Pressing: Smooth Isosurfaces with Flats from Binary Grids

A. Chica, J. Williams[2], C. Andujar, P. Brunet, I. Navazo, J. Rossignac[2], A. Vinacua
Department of Software, Polytechnic University of Catalonia, Barcelona, Spain
[2]IRIS Cluster and GVU Center, College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

Abstract

We explore the automatic recovery of solids from their volumetric discretizations. In particular, we propose an approach, called Pressing, for smoothing isosurfaces extracted from binary volumes while recovering their large planar regions (flats). Pressing yields a surface that is guaranteed to contain the samples of the volume classified as interior and exclude those classified as exterior. It uses global optimization to identify flats and constrained bilaplacian smoothing to eliminate sharp features and high-frequencies from the rest of the isosurface. It recovers sharp edges between flat regions and between flat and smooth regions. Hence, the resulting isosurface is usually a much more accurate approximation of the original solid than isosurfaces produced by previously proposed approaches. Furthermore, the segmentation of the isosurface into flat and curved faces and the sharp/smooth labelling of their edges may be valuable for shape recognition, simplification, compression, and various reverse engineering and manufacturing applications.

1 Introduction

Consider a solid model M whose boundary ∂M is smooth and contains large planar faces (called flats). The set of samples (nodes) of a regular axis-aligned lattice in a box containing M may be divided into the set G of green samples in M and the set R of red samples out of M. The collection of cubical voxels centered at the green samples provides a rough approximation of M.

Different from voxels, the *cells* of the grid are axis-aligned boxes having for vertices a $2\times2\times2$ arrangement of neighboring samples from the grid. Cells with vertices of different colors are said to be *mixed*. The axis-aligned edges of the lattice connecting adjacent nodes of different color are called *sticks*. Let the free space F be the union of the mixed cells, and let S be a triangulated isosurface in F that separates R from G and has as its vertices the midpoints of the sticks (Figure 1-a). We explore isotopies in F that will deform S into S' by sliding its vertices along their sticks. In particular, we strive to increase the smoothness of S and at the same time to reproduce in S' close approximations of the flats of ∂M, using only the information encoded by G and R. The resulting surface S' (Figure 1-c) is called the *pressed* S, and the process for computing it is called *Pressing*.

In both two and three dimensions, Pressing starts by grouping the sticks into clusters that can each be stabbed by a flat separating the red and green samples at the sticks' endpoints. Then Pressing slides the sticks' vertices to their intersections with the flat and *freezes* them. The other *fresh* (non-frozen) vertices are then adjusted along their sticks to smooth the isosurface.

We first illustrate this process with a 2D example. A region M with a boundary ∂M that contains several flats (line segments) (Figure 2-a) is rasterized (Figure 2-b) on a regular axis-aligned lattice by painting green

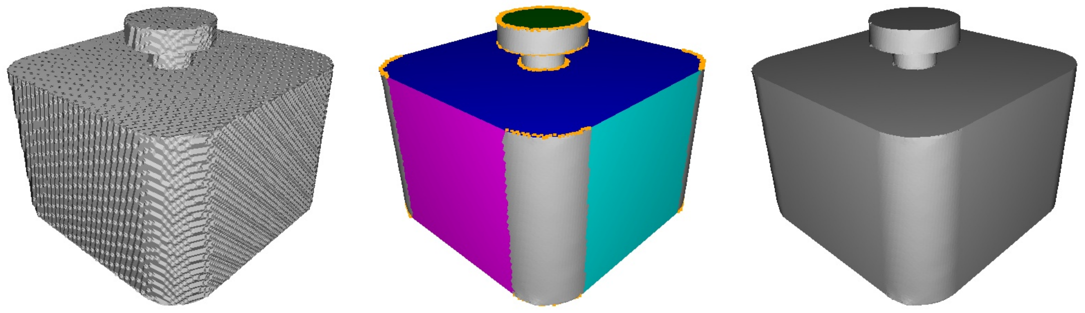

Figure 1: (a) The aliased isosurface was extracted from a 128^3 binary voxelization. Its vertices are at the stick midpoints. (b) The flats were identified and color-coded. The junction points along the boundaries between planar and non-planar regions are also identified and shown as orange dots. Note that a flat may be connected to other flats or to smooth faces through sharp edges and to smooth faces through smooth edges. (c) The resulting pressed isosurface.

the lattice nodes in M and red the other ones.

Pressing starts from this red/green labeling and reconstructs an isocontour S (Figure 2-c) approximating ∂M. The mixed cells are the lattice squares having both red and green vertices. S is contained in their union. A mixed cell is bounded by two or four sticks. A cell with two sticks contributes a single edge to S. A cell with four sticks is called an *X-face*. It contributes two edges to S. There are two ways of constructing these two edges so that they do not intersect. The choice affects the topology of the result. We have dealt with this problem in three dimensions in [ABC*05], which is the solution we will adopt here. We will not dwell further in this issue, as the algorithm presented in this paper will work equally with any correct starting triangulation. For the purpose of this illustration, one can assume we have an oracle that decides for us the connectivity to use.

Next, Pressing identifies the flats. It associates each flat with a subset of the sticks it stabs, so that each stick is associated with at most one flat. Then it snaps the vertices of these sticks to the flat that stabs them, by sliding these vertices along their stick (Figure 2-d). These vertices will remain locked in this position (we say that they are *frozen*). The remaining vertices are said to be fresh. Then Pressing identifies *junction* points among the fresh vertices adjacent to frozen vertices; these will correspond to sharp corners. Finally, Pressing perturbs the fresh vertices through a custom smoothing process that retains sharp corners and produces a polygonal curve S' (Figure 2-e) that is isotopic to S (i.e., may be continuously deformed into S without crossing any lattice point). Note that S' is a close approximation of ∂M and has straight lines, smooth curves, and sharp corners that match those of ∂M.

In a similar fashion, the 3D version of the Pressing algorithm works as follows:

- We cluster sticks [ABC*04] into flats that may be stabbed by a plane (Figure 1-b).
- We identify junction points: vertices between a flat and a curved region (Figure 1-b). (Section 3)
- We fair non-flat regions by an iterative process, which, at each step and for each fresh vertex combines arc-length re-sampling, bilaplacian smoothing, and snapping. These three operations are performed independently on each X, Y, and Z slice [NGH*03]. Their results are combined for each stick using a special filter at borders. (Section 4)

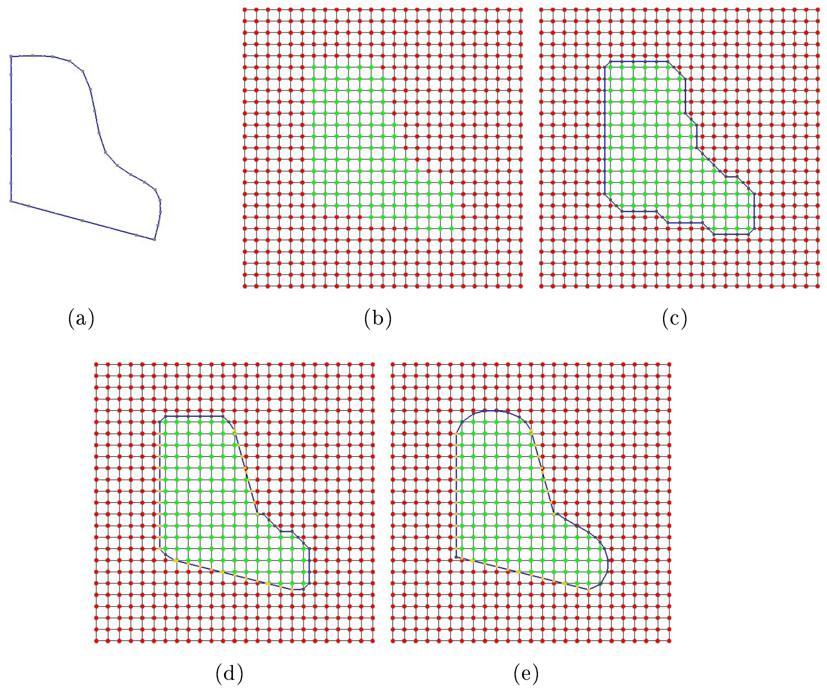

Figure 2: 2D region M bounded by straight and curved edges (a) Red/green classification of grid-samples produced by rasterizing M. (b) Reconstructed isocontour S with vertices at stick midpoints. (c) Straight edges are recovered by snapping the vertices of each cluster to its flat. (d) A pressed version S' of S is obtained after 50 pressing iterations. (e) Its straight and curved edges correspond to those of M.

- Sharp edges are recovered through the use of a modified Edge Sharpener algorithm [AFRS04, AFSR05] (Figure 1-c). (Section 5)

In summary, the technical contributions presented in this paper are:

- We propose to combine segmentation (for recovering flats) and smoothing (for joining them with smooth transition surfaces).
- We modify iso-surfaces smoothing to preserve portions frozen by equality constraints.
- We present a new smoothing operator, which preserves the connectivity of the initial iso-surface representation and achieves smoothness in the presence of equality and inequality constraints.

The combination of these advances leads to new functionalities:

- Flat regions, curved regions, and sharp edges can be automatically recovered from raw binary voxelizations even though scalar field and Hermite data are not available.
- The reconstruction error is therefore bounded, since the final isosurface is constrained to stab the initial sticks.

- The isosurface is automatically segmented into flat and curved regions, which may facilitate shape identification and manufacturing and assembly planning.

2 Prior Art

Algorithms that extract isosurfaces from a discrete sampling of a scalar field on a regular grid strive to ensure topological consistency and geometric fidelity [LC87, NFHL91, Nie03, Lac96, MSS94, ABC*05]. Some approaches base topological decisions on scalar field values [CGMS00, Nie03, LB03] or Hermite data [HWC*05], i.e. the estimates of surface normals at the vertices. Geometric fidelity often depends on the delicate ability to recover sharp features (which may follow smooth curved edges) and to smooth the isosurface away from these edges. The Extended Marching Cubes [KBUS01] detects cells containing features and recovers them by inserting an additional point in each one of these cells. The Dual Contouring algorithm [JLSW02] uses a quadratic error metric to compute a new point inside each of the four cells around each mixed stick and generates a quad connecting these four new points. A similar approach is adopted in [VKSM04]. All these methods make use of Hermite data and thus cannot be applied to recover sharp features on binary grids where only the in/out classification of the grid nodes is stored.

Mesh smoothing algorithms strive to remove noise and high-frequency details from a general triangulated surface by the iterative application of a smoothing operator. Most approaches derive non-shrinking smoothing operators from discrete approximations of the Laplacian [Tau95, Kob97, DMSB99].

A major concern of smoothing techniques is the addition of constraints on the vertex placement to guarantee the separation of in/out grid nodes [Gib98, Whi00, NGH*03, Nie04]. Gibson [Gib98] proposed an algorithm for reducing the terracing artifacts in isosurfaces extracted from binary grids. It places one vertex in each mixed cell and then links the vertices in face-connected mixed cells to form a net. This net is relaxed to reduce the energy measure in the links. A constraint is applied to keep each node in this original cell. After the relaxation a triangulated surface, which may not be a manifold, is generated in a straightforward way. Nielson et al. [NGH*03] proposed a closely related technique to our constrained smoothing approach. Like Pressing, mesh vertices are moved along the sticks and surface smoothing is obtained by combining two orthogonal polygonal smoothing operations. However, the displacement for each vertex in [NGH*03] is driven by a non-linear optimization algorithm that minimizes an energy function defined on each polygonal curve. A different approach is presented in [Nie04] which introduced a smoothing operator based on the dual of the dual surface of a Marching Cubes mesh, locating the vertices at the intersections of the dual's quads with the lattice edges.

The computation of planar regions (flats) approximating a given geometric model is an important problem with wide applications in computer vision, modeling and impostor-based simplification. The technique we apply for detecting flats is related to superfaces [KT94] and face clustering [GWH01, She01], which group connected sets of nearly coplanar faces of a given triangulated surface. Superfaces uses a greedy algorithm to cluster triangles. Each triangle in a superface imposes constraints on the set of feasible approximating planes for the superface, most notably that the triangle's vertices must be within a fixed distance of the planes. Hierarchical face clustering [GWH01] uses quadric error metrics [GH97] to iteratively merge adjacent faces. Cohen-Steiner et al. [CSAD04] adopt a variational geometric partitioning approach to group faces into best-fitting regions according to a normal deviation error metric. Decoret et al. [DDSD03] use an optimization algorithm to find a set of approximating planes, using a discretization of a plane parameterization in spherical coordinates, and propose a greedy optimization algorithm of a density field in this plane parameterization. The main drawback is the time complexity of the plane optimization algorithm and the lack of uniformity in the parameterization of planes.

Unlike the approaches above, which require a triangulated surface, Andujar et al. [ABC*04] propose an efficient algorithm for the computation of the largest flat region (*tile*) in a discrete geometric model. The input of the algorithm is the set of sticks. Using a voting-based approach, the plane that slices a largest number of sticks is computed. The robustness and efficiency of the approach rests on the use of two different parameterizations of the planes with suitable properties. The first of these is exact and is used to retrieve precomputed local solutions of the problem. The second one is discrete and is used in a hierarchical voting scheme to compute the global maximum. The authors demonstrate the merits of the algorithm for efficiently computing an optimized set of textured impostors for a given discretization of a polygonal model.

3 Detection of flats and junction points

The first step of Pressing identifies clusters of sticks and freezes their vertices on the best plane fit for each cluster. To do so, we traverse the sticks, which cast a vote according to their neighborhoods, using the approach proposed in [ABC*04] for computing maximum tiles (flat regions).

As shown in Figure 1-b, the boundary of a flat F may include three types of edges. (1) Edges connecting F to another flat. (2) Sharp edges connecting F to a curved surface. (3) Smooth edges connecting F to a curved surface with normal continuity. Edges of type (1) and (2) will be identified as *sharp edges* and preserved. Edges of type (3) will be faired by our smoothing algorithm.

To detect cases where (2) or (3) apply, and before proceeding to the smoothing step, we consider every edge of S joining a frozen vertex V_p on a planar region and a fresh vertex V_s on a smooth region. Let N_p be the normal of the tile stabbing V_p. Let N_s be normal to the plane produced by a least square fit to the fresh vertices on the 2-ring neighborhood around V_s. When the angle between N_p and N_s exceeds 30 degrees, we decide that the edge is a chamfer-edge cutting through a sharp edge of S and label V_s as a junction point. After experimenting with various angles, we have empirically concluded that a 30 degree threshold leads to the best compromise limiting the false positives and false negatives.

4 Smoothing and snapping

In the next step, we seek to smooth out the surface obtained thus far, with a special consideration for sharp edges between smooth and planar portions. To this end, we iterate the following two steps of our approach: Smoothing and Snapping. Together, they iteratively modify the positions of the fresh vertices.

The 3D smoothing step computes for each fresh vertex C of S a displacement vector W along the stick I of C. W is obtained as a linear combination of the two displacements, computed in each of the two different axis-aligned slices of the grid that contain I.

Consider one such slice. Assume that C lies on a curve where the slice intersects S. To compute the corresponding displacement, we have developed a variant of the bilaplacian smoothing, which uses two points at a fixed arc-length distance from C along the curve to each side of C. The construction is explained below. The resulting displacement is projected onto the line supporting the stick I.

Consider five consecutive vertices (A, B, C, D, E) along a polygonal curve. The bilaplacian smoothing displacement vector $L^2(C)$ associated with vertex C (Figure 3) may be computed as:

$$L^2(C) = \frac{-A + 4B - 6C + 4D - E}{4} \tag{1}$$

We first precompute and store $L^2(C)$ for each vertex C. Then, we move each vertex C to $C'' = C + s^2 L^2(C)$, using $s = 0.85$. Note that this bilaplacian smoothing is equivalent to performing Taubin's

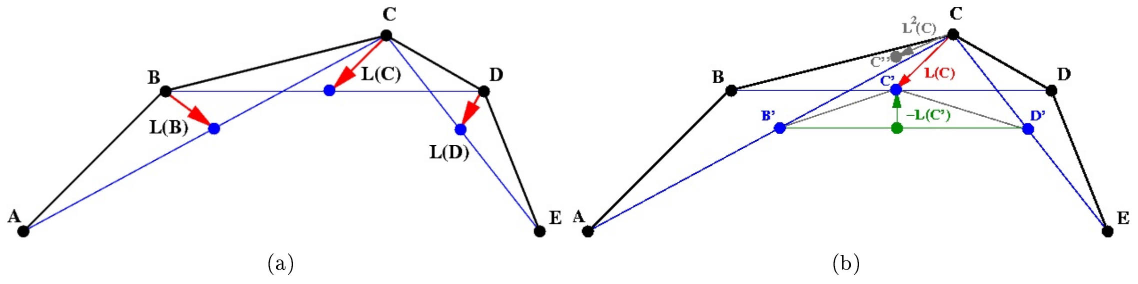

Figure 3: The applied bilaplacian smoothing $L^2(C)$ can be computed as the difference $L(C) - L(C')$ of two Laplacian displacements. $L(C)$ moves C to the average C' of its neighbors. Now assume that B and D have also been moved to the averages B' and D' of their neighbors. $L(C')$ moves C' to the average C'' of B' and D'.

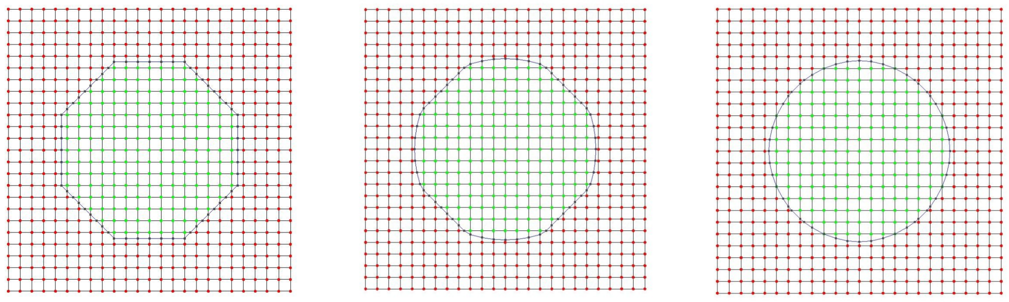

Figure 4: The color-coding of the nodes was obtained by rasterizing a circle. Computing the bilaplacian from neighboring vertices and snapping does not converge to an acceptable approximation of the circle (center). Applying arc-length resampling yields a much better fit (right).

filtering [Tau95] with $\lambda = \mu = s$.

Next, the snapping step projects each displaced vertex C'' to the closest point on the line supporting the stick it came from. But this naïve approach does not converge to the smoothest curve possible subject to the constraints (Figure 4). We found that, rather than approaching zero, some displacement vectors eventually become orthogonal to the sticks, so that projecting a displaced vertex simply returned it to its previous position. Hence, the curve is stuck in a suboptimal shape. Furthermore, when the curve converges to a node of the grid, the node imposes inequality constraints on the displacements of vertices of incident sticks. Thus as two or three of these vertices converge towards the node, the arc-length parameterization of the samples along the curve is no longer uniform. Because the formula in equation 1 was developed for converging to a uniform parameterization, it performs poorly near these nodes, creating sharp discontinuities. To solve this problem, we have explored a variety of alternatives, including using a more general cubic fit to non-uniformly distributed samples, whose parameters in the parametric cubic expression are estimated from the arc-length distances between vertices. We have concluded that the resampling approach described below leads to the most effective smoothing. Hence when computing the bilaplacian $L^2(C)$, we do not use A, B, D, and E — the neighboring vertices of C on the curve. Instead, we compute new samples at a fixed distance d along the

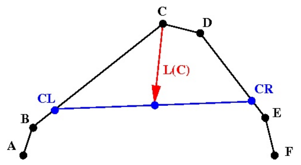

Figure 5: Instead of using the vertices B and D, two virtual neighbors (CR and CL) are computed on each side of C by moving a fixed amount along the curve .

curve in both directions (Figure 5). This arc-length resampling prevents the formation of unwanted corners and yields satisfactory results when used with the snapping. We select d to be 75% of the length of a cell's side.

Next we turn to the combination of these results into a displacement W. We could compute the displacement for each vertex one by one. For example, consider the vertex V on a stick I that is parallel to the Z axis. We could compute its displacements in the $X - Z$ section of the grid that contains I and on the $Y - Z$ section. Then we would combine these displacements ensuring that the resulting vertex remains on I. Instead, for simplicity and implementation efficiency, we perform the smoothing and line-snapping steps on each $X - Y, X - Z$, and $Y - Z$ slice and then collect the results, combining two displacements for each stick and clamping the result to the stick.

Because the stick of each vertex belongs to exactly two slices [NFHL91], we have two suggested displacements for each vertex. We average the two displacement vectors to obtain the vertex's displacement and then snap the displaced vertex to the closest point onto its stick. When averaging the two displacements different weighting techniques can be applied. We have experimented with three methods: (1) equal weights, (2) weights are proportional to the dot products of the stick tangent with the normal to the curve, and (3) weights are proportional to the dot products of the stick tangent with the tangent to the curve. There are special cases in which (2) yields markedly worse results, and cases where (3) does that. We found that the simpler approach in case (1) uniformly yields good results that are close to those of the best of the other two.

We have modified the smoothing filter for vertices that belong to a feature edge. Those vertices do not use the bilaplacian. Instead, they align themselves with the vertices on the smooth side (Figure 6). This amounts

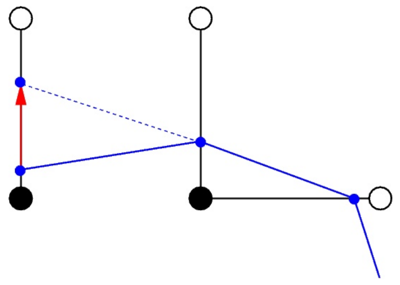

Figure 6: The "smooth" vertices on a feature edge move themselves along their stick to be aligned with the neighbors on their "smooth" side.

to aim at surfaces that approach the junction with zero curvature. In the absence of any information on this junction, we have adopted this unclamped approach, which has been used in generating all the pictures.

5 Sharp Edge Recovering

The errors between the original shape and the iso-surface recovered thus far are usually concentrated near the features, which were not appropriately captured by the regular sampling. To improve the accuracy of the recovered surface, we sharpen the boundaries of planar regions using a variant of EdgeSharpener [AFRS04, AFSR05].

The vertices belong to either a tile (planar face) or to a curved face. We use the term chamfer edges for those mesh edges with vertices on two different faces. A triangle with one or more chamfer edges is called chamfer triangle.

In order to recover the sharp features we apply three steps. First, the chamfer triangles are identified. Then, we subdivide them appropriately by inserting new vertices. Finally, we position the new vertices to better recover the sharp features.

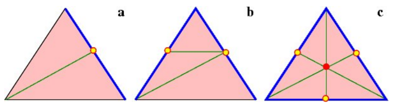

Figure 7: Subdivision of a chamfer triangle with one (a) two (b) or three (c) chamfer edges.

Three cases arise when subdividing the chamfer triangles (Figure 7):

1. Triangles with one chamfer edge are split into two triangles.
2. Triangles with two chamfer edges result in three triangles.
3. When three different faces meet at a triangle we have three chamfer edges. After subdividing we will have six triangles and one interior vertex to represent the corner.

The process is presented in Figures 8 and 9. To find the position of a new vertex V inserted in a chamfer

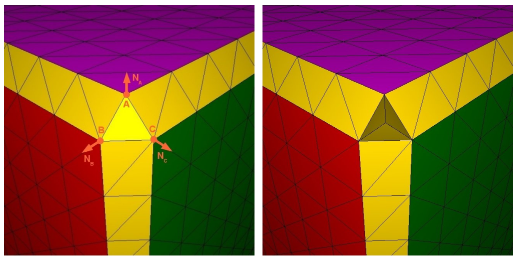

Figure 8: Inserting a new vertex on a triangle with its vertices on three different faces.

edge E, we consider the two original vertices, A and B, of E. We compute a normal N_A for the vertex A using its face, and define a plane P that is orthogonal to N_A and passes through A. Similarly, we compute a normal N_B for the vertex B using its face, and define a plane Q that is orthogonal to N_B and passes through B. Finally, we move V to the closest point on the line of intersection between planes P and Q. When one of these vertices belongs to a curved face, its normal is taken to be the normal to a plane estimated as in the computation of junction points (the minimum square fit to the free neighbors in a 2-ring).

To find the position of an interior vertex W, we consider the vertices A, B and C of the corner triangle. We compute normals N_A, N_B and N_C using their respective faces. Using them we define planes P, Q and

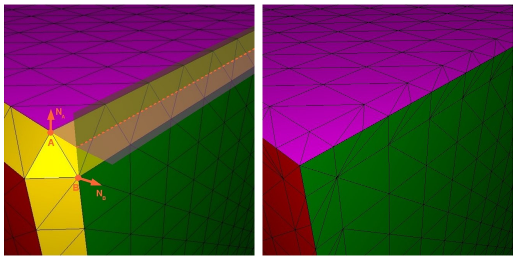

Figure 9: Edges with vertices on two different faces are subdivided to recover a feature.

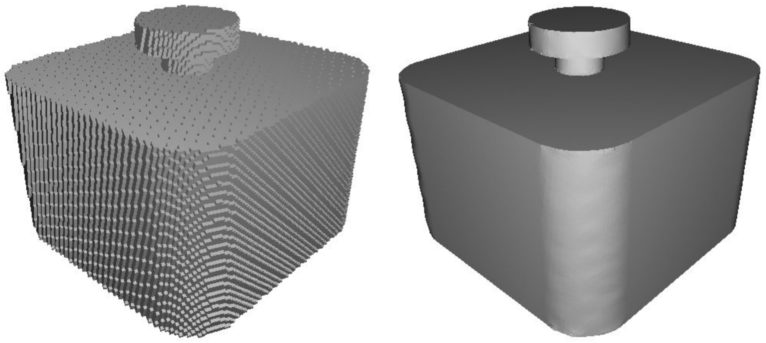

Figure 10: A $128 \times 128 \times 128$ voxelization of a pump model, and the final pressed isosurface.

R which are orthogonal to N_A, N_B and N_C, and pass through A, B and C, respectively. Then, W is moved to the intersection point of P, Q and R.

6 Results and discussion

In this section we present and discuss several examples. They are shown in Figures 10, 11, 12 and 13. The initial surfaces of these models have been converted into a binary voxel representation, and these voxelizations have been used as the input to Pressing. In order to show the performance of Pressing in the most general case, we have intentionally applied a random rotation to the models to ensure that the main faces are not axis-aligned.

All these models show several sharp edge features between flat regions. Although the initial information is only a binary voxelization, our algorithm is successful in detecting and reconstructing flats and sharp edges. In Figures 10 and 11, Pressing also recovers the smooth regions of the model and the curved features between flat and smooth pieces. The following tables summarize the performance of the algorithm to obtain the results depicted in Figures 10, 11, 12 and 13. Notice —by comparing Figure 10 with Figure 1-c— that a larger number of iterations can achieve still smoother results, at the expense of time.

Table 1 presents the computing time of our algorithm, running on a Pentium 4 at 3.4 GHz and 1 GB of RAM. The Max Tiles step for the detection and reconstruction of flat regions is based on a previous work and is not presented as a contribution in this paper. The times for detecting sharp features and edge

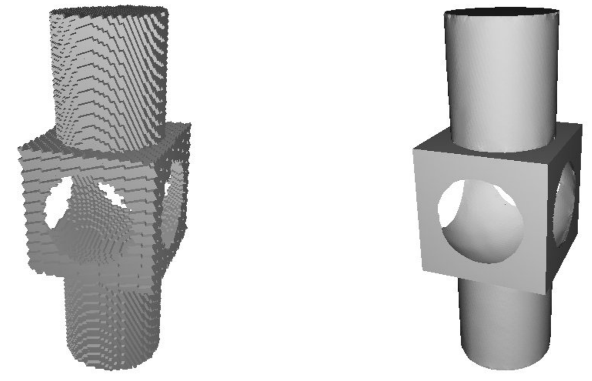

Figure 11: A $128 \times 128 \times 128$ voxelization of a mechanical part, and the final pressed isosurface.

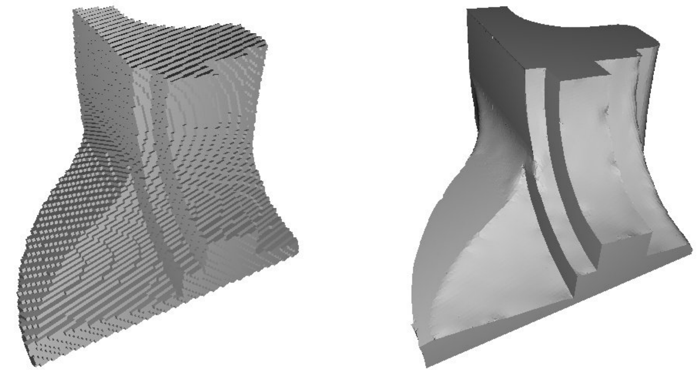

Figure 12: A $128 \times 128 \times 128$ voxelization of the Fandisk model, and the final pressed isosurface.

sharpening are not significant in front of the time complexity of the smoothing part of the algorithm. This smoothing step is however below 10 seconds in all $128 \times 128 \times 128$ models. The higher times in the PART2 model are a consequence of the finer voxelization.

Table 2 presents the evolution of the reconstruction errors, which are computed by taking the average of the squares of the distances between a vertex of the isosurface and the intersection of its stick with the original model. Note that this measure provides an upper bound on the Haussdorf error. The table presents the square roots of these values. Voxels are considered to be of size $1.0 \times 1.0 \times 1.0$.

The first row measures the error between the scanned isosurface and the midpoint isosurface, the second one adds the planar regions, while the third and the fourth ones include the smoothing with 100 and 1000 iterations each. The errors monotonically decrease at each Pressing step, and reach small values after only 100 smoothing operations. The increase of accuracy when the number of iterations goes from 100 to 1000 is not significant.

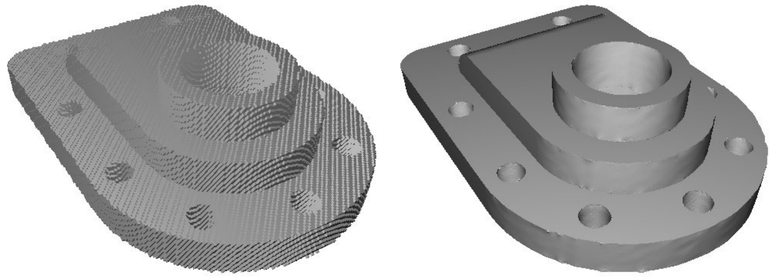
Figure 13: A $256 \times 256 \times 256$ voxelization of a second mechanical part, and the final pressed isosurface.

Stats (times in seconds)	PUMP	MECHANICAL PART	COMPLEX	FANDISK
TIME (MaxTiles)	73.39	15.99	94.38	58.89
TIME (Detect Sharp Features)	0.314	0.185	0.951	0.277
TIME (Smoothing)	8.158	5.563	45.562	5.643
TIME (Edge Sharpening)	0.129	0.108	0.397	0.141
# Tiles	8	8	10	11
# Iterations	49	65	97	54

Table 1: Computing time, number of reconstructed flat regions and number of iterations for each of the presented models.

7 Conclusions

We have proposed a novel smoothing approach for the automatic recovery of solids from binary volumetric discretizations. Our approach uses global optimization to identify flats and a constrained smoothing algorithm to recover the shape of non-planar regions. The proposed smoothing algorithm involves a snapping step after each bilaplacian smoothing step to guarantee that final vertices remain in the initial sticks of the voxelization.

Pressing works on general binary voxelizations and can recover flat and curved regions in cases where no scalar field data or Hermite data are available. The isosurface is automatically segmented by sequences of junction points and it is constrained to stab the initial sticks. The reconstruction error is therefore bounded and the topology is preserved.

We use a three dimensional implementation of the constrained smoothing, which combines two two-dimensional smoothing steps for each vertex, one in each axis-aligned plane containing the vertex's stick, followed by snapping. A special version of the filter for vertices at the borders of the curved regions has been also developed.

Results on a variety of models have been reported and discussed. Pressing achieves small reconstruction errors and successfully recovers flats and sharp features in a reasonable amount of time.

Potential applications include shape recognition, simplification, compression and various reverse engineering and manufacturing problems.

Error	PUMP	MECHANICAL PART	COMPLEX	FANDISK
Midpoint	0,2887	0,2837	0,2874	0,2882
Midpoint + Tiles	0,1254	0,2267	0,1706	0,1835
MidPoint + Tiles + Smoothing (100)	0,0461	0,0854	0,0715	0,0945
MidPoint + Tiles + Smoothing (1000)	0.0416	0,0827	0,0677	0,0868

Table 2: Average square error after the different steps of the Pressing algorithm for each of the presented models.

8 Acknowledgements

Rossignac and Williams's research on this project has been supported by the National Science Foundation under Grant 0138420.

Andujar, Brunet, Chica, Navazo and Vinacua's research on this project has been supported by the CICYT Spanish Agency under Grant TIN-2004-08065-C02-01

Chica's Research has also been supported by a Graduate Research Fellowship from the Spanish Government.

References

[ABC*04] Andújar C., Brunet P., Chica A., Navazo I., Rossignac J., Vinacua A.: Computing maximal tiles and application to inpostor-based simplification. *Computer Graphics Forum 23*, 3 (2004). Proceedings of Eurographics'04.

[ABC*05] Andújar C., Brunet P., Chica A., Navazo I., Rossignac J., Vinacua À.: Optimazing the topological and combinational complexity of isosurfaces. *Computer-Aided Design 37*, 8 (2005), 847–857.

[AFRS04] Attene M., Falcidiano B., Rossignac J., Spagnuolo M.: Edge-sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces. In *Proc. of EG/ACM SIGGRAPH Symposium on Geometry Processing 2004* (2004), pp. 62–69.

[AFSR05] Attene M., Falcidiano B., Spagnuolo M., Rossignac J.: Sharpen&blend: Recovering curved edges in triangle meshes produced by feature-insensitive sampling. *IEEE Transactions on Visualization and Computer Graphics 11*, 2 (2005), 83–91.

[CGMS00] Cignoni P., Ganovelli F., Montani C., Scopigno R.: Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computers and Graphics 24*, 3 (2000), 399–418.

[CSAD04] Cohen-Steiner D., Alliez P., Desbrun M.: Variational shape approximation. *ACM Trans. Graph. 23*, 3 (2004), 905–914.

[DDSD03] Décoret X., Durand F., Sillion F. X., Dorsey J.: Billboard clouds for extreme model simplification. *ACM Transactions on Graphics 22*, 3 (July 2003), 689–696.

[DMSB99] Desbrun M., Meyer M., Schröder P., Barr A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 317–324.

[GH97] Garland M., Heckbert P. S.: Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 209–216.

[Gib98] Gibson S.: Constrained elastic surface nets: generating smooth surfaces from binary segmented data. In *MICCAI'98, Medical Image Computation and Computer Assisted Surgery* (1998).

[GWH01] Garland M., Willmott A., Heckbert P. S.: Hierarchical face clustering on polygonal surfaces. In *Proceedings of ACM Symposium on Interactive 3D Graphics* (Mar. 2001), ACM Press.

[HWC*05] Ho C.-C., Wu F.-C., Chen B.-Y., Chuang Y.-Y., Ouhyoung M.: Cubical marching squares: Adaptive feature preserving surface extraction from volume data. *Computer Graphics Forum (Eurographics 2005) 24*, 3 (2005), 537–546.

[JLSW02] Ju T., Losasso F., Schaefer S., Warren J.: Dual countouring of hermite data. *ACM Transactions on Graphics 21*, 3 (2002), 339–346. Proc of Siggraph'02.

[KBUS01] Kobbelt L. P., Botsch M., U. Schwanecke H. P. S.: Feature sensitive surface extraction from volume data. *ACM Computer Graphics (Siggraph 2001)* (2001), 57–66.

[Kob97] Kobbelt L.: Discrete fairing. In *Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces* (1997), pp. 101–131.

[KT94] Kalvin A. D., Taylor R. H.: Superfaces: Polyhedral approximation with bounded error. In *Medical Imaging: Image Capture, Formatting, and Display* (Feb. 1994), vol. 2164, SPIE, pp. 2–13. (Also IBM Watson Research Center tech report RC 19135).

[Lac96] Lachaud J.-O.: Topologically defined iso-surfaces. In *Proc. 6th Discrete Geometry for Computer Imagery (DGCI'96), Lyon, France* (1996), Springer-Verlag, Berlin, pp. 245–256.

[LB03] Lopes A., Brodlie K.: Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (2003), 16–29.

[LC87] Lorensen W., Cline H.: Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics 21*, 4 (1987), 163–169.

[MSS94] Montani C., Scateni R., , Scopigno R.: Discretized marching cubes. In *IEEE Visualization* (1994), pp. 281–287.

[NFHL91] Nielson G., Foley T., Hamann B., Lane D.: Visualizing and modeling scattered multivariate data. *IEEE Computer Graphics and Applications 11*, 3 (1991), 47–55.

[NGH*03] Nielson G., Graf G., Holmes R., Huang A., Phielipp M.: Shrouds: Optimal separating surfaces for enumerated volumes. In *EG-IEEE TCVG Symposium on Visualization 2003* (2003), pp. 75–84.

[Nie03] NIELSON G.: On marching cubes. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (2003), 283–297.

[Nie04] NIELSON G.: Dual marching cubes. In *IEEE Visualization 2004* (2004), pp. 489–496.

[She01] SHEFFER A.: Model simplification for meshing using face clustering. *Computer Aided design 33*, 13 (2001), 925–934.

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM Press, pp. 351–358.

[VKSM04] VARADHAN G., KRISHNAN S., SRIRAM T., MANOCHA D.: Topology preserving surface extraction using adaptive subdivision. In *Proc. of EG/ACM SIGGRAPH Symposium on Geometry Processing 2004* (2004), pp. 241–250.

[Whi00] WHITAKER R. T.: Reducing aliasing artifacts in iso-surfaces of binary volumes. In *VVS '00: Proceedings of the 2000 IEEE symposium on Volume visualization* (New York, NY, USA, 2000), ACM Press, pp. 23–32.