordinates of points 1, 2, 3 and 4 trigger a height propagation along the whole strip. Experiments have been done with the strips in **b** and **c**, also shown in **d** together with their spatial reconstructions. **e,f** The maximum relative error ρ of the computed heights on the two strips respectively when the XY coordinates of the points are perturbed within a circle of radius r. See the text for details

on previous heights may accumulate rapidly along the height propagation. We can see this effect on the two strips in Fig. 7b,c, whose angles are $\alpha = \beta = 120°$ and $\alpha = 175°$, $\beta = 10°$, respectively, with all interior edges of length 1. Their spatial reconstructions are shown in Fig. 7d, for $z_1 = 0.01$, $z_2 = z_3 = 0.05$ and $z_4 = 0.1$. For the two strips respectively, the plots in Fig. 7e,f show the maximum relative error ρ in the height of point i, when the XY coordinates of all points are perturbed within a circle of

radius r around their nominal position. The bounds on ρ have been obtained by randomly perturbing all points and then computing the height propagation, repeating this process 10 000 times. The results show that, for a fixed r, the errors accumulate linearly as the propagation proceeds. Moreover, by linearly varying r, the errors increase linearly as well. Note that, while for the first strip the errors keep moderately low, they rapidly increase for the second due to the high degeneracy of the height propagation. Thus, an algorithm to find height propagations that avoid these degeneracies is needed. The rest of the section presents one, based on finding the least-cost solution of cyclic AND/OR graphs [8]. We now recall some preliminary concepts about this kind of graphs.

4.1 Cyclic AND/OR graphs

An AND/OR *directed* graph G, can be regarded as a hierarchic representation of possible solution strategies for a major problem, represented as a *root node*, r, in G. Any other node v represents a subproblem of lower complexity whose solution contributes to solve the problem at hand.

There are three types of nodes: AND nodes, OR nodes and TERMINAL nodes. Every node v has a set $S(v)$ of *successor nodes*, possibly empty, to which it is connected in either of two ways:

- An AND node v is linked to all nodes $s_i \in S(v)$ through directed AND arcs (v, s_i), meaning that the subproblem for v can be trivially solved once *all* subproblems for the nodes in $S(v)$ have been solved.
- An OR node v is linked to all nodes $s_i \in S(v)$ through directed OR arcs (v, s_i), meaning that the subproblem for v can be trivially solved once *any one* of the subproblems for the nodes in $S(v)$ has been solved.
- A TERMINAL node represents a yet-solved or trivial subproblem and has no successors.

With this setting, a *feasible solution* to the problem becomes represented as a directed subgraph T of G verifying:

- r belongs to T.
- If v is an OR node and belongs to T, then exactly one of its successors in $S(v)$ belongs to T.
- If v is an AND node and belongs to T, then every successor in $S(v)$ belongs to T.
- Every leaf node in T is a TERMINAL node.
- T contains no cycle, it is a tree.

One can also assign a cost $c(u, v) > 0$ to every arc (u, v) in G and ask for the solution T with minimum overall cost $C(T) = \sum_{(u,v)\in E(T)} c(u, v)$, where $E(T)$ is the set of arcs of T. Note that, as defined, G can contain cycles. This turns out to be the main difficulty for this optimization problem, which, in the past, was usually tackled by a rather inefficient trick: "unfolding" the cycles and applying standard AND/OR search methods for acyclic graphs. However, explicit treatment of cycles has recently been considered, and an efficient algorithm is achieved in [8].

The search for an optimal height propagation is next reduced to this model. This amounts to (1) constructing an AND/OR graph G_{hp} whose feasible solutions define a height propagation, and (2) define a cost function that promotes non-degenerate propagations over degenerate ones.

4.2 Feasible height propagations

A height propagation can be defined by the following rules, with the given straightforward translation into AND/OR subgraphs.

R1: *Four selected vertices of the projection trigger the propagation.* For this, we put a TERMINAL node for each of the triggering vertices.

R2: *Every face in the polygonization can be determined once the heights of any three of its vertices are determined.* If $\deg(f)$ denotes the number of vertices of face f, then there are $c_f = \binom{\deg(f)}{3}$ possible combinations of three vertices determining f. If we put a node in G_{hp} for every vertex, except for the four triggering ones, then this rule is translated by adding an OR node for every face, linked to c_f new "dummy-face" AND nodes, each representing one of the above combinations. Each dummy-face node is in turn linked with arcs to the three involved vertices in the combination. Figure 8a gives a schematic representation. The newly introduced vertex nodes have not been assigned a type yet. This type is induced by the following rule.

R3: *Except for the initial four vertices, the height of every other vertex is determined once one of its incident faces has a determined plane.* This

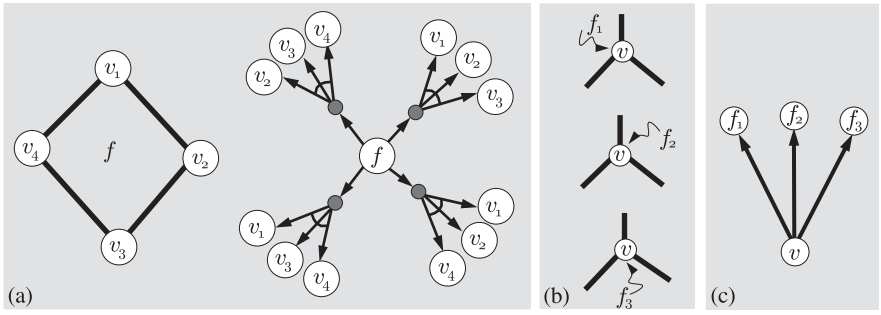

Fig. 8a–c. AND/OR subgraphs for the propagation rules. AND nodes are indicated by joining all their emanating arcs. **a** Constructed subgraph translating rule R2 for a quadrilateral face. Dummy-face nodes are shadowed in *grey*. Note that, actually, there is only one vertex node for each vertex in the trihedral mesh, but for clarity they are here duplicated. **b** Propagation waves reaching a vertex. **c** Subgraph for rule R3, with an arc for each of the possibilities in **b**

implements the fact that the propagation wave fixing the height of a vertex can come from any of its three incident faces (Fig. 8b). This rule can be represented by setting each vertex node to OR type, and linking it to the face nodes of its incident faces (Fig. 8c).

R4: *The height propagation must reach all vertices.* For this, we add a root AND node r to G_{hp} and link it to all vertex nodes.

Note that a feasible solution tree of G_{hp} provides instructions to derive a height propagation that reaches all vertices, starting at the four pre-specified heights.

4.3 Cost function

In order to penalize propagations using sets of almost-aligned vertices, we proceed as follows. Consider a height propagation that fixes a face-plane f from the point coordinates of three previously fixed vertices v_i, v_j and v_k. We can simply penalize the corresponding arcs in G_{hp} emanating from f by giving them a cost that is inversely proportional to the area of the triangle defined by v_i, v_j and v_k in the projection. The rest of arc costs are actually irrelevant, but need to be positively defined [8]. In sum, for every directed arc (u, v) we define its cost as follows:

1. $c(u, v) = 1/\det(v_i, v_j, v_k)$, if u is a dummy-face AND node and v is any one of its descendants. Here, v_i, v_j and v_k are the homogeneous coordinates of the vertices associated with the three descendants of u, with a one in the last coordinate.
2. $c(u, v) = 1$, otherwise.

Once the least-cost solution T is found, the projection can be made robust to slight vertex perturbations as follows. At a vertex v receiving more than one propagation wave, we put a T/TT-transformation on all faces fixing v, except on the one in the propagation wave represented in T.

4.4 Complexity analysis

The worst-case complexity of computing the optimal solution of a cyclic AND/OR graph with n nodes is $O(n^3)$ [8]. We now prove that the number of nodes in G_{hp} grows linearly with the number of vertices of the trihedral polygonal mesh.

Let e, v and f be the number of edges, vertices and faces of the given mesh. Then, $2e = 3v$ because the mesh is trihedral. Moreover, if the mesh has h holes, with "the outside" of the mesh counting as a hole too, then Euler's relation says that $v - e + f = 2 - h$. From these two equalities the number of faces of the mesh can be written in terms of the number of vertices and holes, $f = \frac{v+4}{2} - h$. Let us now count the number of nodes added by each of the rules R1, ..., R4:

- Rule R1 adds four vertex nodes.
- Rule R2 adds one OR node for each face, amounting to $f = \frac{v+4}{2} - h = O(v)$ total nodes, assuming a constant number of holes. Also, for every

face f this rule adds $c_f = \binom{\deg(f)}{3}$ dummy-face AND nodes. Although this number is clearly in the worst case $O(\deg(f)^3)$, if we divide the sum of face degrees by the number of faces, the average face degree is six, at an increasing number of randomly placed vertices in the mesh:

$$\frac{\sum_{\text{allfaces}} \deg(f_i)}{f} = \frac{3v}{\frac{v+4}{2} - h} = \frac{6v}{v+4-2h}$$

which will keep the number of dummy-face AND nodes linearly growing:

$$\binom{6}{3} f = 20\left(\frac{v+4}{2} - h\right) = O(v)$$

- Rule R3 adds a linear number of OR vertex nodes.
- Rule R4 only adds one AND node, the root.

Up to now we have assumed that the four vertices triggering the propagation are a priori selected. But other height propagations starting at other four vertices could yield better height propagations. To test all possibilities, we do not need to repeat the AND/OR search for every different combination of four vertices. Indeed, note that these vertices just fix the planes of the faces they belong to. So, any other set of four vertices on these faces will yield the same optimal propagations, provided that two of them lie on the common edge. We can equivalently think of pairs of faces triggering the propagation and use their face nodes as TERMINAL in G_{hp}. The choice of TERMINAL vertices (instead of TERMINAL faces) was done to be coherent with previous explanations. In sum, if one wants to search over all possible starting places of propagation, then for each pair of adjacent faces the AND/OR search needs to be repeated. This amounts to solving $e = \frac{3}{2}v$ optimization problems in the worst case, meaning that the overall complexity will be $O(v^4)$, under the assumption that the face degree is six.

5 Conclusion

We have shown how trihedral mesh projections can capture the spatial shape of a given object's surface, once the heights of four vertices are known, and we have given an algorithm to derive trihedral meshes from arbitrary input surfaces. We have also presented a local strategy that takes a trihedral projection as input and places some triangular faces at strategic places until its reconstructibility is made robust to perturbations in its vertex coordinates. Finally, we have found how to put these triangles so that the spatial reconstruction is performed in the most accurate way possible, avoiding height propagations along degenerate paths.

One issue that must be solved is that errors accumulate from one vertex to another along a height propagation, as shown in Fig. 7. The AND/OR search algorithm reduces these errors but, for the same purpose, one could additionally extend the number of vertices for which a Z value is given. Optimizing the position of such vertices is almost certainly intractable, but simple heuristics could be used to add vertices to a "specified Z" set whenever errors accumulated along a propagation path become unacceptable. This point deserves further attention.

The main issue currently concentrating our efforts is to find an enhancement of (or an alternative to) the trihedrization algorithm presented in Sect. 2. Clearly, as observed in the middle row of Fig. 4, the obtained polygonization contains small polygons, almost orthogonal to the input surface. This kind of "staircase effect" may produce numerical instabilities when attempting to reconstruct the spatial shape from its projection, and should be avoided. A possible solution might be found by using projective polarities [4]. It is well known that the projective polar of a triangulated polyhedron (with respect to a quadric) is a trihedral polyhedron. Thus, in principle, to get a trihedral mesh of a surface $\mathcal{S}$, one could derive an auxiliary surface $\mathcal{S}^*$, the polar of $\mathcal{S}$ through a polarity $\mathbf{P}$ [7], and then apply $\mathbf{P}^{-1}$ to a triangulation of $\mathcal{S}^*$, to obtain a trihedral mesh approximating $\mathcal{S}$. We believe this is a promising method to increase the smoothness of the obtained polygonization.

Acknowledgements. We are very grateful to an anonymous reviewer for his helpful comments, and to Francisco Pérez and Pablo Jiménez for fruitful discussions on several parts of the paper. This work has been partially funded by the Spanish CICYT under contracts TIC-2000-0696 and TAP99-1086.

References

1. Barber CB, Dobkin DP, Huhdanpaa HT (1996) The quickhull algorithm for convex hulls. ACM Trans Math Softw 22(4):469–483
2. Boor C (1987) Cutting corners always works. Comput Aided Geom Des 4:125–131
3. Chuang JH, Hoffmann CM, Ko KM, Hwang WC (1998) Adaptive polygonization of geometrically constrained surfaces. Vis Comput 14(10):455–470

4. Coxeter HSMcD (1987) Projective Geometry, 2nd edition. Springer, Berlin Heidelberg New York
5. Fox D, Joy KI (1998) On polyhedral approximations to a sphere. Proceedings of the IEEE Computer Graphics International Conference, pp 426–432, IEEE Press
6. Heckbert P, Garland M (1997) Survey of polygonal surface simplification algorithms. In: Multiresolution Surface Modeling Course SIGGRAPH'97, May 1997. Available at http://www.cs.cmu.edu/∼ph
7. Hoschek J (1983) Dual Bézier curves and surfaces. In: Barnhill RE, Boehm W (eds), Surfaces in Computer Aided Geometric Design. North-Holland, Amsterdam, pp 147–156
8. Jiménez P, Torras C (2000) An efficient algorithm for searching implicit AND/OR graphs with cycles. Artif Intell 124:1–30
9. Qhull (2002) Qhull home page at the Geometry Center, University of Minnesota. http://www.geom.umn.edu/software/qhull/. Cited 17 July 2002
10. Ros L (2000) A kinematic-geometric approach to spatial interpretation of line drawings. PhD thesis, Polytechnic University of Catalonia. Available at http://www-iri.upc.es/people/ros
11. Ros L, Thomas F (2002) Overcoming superstrictness in line drawing interpretation. IEEE Trans Pattern Anal Mach Intell 24(4):456–466
12. Seibold W, Wyvill G (1998) Towards an understanding of surfaces through polygonization. Proceedings of the IEEE Computer Graphics International Conference, pp 416–425, IEEE Press
13. Sugihara K (1984) An algebraic approach to shape-from-image problems. Artif Intell 23:59–95
14. Sugihara K (1986) Machine interpretation of line drawings. MIT Press, Cambridge, Mass.
15. Sugihara K (1999) Resolvable representation of polyhedra. Discrete Comput Geom 21(2):243–255
16. Whiteley W (1988) Some matroids on hypergraphs with applications to scene analysis and geometry. Discrete Comput Geom 4:75–95
17. Whiteley W (1994) How to design or describe a polyhedron. J Intell Rob Syst 11:135–160
18. Wünsche B, Lobb R (1999) Triage polygonization of rounded polyhedra. Vis Comput 15(1):36–54

Photographs of the authors and their biographies are given on the next page.

LLUÍS ROS was born in Vilafranca del Penedès (Barcelona, Catalonia), on June 30, 1968. He received the mechanical engineering degree in 1992, and the Ph.D. degree (with honors) in industrial engineering in 2000, both from the Polytechnic University of Catalonia. From 1993 to 1996 he worked with the Control of Resources group of the Cybernetics Institute in Barcelona, involved in the application of constraint logic programming to the control of electric and water networks. Since August 2000, he has been a Research Scientist in the Institut de Robòtica i Informàtica Industrial of the Spanish High Council for Scientific Research. His current research interests are in geometry and kinematics, with applications to robotics, computer graphics and machine vision. His web page is `http://www-iri.upc.es/people/ros`.

KOKICHI SUGIHARA received B.Eng., M.Eng. and Dr. Eng. degrees in mathematical engineering from the University of Tokyo in 1971, 1973 and 1980 respectively. In 1986 he moved to the Department of Mathematical Engineering and Information Physics of the University of Tokyo, and he is now a professor of the Department of Mathematical Informatics of the Graduate School of Information Science and Technology of the University of Tokyo. His research interests include computational geometry, computer graphics and computer vision. He is a member of the Information Processing Society of Japan, the Operational Research Society of Japan, Japan SIAM, IEEE and ACM.

FEDERICO THOMAS was born in Barcelona, on October 5, 1961. He received the B.S. degree in telecommunications engineering in 1984, and the Ph.D. degree (with honors) in computer science in 1988, both from the Polytechnic University of Catalonia. Since March 1990, he has been a Research Scientist in the Institut de Robòtica i Informàtica Industrial of the Spanish High Council for Scientific Research. His research interests are in geometry and kinematics, with applications to robotics, computer graphics and machine vision. He has published more than forty research papers in international journals and conferences. His web page is `http://www-iri.upc.es/people/thomas`.